

- [Table of Contents](#)
- [Index](#)

Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking

By [Wayne D. Grover](#)

[Start Reading](#) ▶

Publisher: Prentice Hall PTR

Pub Date: August 26, 2003

ISBN: 0-13-494576-X

Pages: 880

"Always on" information networks must automatically reroute around virtually any problem-but conventional, redundant ring architectures are too inefficient and inflexible. The solution: mesh-based networks that will be just as survivable-and far more flexible and cost-effective. Drawing heavily on the latest research, Wayne D. Grover introduces radical new concepts essential for deploying mesh-based networks. Grover offers "how-to" guidance on everything from logical design to operational strategy and evolution planning-including unprecedented insight into migration from ring topologies and the important new concept of p-cycles.

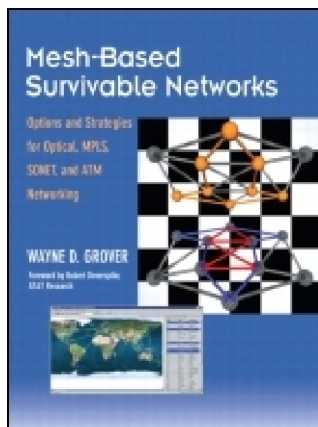
- Mesh survivability: realities and common misunderstandings
- Basic span- and path-restoration concepts and techniques
- Logical design: modularity, non-linear cost structures, express-route optimization, and dual-failure considerations
- Operational aspects of real-time restoration and self-organizing pre-planning against failures
- The "transport-stabilized Internet": self-organizing reactions to failure and unforeseen demand patterns
- Leveraging controlled oversubscription of capacity upon restoration in IP networks
- "Forcers": a new way to analyze the capacity structure of mesh-restorable networks
- New techniques for evolving facility-route structures in mesh-restorable networks
- p-Cycles: combining the simplicity and switching speed of ring networks with the efficiency of mesh networks

- Novel Working Capacity Envelope concept for simplified dynamic demand provisioning
- Dual-failure restorability and the availability of mesh networks

This is the definitive guide to mesh-based networking for every system engineer, network planner, product manager, researcher and graduate student in optical networking.

[\[Team LiB \]](#)

NEXT ▶



- [Table of Contents](#)
- [Index](#)

Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking

By [Wayne D. Grover](#)

[Start Reading ▶](#)

Publisher: Prentice Hall PTR

Pub Date: August 26, 2003

ISBN: 0-13-494576-X

Pages: 880

[Copyright](#)

[About the Book's Web Site](#)

www.ee.ualberta.ca/~grover/

[Foreword](#)

[Preface](#)

[Acknowledgements](#)

[Introduction and Outline](#)

[Historical Backdrop](#)

[The Case for Mesh-based Survivable Networks](#)

[Outline](#)

[Part 1. Preparations](#)

[Chapter 1. Orientation to Transport Networks and Technology](#)

[Section 1.0.1. Aggregation of Service Layer Traffic into Transport Demands](#)

[Section 1.0.2. Concept of Logical versus Physical Networks: Virtual Topology](#)

[Section 1.0.3. Multiplexing and Switching](#)

[Section 1.0.4. Concept of Transparency](#)

[Section 1.0.5. Layering and Partitioning](#)

[Section 1.1. Plesiochronous Digital Hierarchy \(PDH\)](#)

[Section 1.2. SONET / SDH](#)

[Section 1.3. Broadband ISDN and Asynchronous Transfer Mode \(ATM\)](#)

[Section 1.4. Concept of Label-Switching: The Basis of ATM and MPLS](#)

[Section 1.5. Network Planning Aspects of Transport Networks](#)

[Section 1.6. Short and Long-Term Transport Network Planning Contexts](#)

[Chapter 2. Internet Protocol and Optical Networking](#)

[Section 2.1. Increasing Network Efficiency](#)

[Section 2.2. DWDM and Optical Networking](#)

[Section 2.3. Optical Cross-Connects \(OXC\)](#)

[Section 2.4. Data-Centric Payloads and Formats for the Transport Network](#)

[Section 2.5. Enhancing SONET for Data Transport](#)

[Section 2.6. Optical Service Channels and Digital Wrapper](#)

[Section 2.7. IP-Centric Control of Optical Networks](#)

[Section 2.8. Basic Internet Protocols](#)

[Section 2.9. Extensions for IP-Centric Control of Optical Networks](#)

[Section 2.10. Network Planning Issues](#)

[Chapter 3. Failure Impacts, Survivability Principles, and Measures of Survivability](#)

[Section 3.1. Transport Network Failures and Their Impacts](#)

[Section 3.2. Survivability Principles from the Ground Up](#)

[Section 3.3. Physical Layer Survivability Measures](#)

[Section 3.4. Survivability at the Transmission System Layer](#)

[Section 3.5. Logical Layer Survivability Schemes](#)

[Section 3.6. Service Layer Survivability Schemes](#)

[Section 3.7. Comparative Advantages of Different Layers for Survivability](#)

[Section 3.8. Measures of Outage and Survivability Performance](#)

[Section 3.9. Measures of Network Survivability](#)

[Section 3.10. Restorability](#)

[Section 3.11. Reliability](#)

[Section 3.12. Availability](#)

[Section 3.13. Network Reliability](#)

[Section 3.14. Expected Loss of Traffic and of Connectivity](#)

[Chapter 4. Graph Theory, Routing, and Optimization](#)

[Section 4.1. Graph Theory Related to Transport Networking](#)

[Section 4.2. Computational Complexity](#)

[Section 4.3. Shortest Path Algorithms](#)

[Section 4.4. Bhandari's Modified Dijkstra Algorithms](#)

[Section 4.5. \$k\$ -Shortest Path Algorithms](#)

[Section 4.6. Maximum Flow: Concept and Algorithm](#)

[Section 4.7. Shortest Disjoint Path Pair](#)

[Section 4.8. Finding Biconnected Components of a Graph](#)

[Section 4.9. The Cut-Tree](#)

[Section 4.10. Finding All Cycles of a Graph](#)

[Section 4.11. Optimization Methods for Network Design](#)

[Section 4.12. Linear and Integer Programming](#)

[Section 4.13. Duality](#)

[Section 4.14. Unimodularity and Special Structures](#)

[Section 4.15. Network Flow Problems](#)

[Section 4.16. Techniques for Formulating LP/ILP Problems](#)

[Section 4.17. Lagrangean Techniques](#)

[Section 4.18. Other Combinatorial Optimization Methods: Meta-Heuristics](#)

[Part 2. Studies](#)

[Chapter 5. Span-Restorable and Span-Protected Mesh Networks](#)

[Section 5.1. Updating the View of Span Restoration](#)

[Section 5.2. Operational Concepts for Span Restoration](#)

[Section 5.3. Spare Capacity Design of Span-Restorable Mesh Networks](#)

[Section 5.4. Jointly Optimized Working and Spare Capacity Assignment](#)

[Section 5.5. The Forcer Concept](#)

[Section 5.6. Modular Span-Restorable Mesh Capacity Design](#)

[Section 5.7. A Generic Policy for Generating Eligible Route Sets](#)

[Section 5.8. Chain Optimized Mesh Design for Low Connectivity Graphs](#)

[Section 5.9. Span-Restorable Capacity Design with Multiple Service Classes](#)

[Section 5.10. Incremental Capacity Planning for Span-Restorable Networks](#)

[Section 5.11. Bicriteria Design Methods for Span-Restorable Mesh Networks](#)

[Chapter 6. Path Restoration and Shared Backup Path Protection](#)

[Section 6.1. Understanding Path Protection, Path Restoration and Path Segments](#)

[Section 6.2. A Framework for Path Restoration Routing and Capacity Design](#)

[Section 6.3. The Path Restoration Rerouting Problem](#)

[Section 6.4. Concepts of Stub Release and Stub Reuse in Path Restoration](#)

[Section 6.5. Lower Bounds on Redundancy](#)

[Section 6.6. Master Formulation for Path Restoration Capacity Design](#)

[Section 6.7. Simplest Model for Path Restoration Capacity Design](#)

[Section 6.8. Comparative Study of Span and Path-Restorable Designs](#)

[Section 6.9. Shared BackupPath Protection \(SBPP\)](#)

[Section 6.10. Lagrangean Relaxation for Path-Oriented Capacity Design Problems](#)

[Section 6.11. Heuristics for Path-Restorable Network Design](#)

[Section 6.12. Phase 1 Heuristics?Design Construction](#)

[Section 6.13. Putting Modularity Considerations in the Iterative Heuristic](#)

[Section 6.14. Phase 2 Forcer-Oriented Design Improvement Heuristic](#)

[Section 6.15. A Tabu Search Heuristic for Design Tightening](#)

[Section 6.16. Simulated Allocation Type of Algorithm for Design Tightening](#)

[Chapter 7. Oversubscription-Based Design of Shared Backup Path Protection for MPLS or ATM](#)

[Section 7.1. Concept of Oversubscription](#)

[Section 7.2. Overview of MPLS Shared Backup Path Protection and ATM Backup VP Concepts](#)

[Section 7.3. The Oversubscription Design Framework](#)

[Section 7.4. Defining the Oversubscription Factor \$X_{i,j}\$](#)

[Section 7.5. KST Algorithm for Backup Path Capacity Allocation](#)

[Section 7.6. Oversubscription Effects with KST-Alg](#)

[Section 7.7. Minimum Spare Capacity with Limits on Oversubscription](#)

[Section 7.8. Minimum Peak Oversubscription with Given Spare Capacity](#)

[Section 7.9. OS-3: Minimum Total Capacity with Limited Oversubscription](#)

[Section 7.10. Related Bounds on Spare Capacity](#)

[Section 7.11. Illustrative Results and Discussion](#)

[Section 7.12. Determining the Maximum Tolerable Oversubscription](#)

[Section 7.13. Extension and Application to Multiple Classes of Service](#)

[Chapter 8. Dual Failures, Nodal Bypass and Common Duct Effects on Design and Availability](#)

[Section 8.1. Are Dual Failures a Real Concern?](#)

[Section 8.2. Dual Failure Restorability Analysis of Span-Restorable Networks](#)

[Section 8.3. Determining the Network Average Dual Failure Restorability \$R_2\$](#)

[Section 8.4. Relationship Between Dual Failure Restorability and Availability](#)

[Section 8.5. Dual Failure Availability Analysis for SBPP Networks](#)

[Section 8.6. Optimizing Spare Capacity Design for Dual Failures](#)

[Section 8.7. Dual Failure Considerations Arising From Express Routes](#)

[Section 8.8. Optimal Capacity Design with Bypasses](#)

[Section 8.9. Effects of Dual Failures Arising from Shared Risk Link Groups](#)

[Section 8.10. Capacity Design for a Known Set of SRLGs](#)

[Section 8.11. Effects of SRLGs on Spare Capacity](#)

[Chapter 9. Mesh Network Topology Design and Evolution](#)

- [Section 9.1. Different Contexts and Benefits of Topology Planning](#)
- [Section 9.2. Topology Design for Working Flow Only](#)
- [Section 9.3. Interacting Effects in Mesh-Survivable Topology](#)
- [Section 9.4. Experimental Studies of Mesh Capacity versus Graph Connectivity](#)
- [Section 9.5. How Economy of Scale Changes the Optimal Topology](#)
- [Section 9.6. The Single-Span Addition Problem](#)
- [Section 9.7. The Complete Mesh Topology, Routing, and Spare Capacity Problem](#)
- [Section 9.8. Sample Results: Studies with MTRS](#)
- [Section 9.9. A Three-Part Heuristic for MTRS](#)
- [Section 9.10. Studies with the Three-Part Heuristic for MTRS](#)
- [Section 9.11. Ezema's Edge-Limiting Criteria](#)
- [Section 9.12. Successive Inclusion Heuristic](#)
- [Section 9.13. Graph Sifting and Graph Repair for Topology Search](#)
- [Section 9.14. A Tabu Search Extension of the Graph Sifter Architecture](#)
- [Section 9.15. Range Sweeping Topology Search](#)
- [Section 9.16. Overall Strategy and Applications for Topology Planning](#)

[Chapter 10. \$p\$ -Cycles](#)

- [Section 10.1. The Concept of \$p\$ -Cycles](#)
- [Section 10.2. Cycle Covers and "Protection Cycles" per Ellinas et al.](#)
- [Section 10.3. Optimal Capacity Design of Networks with \$p\$ -Cycles](#)
- [Section 10.4. Application of \$p\$ -Cycles to DWDM Networks](#)
- [Section 10.5. Schupke et al. ? Case Study for DWDM \$p\$ -Cycles](#)
- [Section 10.6. Results with Jointly Optimized \(VWP\) \$p\$ -Cycles](#)
- [Section 10.7. Heuristic and Algorithmic Approaches to \$p\$ -Cycle Design](#)
- [Section 10.8. Concept of a Straddling Subnetwork and Domain Perimeter \$p\$ -Cycles](#)
- [Section 10.9. Extra Straddling Relationships with Non-Simple \$p\$ -Cycles](#)
- [Section 10.10. Hamiltonian \$p\$ -Cycles and Homogeneous Networks](#)
- [Section 10.11. An ADM-like Nodal Device for \$p\$ -Cycles](#)
- [Section 10.12. Self-Organized \$p\$ -Cycle Formation](#)
- [Section 10.13. Virtual \$p\$ -Cycles in the MPLS Layer for Link and Node Protection](#)
- [Section 10.14. Node-Encircling \$p\$ -Cycles for Protection Against Node Loss](#)

[Chapter 11. Ring-Mesh Hybrids and Ring-to-Mesh Evolution](#)

- [Section 11.1. Integrated ADM Functions on DCS/OXC: an Enabler of Hybrids](#)
- [Section 11.2. Hybrids Based on the Ring-Access Mesh-Core Principle](#)
- [Section 11.3. Mesh-Chain Hybrid Networks](#)
- [Section 11.4. Studies of Ring-Mesh and Mesh-Chain Hybrid Network Designs](#)
- [Section 11.5. Optimal Design of Ring-Mesh Hybrids](#)
- [Section 11.6. Forcer Clipping Ring-Mesh Hybrids](#)
- [Section 11.7. Ring to Mesh Evolution via "Ring Mining"](#)
- [Section 11.8. Ring Mining to \$p\$ -Cycles as the Target Architecture](#)

[Bibliography](#)

[Index](#)

[[Team Lib](#)]

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Copyright

Library of Congress Cataloging-in-Publication Data

Wireless communications: signal processing perspectives / [edited by]

H. Vincent Poor, Gregory W. Wornell.

p. cm.--(Prentice Hall signal processing series)

Includes bibliographical references and index.

ISBN 0-13-620345-0

1. Wireless communication systems. 2. Signal processing. I. Poor, H. Vincent.

II. Wornell, Gregory W. III. Series.

TK5103.2.W5718 1998 98-9676

621.382--dc21 CIP

Editorial/production supervision *Mary Sudul*

Cover design director *Jerry Votta*

Cover design *Talar Booruji and Nina Scuderi*

Manufacturing manager *Alexis Heydt-Long*

Manufacturing buyer *Maura Zaldivar*

Publisher *Bernard Goodwin*

Editorial assistant *Michelle Vincenti*

Marketing manager *Dan DePasquale*

© 2004 by Prentice Hall PTR

Pearson Education, Inc.

Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact Corporate Sales Department, Phone: 800-382-3419; FAX: 201- 236-7141; E-mail: corpsales@prenhall.com

Or write: Prentice Hall PTR, Corporate Sales Dept., One Lake Street, Upper Saddle River, NJ 07458.

Other company and product names mentioned herein are the trademarks or registered trademarks of their respective owners.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America 1st Printing

Pearson Education LTD.

Pearson Education Australia PTY, Limited

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd.
Pearson Education Canada, Ltd.
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan
Pearson Education Malaysia, Pte. Ltd.

Dedication

To my wife, Jutta—to my son, Edward (Teddy) and to... TRLabs—still the noble experiment

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

About the Book's Web Site

www.ee.ualberta.ca/~grover/

A number of appendices and supplements are web-based, so that they may be kept current and easily updated or extended as desired. This includes:

- Glossary
- AMPL models to implement design formulations.
- DATPrep programs: These are programs that can be used as is or adapted to new problems for the creation of network specific DAT files that required for execution of the AMPL models.
- A library of test-case network and demand files.
- A library of programs for basic functions such as routing or cycle enumeration.
- Frequently Asked Questions and discussion on survivable networking issues.
- Errata for the book.
- Technical Reports produced by the authors research group (as available).
- Recommended Links.
- Selected Lecture Notes on Survivable Networks.
- MeshBuilder: prototype versions of a mesh-based planning and analysis tool.
- Student Problems and Research Projects

The web site is <http://www.ee.ualberta.ca/~grover/>

Follow the link to "Mesh-Based Survivable Networks." Access to the book's web resources requires the user to have a copy of the book on hand. In future, the URL will change to <http://www.ece.ualberta.ca/~grover/>.

Foreword

Let me first speak to the topic of mesh-based survivable networks—its history and importance, and then add my comments on the significance of this first comprehensive book on the topic.

One milepost in our society's evolution towards higher expectations of telecommunications network reliability is illustrated by a press release in the early 1990s (paraphrased):

"In response to several consecutive major telephone network outages and at the request of key Members of Congress, on Sept 19, 1991 the FCC issued a Notice of Proposed Rulemaking requesting that all facilities-based common carriers report any major outages that affects over 50 thousand customers for more than ½ hour [eventually reduced to 30 thousand customers]."

In the years prior, networks in the United States mostly consisted of voice (telephone) calls carried over circuit switches. Outages of the underlying transport network were handled at the circuit-switched layer. Most of these remedial actions consisted of forced alternate routing through manual switch overrides by network operators in central Network Operations Centers (NOCs). Over time the capacity of transport networks increased, data overlay networks from other competitive carriers were created, and large end-users instituted private voice and packet networks. The links of such switched overlay (or "service") networks were transported as circuit-based services by carriers and thus the private line business emerged.

The result was that a large carrier transported links of its own switched voice and data overlay networks, as well as the links of other service-layer networks. It became limiting to react to transport network failures only in the service network. From the perspective a customer with its own service-layer network who leases its links from the carrier, there is no knowledge of the physical layer over which it routes its links and thus it is difficult to plan or react to network failures. Conversely, from the perspective of the carrier, it is unreasonable to require that the overlay network handle transport layer failures: the switches of these networks are usually unknown and outside the span of control of the carrier. Furthermore, as transport network bandwidth increased, it became clear that the restoration of services affected by transport failures would be more expensive to do in the service (overlay) networks than at the transport layer itself. At the transport layer multiplex bundling and better economies of scale could be achieved. It thus became clear that automatic restoration should be offered at the transport layer itself.

For the first digital transmission systems (T1-T3) restoration or "protection" was provided on a 1-to-N ratio (1:N) where one standby system would be switched in to protect against failure in any one of the N other systems. However, the protection channel usually was "in-line"—it resided in the same conduit or cable as the working channels and thus provided little protection against cable cuts. But the impact was minimal because, in the early days, T1 cables did not carry many circuits and businesses had not developed a reliance on high network availability. So the failure had minimal impact or, as importantly at least, it garnered little interest from the press.

As fiber optic transmission and eventually Wavelength Division Multiplexing emerged, the bandwidth of a single fiber soared. With so much capacity and relatively failure-prone laser and electro-optic devices, 1:1 protection was required for the transmitter and receiver line cards. The same "in-line" protection methods were expensive compared to copper or coax -based T1-T3 systems, yet ultimately ineffective from a restoration standpoint. Thus, the deployment of 1:1 protection with diverse geographic routing of the protection path evolved. This solution was tolerable for a while, but as demand for bandwidth and transmission rates grew further, the economics of many overlapping point-to-point transmission systems, each with its own dedicated diverse fiber path, became unattractive.

In the 1990s, Bellcore developed the SONET standard and standardized the concept of self-healing rings, quickly followed by its international twin standard, SDH. Transmission engineers rejoiced. This appeared to be the final answer: one could replace all these cumbersome and expensive point-to-point transmission systems with a few multi-node, self-healing SONET rings. In contrast, AT&T decided to address the growing restoration problem for its long distance network with one of the first automatic centrally-controlled mesh-restoration schemes, called FASTAR, based on DS3 Digital Cross-connect Systems (DCSs). Its restoration speed would not match the eventual SONET rings, but by prioritizing restoration of private line services (especially 1-800 services) it was more than adequate. Moreover, it proved to be very economic in its use of extra transport capacity needed for restoration, often termed the "restoration overbuild" (—what Grover calls the "spare" capacity).

Many other carriers jumped on the SONET ring bandwagon. This started off an industry debate that rages to this day. Which is better,

ring or mesh-based restoration? Rings seemed to win the battle through the mid 1990s, until increasing bandwidth again crossed a threshold. Many network researchers found multiple, overlapping SONET rings to be economically unattractive compared to DCS mesh restoration in certain types of networks. However, compared to the early forms of centrally-controlled mesh restoration (where DCSs have no distributed routing intelligence or inter-nodal signaling), rings had the advantage that once the ring was installed, restoration is automatic. In contrast, the centrally-controlled mesh-restoration schemes required a sophisticated central system with a separate signaling network provided between the NOC and the DCSs. This operational difference provided the motivation for the idea of intelligent DCSs, that is, network elements capable of distributed routing, detecting failures of their links, and passing topology, path set-up, and routing messages among themselves via inter-nodal signaling. This was not a unique idea, in that it emulated distributed data networks, but the techniques for distributed restoration would be fundamentally different in some important ways.

In a visionary role and long before this debate peaked, Wayne Grover was the first to observe that the ideas of self-routing in data networks could be extended and applied to circuit switched transport networks in his seminal paper, *The Selfhealing Network: A Fast Distributed Restoration Technique for Networks Using Digital Cross-Connect Machines* that appeared in Globecom in 1987. However, it was not until the technological advent of the intelligent DCS (now enhanced with Gigabit rate Optical interfaces) in the late 1990s that this debate shifted. AT&T again led the mesh-restoration charge with its decision to implement the Intelligent Optical Switch (IOS) for its long distance transport network at the turn of the new century. This decision resulted from the new, automatic restoration capabilities of intelligent cross-connects with compact and integrated Gigabit-rate optical interfaces, years of network optimization studies at AT&T Labs of rings versus mesh network design, and the final determination that mesh wins economically in long distance networks of sufficiently large demand for bandwidth, size, and geographically diverse path connectivity.

As of this writing, SONET rings still dominate in metropolitan networks, where there is less geographic diversity and less required bandwidth than in intercity networks, but we find that intelligent cross-connects are encroaching further into metro networks as well. Also, as optical cross-connect and WDM switching technologies mature, mesh-based restoration for pure optically-switched networks are under increased interest because of the reduced costs of optical-electrical conversion and economies of scale for integration of WDM and electronics. This option is especially attractive to transport the very-high rate links of IP service-layer networks.

This brings us to the important contribution that this book now makes to the field. During this evolution, many different alternatives for mesh networking have evolved. It is important to understand them and to master the tools needed to evaluate how they work, how to design the network to meet the reliability objective, where they are best deployed, and how to compare alternative architectures and protection methods. This book is much more than just a survey of these topics, but rather forms a series of in-depth tutorials based on the author's own internationally-recognized research. In particular, Grover introduces a large amount of previously unpublished material and, for those published topics, he provides insights, depth, and explanatory discussions beyond that of the original publications.

For example, there are single chapters alone that will prove invaluable to network operators and their equipment suppliers, such as how to design and operate span-restorable and path-restorable networks, how to design ring-mesh hybrid networks, how to analyze the availability of mesh networks under multiple failures, and how to evolve ring networks into a mesh-restorable network. Besides providing invaluable background in these "classic" areas, the book provides a wealth of new options and alternatives that have exploded recently upon this field, all developed by Grover and his team of graduate researchers at TRILabs—University of Alberta. A few of the original research areas treated in-depth in this volume include p -cycles, forcer-analysis, distributed pre-planning, and self-organization of transport networks. Grover covers restoration in packet-based networks as well, including topics such as the controlled oversubscription design of MPLS networks, which is of particular research interest to me in recent work on the convergence of transport and packet networks. Other new topics include the "meta-mesh" technique for very sparse networks, and the "working capacity envelope concept."

My own work over the years has demonstrated that efficiency in restoration and design of transport and packet networks saves hundreds of millions of dollars in capital expense in carrier networks—and the concomitant increase in intelligence of network elements results in equally significant operational savings. Based on this, I highly recommend this book to network operators and their equipment suppliers. As for graduate students and neophytes to transport networks, I find the first four preparatory chapters alone to constitute a reference book on graphs/networks, transport network architectures, and network routing and optimization algorithms, plus the "Studies" chapters contain many further ideas for research projects.

In conclusion, the quality and originality of work from Grover and his group is known world-wide on the topic of mesh-based network survivability—a topic for which he was signified as an IEEE Fellow in 2001. Therefore, I think I can speak for those of us who have contributed to the field, both in theory and practice, and state that no one is better qualified than Grover to write the first comprehensive book on this topic. I whole-heartedly commend this timely and valuable volume to you.

Dr. Robert D. Doverspike
Director, Transport Network Evolution Research
AT&T Labs - Research
Middletown, NJ

June 6, 2003

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Preface

"Internet hindered by severed cable."

"Internet chaos continues today..."

"Cable break disrupts Internet in several countries."

"Calling and major businesses down from cable cut."

These are headlines arising in just one month from fiber optic cable disruptions. Despite the enormous advantages of fiber optics and wave-division multiplexing, the truth is that the information economy—fueled by fiber-optic capacity—is based on a surprisingly vulnerable physical medium. Every effort can be made to protect the relatively few thumb-sized cables on which our information society is built, but the cable-cuts and other disruptions just don't stop. From deep-sea shark bites to the fabled "backhoe fade," serious transmission outages are common and of increasing impact. Some form of fast rerouting at the network level has become essential to achieve the "always on" information networks that we depend upon.

One way to survive optical network damage is to duplicate every transmission path. In the form of rings and diverse-routed protection switching schemes, this is actually the most widespread solution in use today. Our view has long been that this is an inefficient expedient—the "get a bigger hammer" approach to solving a problem. Admittedly, rings filled the void when survivability issues reached crisis proportions in the 1990s. But now network operators want options that are just as survivable but more flexible, more growth-tolerant, able to accommodate service differentiation, and *far* more efficient in the use of capacity. This is where "shared-mesh" or "mesh-restorable" networks take the stage.

The author's love affair with the mesh-survivability approach began in 1987—with the naive certainty back then that sheer elegance and efficiency would suffice to see it adopted in SONET by 1990! The actual journey has been much longer and complicated than that. The Internet and Optical Networking had to happen first. And all the ideas had much more development to undergo before they would be really ready for use. Today, we understand the important values of mesh-based survivability go beyond just efficiency (and of course you can rarely make money off of elegance alone). Flexibility, automated provisioning, differentiated service capabilities, and the advent of Internet-style signaling and control are all important in making this a truly viable option. But its full exploitation still requires new concepts and ideas about network operation—letting the network self-organize its own logical configuration, for example, and letting it do its *own* preplanning and self-audit of its current survivability potential. New planning and design models are also required. These are the central topics of this book—designed to stimulate and facilitate the further evolution toward highly efficient, flexible and autonomous *mesh-based* survivable networks.

The book is written with two main communities in mind. One is my colleagues in industry; the system engineers, research scientists, technology planners, network planners, product line managers and corporate technology strategists in the telcos, in the vendor companies, and in corporate research labs. They are the key people who are continually assessing the economics of new architectural options and guiding technology and standards developments. Today these assessments of network strategy and technology selection put as much emphasis on operational expense reduction as on capital cost—capacity efficiency *and* flexibility are both important in future networks. Network operators are in an intensely competitive environment with prices dropping and volumes rising and success is dependent on all forms of corporate productivity enhancement. Mesh networking can provide fundamental productivity enhancements through greater network efficiencies and flexibility. The book aids the operating companies in finding these new efficiencies by giving many new options and ideas accompanied with the "how to" information to assess and compare the benefits on their own networks. Examples of the new directions and capabilities the book provides are in topology evolution, ring-to-mesh conversion by "ring-mining," multiple Quality-of-Protection design, tailoring restoration-induced packet congestion effects in a controlled manner, simplifying dynamic demand provisioning, and so on. An important plus is that the book also contains the first complete treatment of the intriguing and promising new concept called "*p*-cycles"—offering solutions with ring-speed *and* mesh-efficiency.^[1]

[1] There was thought about a separate book on *p*-cycles alone, but it seemed more important to get the information out without further delay. That, in part, has been the cause of a bigger book than initially planned.

Providers of the networking equipment, the vendors, are—as the saying goes—“fascinated by anything that interests their boss.” That means their network operating customers. Vendors must not just keep in step with the problems, opportunities, and thinking of their customers, but also aspire to bring their own unique equipment design strategies to the market and to provide leadership in development of advantageous new networking concepts. The vendor community will therefore be especially interested in the techniques for split-second mesh restoration and self-organizing traffic-adaptation as features for their intelligent optical cross-connects and Gigabit routers, for instance—as well ideas for new transport equipment such as the straddling span interface unit that converts an existing add/drop multiplexer into a p -cycle node. All of the other topics covered are of interest to vendors too because they enhance their ability to assist customers with network planning studies as part of the customer engagement and sales process.

Developers of network modeling, simulation and planning software will also be interested in many ideas in the book. By incorporating capabilities to design all types of architecture alternatives or, for example, to simulate dynamic provisioning operations in a protected working capacity envelope, or to model the incremental evolution of a survivable capacity design in the face of uncertain demand, or to support ring-mining evolutionary strategies—these suppliers enable their customers to pursue a host of interesting new “what if” planning studies.

The second main community for whom the book is intended is that of graduate-level teaching and research and new transport networking engineers who want a self-contained volume to get bootstrapped into the world of transport networking planning or to pursue thesis-oriented research work. A principle throughout has been to draw directly on my experience since 1992 at the University of Alberta of teaching graduate students about survivable transport networking. This allowed me to apply the test: “What needs to be included so that my graduate students would be empowered both to do advanced investigations in the area, but also to be knowledgeable in general about transport networking?” On one hand, I want these students to be able to defend a Ph.D. dissertation, but on the other hand also to have enough general awareness of the technology and the field to engage in discussion with working engineers in the field. This is really the reason the book has two parts.

The test of needed background has guided the definition of [Chapters 1 to 4](#) which are called the “Preparatory” chapters. These chapters cover a lot of generally useful ground on IP and optical technology, routing algorithms, graph theory, network flow problems and optimization. Their aim is to provide a student or new engineer with tools to use, and an introductory understanding of issues, trends, and concepts that are unique to *transport* networking. In my experience, students may have done good theoretical research, but at their dissertations a committee member may still stump them with a down-to-earth question like “How often do cables actually get cut?” “Is this just for SONET or does it work for DWDM too?”, “How does restorability affect the availability of the network?” or (perennially it seems) “...but I don't see where are you rerouting each phone call or packet.” These few examples are meant just to convey my philosophy that as engineers we should know not only the theory, the mathematical methods, and so on, to pursue our “neat ideas” but we also need to know about the technology and the real-world backdrop to the research or planning context. This makes for the best-prepared graduate students and it transfers to the training of new engineers in a company so that they are prepared to participate and contribute right away in all discussions within the network planning group he or she joins. An engineer who can, for example, link the mathematics of availability analysis to a contentious, costly, and nitty-gritty issue such as how deep do cables have to be buried, is exactly the kind of valuable person this book aims to help prepare. Someone who can optimize a survivable capacity envelope for mixed dynamic services, but is also savvy enough to stay out of the fruitless “50 ms debate” is another conceptual example of the complimentary forms of training and knowledge the book aspires to provide.

The book is ultimately a network planners or technology strategists view of the networking ideas that are treated. It employs well-grounded theoretical and mathematical methods, but those are *not* the end in itself. The book is also not filled with theorems and proofs. The emphasis is on the network architectures, strategies and ideas and the benefits they may provide, not primarily on the computational theory of *solving* the related problems in the fastest possible way. Our philosophy is that if *networking* ideas or science look promising, then the efforts on computational enhancement are justified and can follow. Fundamental questions and ideas about *networks*, and network architecture, (which is the main priority in my group) stand on their own, not to be confused with questions and ideas about *algorithms and solution techniques* to solve the related problems as fast as possible (others are stronger in that task). Obviously work in this area involves us in both networking science and computation, but the logical distinction is important—and often seems lost in the academic literature.^[2] The book is also not a compendium or survey of previously published papers. While suitably referenced, its content is unabashedly dominated by the author's own explorations and contains a large amount of previously unpublished material. Although setting the context in terms of modern transport technologies (WDM, SONET, ATM, IP, MPLS) our basic treatment of the networking ideas and related planning problems is in a generic logical framework. The generic models can be easily adapted for to any specific technologies, capacities, costs, or signaling protocols, etc. The book thus provides a working engineer or a new researcher with a comprehensive, theoretically based, reference book of basic architectural concepts, design methods and network strategy options to be applied on mesh-survivable networks now and in the future.

[2] For instance an *algorithm* for optimal p -cycle *design* might be NP-hard, but it would be nonsense to say that p -cycles *themselves* are complex because of this. One is an algorithm, the other is a network architecture. In the same spirit—we tend to say “*so what*” if Integer Programming is *theoretically* NP-hard—in practice we can solve enormously large and useful problems with it in 15 minutes! And if not, *then* we pursue whatever else we have to do. But networking insight is the *end*, computation is only the *means*.

A few words about the flow of the book. The Introduction gives a much fuller roadmap of the content and novelty in each chapter. Briefly, however, [Chapter 1](#) is an orientation to transport networking. [Chapter 2](#) is devoted to background on IP and DWDM optical networking developments, as the technological backdrop. Part of [Chapter 3](#) is partly just "interesting reading" on the effects of failures and the range of known schemes and techniques to counteract or avoid failures. The rest of [Chapter 3](#) includes a more technical "sorting out" of the "*—ilities*": *availability*, *reliability*, *network reliability*, *restorability*, and other measures. [Chapter 4](#) is then devoted to graph theory, routing algorithms and optimization theory and techniques but only as these topics specifically relate to transport network problems. [Chapter 5](#) starts the second part of the book on more advanced studies and applications with an in-depth treatment of span protection and restoration. It has its counterpart devoted to path restoration in [Chapter 6](#). [Chapter 5](#) considerably "updates" the thinking about span-oriented survivability in optical networks with dynamic traffic.

If the book was a musical score, [Chapters 7](#) through [11](#) would be the "variations." Each chapter treats a more advanced topic or idea selected by the author because of its perceived usefulness or possible influence on the direction of further research and development.

These are some of the author's "shiny pebbles" (in the earnestly humble sense of Newton^[3]). [Chapter 7](#) recognizes an important difference—and opportunity—in cell or packet-based transport: that of controlled oversubscription of capacity upon restoration. This is a unique advantage for MPLS/IP-based transport survivability. [Chapter 8](#) is devoted to all aspects of dual-failure considerations in mesh restorable networks. An especially interesting finding is that with a "first-failure protection, second-failure restoration" concept, *higher than 1+1 availability* can be achieved for premium service paths at essentially no extra cost. [Chapter 9](#) treats the challenging, and so far almost unaddressed, problem of optimizing or evolving the basic facility-route (physical layer) topology for a mesh-restorable network. [Chapter 10](#) explains the new (and to us, very exciting) concept of *p*-cycles, which are rooted in the idea of pre-configuration of mesh spare capacity.

[3] A quote attributed to Newton paints a humbling but touching image—that all of us (researchers) are as yet like children on a beach—calling out to each other about the "shiny pebbles" we have found. He said: "I have been only like a boy, playing on the sea-shore... now and then finding a smoother pebble or a prettier shell, while the great ocean of truth lay all undiscovered before me."

p-Cycles are, in a sense, so simple, and yet they combine the fast switching of ring networks with the capacity-efficiency of mesh-based networks. We include *p*-cycles as a mesh-based survivable architecture because they exhibit extremely low mesh-like capacity redundancy and because demands are routed via shortest paths over the entire facilities graph. They are admittedly, however, a rather unique form of protection scheme in their own right in that lies in many regards in-between rings and mesh. Candidly, I venture that many colleagues who went through the decade-long ring-versus-mesh "religious wars" of the 1990s would understand when I say that *p*-cycles call for that forehead-bumping gesture of sudden realization—this solution (which *combines* ring and mesh) was unseen for the whole decade-long duration of this debate! As of this writing the author knows several research groups that are shifting direction to work on *p*-cycles as well as a half-dozen key industry players looking closely at the concept.

[Chapter 11](#) on ring-mesh hybrids and ring to mesh evolution is placed at the end. The logic is that if we assume success of the prior chapters in motivating the mesh-based option then the "problem" this creates is that many current networks are ring-based. Its like "Ok, we believe you—but how do we get there now?" The closing chapter therefore devotes itself to bridging the gulf between existing ring-based networks and future mesh or *p*-cycle based networks by considering the design of intermediate ring-mesh hybrid networks and "ring-mining" as a strategy to get to a mesh future from a ring starting-point today.

The Appendices, and other resources such as chapter supplements, a glossary, student problems, research project ideas, network models, and more are all web-based—so they can be continually updated and expanded in scope and usefulness. Many directly usable tools and resources are provided for work in the area of mesh-survivable networking. This includes AMPL models and programs to permit independent further study of most of the planning strategies presented, plus Powerpoint lectures on a selection of topics, technical reports, and additional references and discussions. The aim has been to create a highly useful and hopefully interesting book that is laden with new options, ideas, insights and methods for industry and academia to enjoy and benefit from.

Wayne D. Grover,
TRLabs and the University of Alberta, Edmonton, Canada.

July 11, 2003

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Acknowledgements

This project reflects contributions and support from many people and a few key organizations. First and foremost, I have been especially fortunate in the four years that this book was in preparation to work with TRILabs leading an outstanding group of likable, highly capable, and seemingly inexhaustible graduate students who have shown terrific enthusiasm for this book and the related EE 681 course at the University of Alberta. Just as the commercial says—these are the people that "live, breathe and eat this stuff"—they love it and they're experts at it too. Among those that contributed—not only with repeated proofreading, suggestions, questions and comments, but in most cases also with direct contributions of sample results or other data and/or diagrams for inclusion in the book are:

- John Doucette, Ph.D. candidate and TRILabs Research Engineer
- Matthieu Clouqueur, Ph.D. candidate and TRILabs Research Engineer
- Anthony Sack, M.Sc. candidate
- Gangxiang Shen, Ph.D. candidate
- Govind Kaigala, M.Sc. candidate
- Adil Kodian, M.Sc. candidate
- Dion Leung, Ph.D. candidate
- Dominic Schupke, Doctoral candidate, Technische Universitat Munchen

Specific papers and materials related to collaboration with these and other students are noted throughout the book. But I want to make special mention of John, Matthieu and Anthony. In the course of working at the book (off and on) over the last 4 years, I would repeatedly call up John or Matthieu, Anthony too, and begin a conversation with: "I wonder if we could....—". Almost every conversation led to production of some original new test-case results or other experiment, data or drawings to be used in the book. Often it led to whole projects that these students carried out enthusiastically and capably and were then synopsized in the book and/or became the basis for whole separate research papers. Collaborations with John to produce results and other material in [Chapters 5](#) (Span Restoration) and [9](#) (Topology) were especially fruitful, but John's hand is found in [Chapters 6, 8](#) and [10](#) as well. Work with Matthieu, especially on multi-QoP and mesh availability is found in [Chapter 5](#) and is central to [Chapter 8](#). And by now Anthony must feel wedded to the p -cycles chapter, having been through it with a fine-tooth comb, and been an intimate collaborator on the Hamiltonian-related aspects of p -cycles. I deeply appreciate his critical-thinking skills, and time spent providing exceptionally thoughtful, detailed markups of most chapters. I can hardly praise enough the skill and tirelessness of these three in supporting and enriching this project.

Another group of students have finished and moved on, but left theses and other materials from investigations we undertook, that I have adapted, summarized, or otherwise blended into the mix. Material I selected was—in my view—useful or valuable work, or good tutorial background, that was previously published only in theses or documented in internal reports or notes. In other cases I have drawn upon, or extended, the unpublished results of past collaborations with visiting researchers or TRILabs Research Staff—bringing culmination to some "good stuff" that was never published. Beyond the normal use of citations for acknowledgement, I want to especially mention:

- Dave Morley, former Ph.D. student and now TRILabs Director of Business Development
- Demetrios Stamatelakis, former M.Sc. student and now TELUS Research Engineer
- Jim Slevinsky, former M.Sc. student and now with TELUS Technology Strategy
- Rainer Iraschko, former Ph.D. student and management of ONI and Network Photonics.
- Chioma Nebo (nee Ezema), former M.Sc. student now with Shell Petroleum
- Brad Venables, former M.Sc. student now with Nortel Networks

- Yong Zheng, former M.Sc. student
- Mike MacGregor, former Ph.D. student and now U. of A. Assoc. Prof. Computing Science
- Graeme Brown, BT Labs, Visiting Researcher to TRILabs
- Oliver Yang and Donna He, SITE, University of Ottawa
- Mandana Asadi, former M. Eng. student and with Rogers Communications
- Kent Lam, former M.Eng student
- Matthias Scheffel, former exchange student Technische Universitat Munchen
- Nelly Hamon, former ENSEA (France) exchange student
- Chee Yoon Lee, former M.Sc. student and with Nortel Networks
- Ernest Siu, former M.Sc. student now with Yotta Yotta
- Dave Allen at MCI
- Kent Felske, Jeff Fitchett, Alan Graves and others at Nortel Networks

Organizations?— TRILabs and the University of Alberta! Many thanks for the environment that made the book project possible as well as all of the research that fed into it. TRILabs and its sponsors have fostered the 17 years of research in transport networks that is behind this work. To TRILabs and its leaders, present (Dr. Roger Pederson) and past (Glenn Rainbird and Ray Fortune), I can only say "Thanks for persisting—and believing." The TRILabs story is a terrific example of vision and success in industry-university collaboration (please see www.trilabs.ca). I am just pleased to add this book to the list of TRILabs' accomplishments. The many years of interacting with people at TRILabs sponsor companies—especially Nortel, TELUS, MCI, SaskTel, but others as well—have also been a tremendous benefit. It feels tough when they don't immediately accept our ideas, and they challenge our thinking. But this improves the "product" immensely in the end and fuels the engine of further work. It forces my students and I to refine our understandings of the issues and always work harder to combine academic depth with practical relevance. So, to all those TRILabs companies over the years that patiently heard our presentations, explained what we were missing, and suggested directions for the next steps—Thank you! Also my sincere appreciation goes to Linda Richens at TRILabs who has so capably and efficiently handled many TRILabs administrative and management-support tasks that maximize my time available for research, students, sponsors, and writing.

The University of Alberta in turn fosters TRILabs and my academic position that led to this book. I am especially grateful for the 1999/2000 sabbatical year that allowed me to launch this project and to work in residence with Level(3)—(Thanks there to Lorraine Lotosky, Linos Frantzeskakis, Russ Rushmeier, Robert Feuerstein and others) and to give specialty lectures in the area at the University of Colorado—(Thanks there to Prof. Frank Barnes). Back at home, ECE chair Witold Pedrycz, has been constantly encouraging. Reminders that he "wants a copy of the book to display in the cabinet" were a clever and gentle device to encourage me to *finish* —to visualize it in the cabinet. I also want to thank NSERC. Through the Discovery Grant program and added funding and time from the E.W.R Steacie Fellowship, combined with TRILabs, I was able to build a team of 14 students, all working on topics in transport networking.

At Prentice-Hall, Bernard Goodwin feels like my "book-writing father" now! Bernard and I went on an odyssey where *he* taught *me* how to get a book written and—believe it or not—*I* made him into an advocate for mesh-based networking! Bernard explains this book to others in the publishing field as well as I could now. But I'd do a lot of things differently next time—having learned a lot, thanks to Bernard. I also salute him for his leadership on some tough decisions. I had started out on an even bigger (but unrealistic) initial plan—Bernard saw that first and got me to realize it. Then the economy was terrible while the book was written, and telecom was down the most. Add to that, that this is a *big* book *and* it is *two years* overdue—so I really appreciate Prentice-Hall's (read Bernard's) faith in the project, and patience and persistence in getting what this mesh-networking stuff is, and where it fits in. As of writing these notes, my main journey with Mary Sudul is not over yet—the production process. Lots of wrestling with PDF! Thanks Mary, for all the editorial markups, advice, cover design, and FrameMaker tips so far!

That leaves the home-front. Thanks Jutta, especially for your shared experience with scientific writing, and with helping me realize I had to *stop* working on it at some point. Thanks also to both Jutta and Teddy for all the nights I was over at the office with the lights on late, or at the computer at home. Teddy—this is what I was doing all that time. I hope you like it!

Wayne D. Grover,
TRILabs/University of Alberta,

July 12, 2003

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Introduction and Outline

Network survivability is a fascinating topic, both as an area of technical study and because it is not removed from our everyday lives—a failure can affect our plans, in some cases even our lives, fairly directly. The growth of the Internet, the increasing number of "mission critical" business functions that rely on communication networks, and the emergence of general societal dependencies on communications, all make survivability a now-essential, but only fairly recently considered, aspect of network design. With up to 100 terabits per second of data flowing through a single fiber with DWDM, failure can have catastrophic and far-reaching consequences. And such events, cable-cuts in particular, are surprisingly frequent. In the first eight months of 2002 alone, the FCC logged 116 network outages in the United States with wide-ranging effects and often peculiar causes. On February 13 in Yadkinville, NC, town workers severed a Sprint cable while repairing a water line, cutting 52 trunk groups and 13 DS-3 links for over 5 hours. A week later a fire in a Maryland power transformer melted a Verizon fiber cable affecting 5000 customers for over 9 hours. On March 14, a contractor accidentally cut functional fiber during removal of retired cable, cutting 911 service to a part of San Diego for over 4 hours. Despite considerable efforts at physical protection of cables, FCC statistics are that metro networks annually experience 13 cuts for every 1000 miles of fiber, and long haul networks experience 3 cuts for 1000 miles fiber. The numbers may sound small, but even the lower rate for long-haul implies a cable cut *every four days* on average in a network with 30,000 route-miles of fiber. And such failures are having increasing societal impact. Consider the following estimate of the direct revenue impact alone:

"Through 2004, large U.S. enterprises will have lost more than \$500 million in potential revenue due to network failures that affect critical business functions." [Gartner Group, 2002]

We are now almost as dependent on the availability of communications networks as on other basic infrastructure like roads, water, and power. The phrase "mission critical business functions" has even been coined to refer to applications that must be running and available over communication networks 24 hours a day, seven days a week, for the related businesses to survive. Put together, these factors mean that survivability must be a foremost consideration in the basic design of any network, not an afterthought.

Despite considerable efforts to physically protect network elements—fiber optic cables in particular, the real world is surprisingly creative in finding ways to cut them. Who would have guessed failure of the TAT-8 undersea fiber system from deep-sea shark bites? Who would have foreseen the loss of air-traffic control at JFK airport at the end of a causality chain beginning with street works that cut a cable? Who would have foreseen youths building a fire under a bridge abutment—below the cable trays? On the other hand, imagine the quiet excitement of the engineer who watched his network react spontaneously, literally in the "blink of an eye," completely hiding from 129,000 users the fact that an ice-storm had finally pulled down the most burdened aerial cable section. Perhaps especially for engineers, it is fascinating to learn about the often elegant concepts that, by-and-large are unseen by the public, but underlie the basic infrastructures of our lives. This book is about one of the most important of those infrastructures—optical transport networks—and a range of techniques to make them withstand such failures *by design*, and as efficiently as possible.

Historical Backdrop

Telecommunication equipment makers, carriers, and users have historically been concerned with reliability, but not with such an intense focus as in recent times. Previously, attention was paid to ensuring certain levels of *availability*^[1] in network elements, through redundant design at the *equipment* level. Operators would estimate service availability over such network elements configured in series "hypothetical reference path" configurations. The situation could be described as one where the transmission *equipment* was redundantly designed, but not the *network* as a whole. If a failure arose that overcame these measures, a service outage would occur until repair was completed, or until a manual effort at partial rerouting was completed. A carrier would measure performance by the overall annual fraction of time a hypothetical reference path would be in the "up" state, but there was no expectation of *split-second* recovery times against a major failure such as a cable cut.

^[1] A technical concept to which we will return.

Today the goal, and increasingly the reality, is one of virtually instantaneous recovery against the most significant and frequent types of failure. What changed to cause this escalation in our expectations? Fiber optic technology, deregulation and competition, and unprecedented growth in the use of communication service, especially Internet-based services and applications, are perhaps the three main factors. Optical networks based on wave-division multiplexing over fiber optics offer huge point-to-point capacities—over a terabit per second in total over each fiber. The advantages of optical fiber as a transmission medium include low loss, light weight, electromagnetic immunity, high bandwidth, low cost, and so on. But a humbling reality is that no matter how advanced the fiber and system technology becomes, this is a *cable-based* technology—it goes in the ground or on poles, or it lies on the ocean bottom. In all cases it is surprisingly vulnerable. With thousands of route-kilometers of fiber deployed in national or regional networks, cable cuts and other line-related disruptions are an operational *certainty*. Consistent with the FCC data given for the U.S.A., Hermes, a pan-European "carrier's carrier" has independently estimated an average of *one cable cut every four days* on their network. At the same time, the high capacity and economy of scale of fiber optics at higher transmission rates drives operators toward relatively sparse backbone topologies, making each cable section even more important to the network as a whole.

Deregulation in the 1980s also led to pell-mell competition between incumbents and new entrants in the 1990s. This may have contributed to some of the headline-making failures of the era but it also made survivability a selling point, especially in competition for large corporate users and backbone Internet providers. The costs of redundancy for survivability can be very high, however, compared to a corresponding network designed only to serve the working demands under nominal conditions. Without careful choices of architecture and design methods it is easy to find the costs of a survivable network approaching *twice* the cost of a non-survivable network. This is a considerable motivation to look at new ways of designing and operating a survivable transport infrastructure.

Mesh survivability schemes can be viewed in one sense as the extension and automation of the essentially manual restoration methods used in the 1970s and early 1980s. In that era, 1:N or 1+1 automatic protection switching (APS) would often be designed into the transmission systems, but there would be no automated means for restoration from a complete facility cut. A great deal of the backbone transport network of the time was based on microwave radio that inherently has high structural availability. Individual equipment items could experience failures and radio paths could suffer from fading, but the APS systems would take care of that. When the need for restoration did arise, it was generally handled on a best-efforts basis, manually, by rearrangements at the DSX-3 patch panels. The DSX-3 panel was a manual patch board, similar to the old operator boards in concept, at which DS-3 level rearrangements of signals on and off of the transmission systems could be made. The process of rearranging connections between equipment with manual patch cords was called "cross-connection" and gives its name to the automated optical or SONET layer cross-connect systems of the current era.

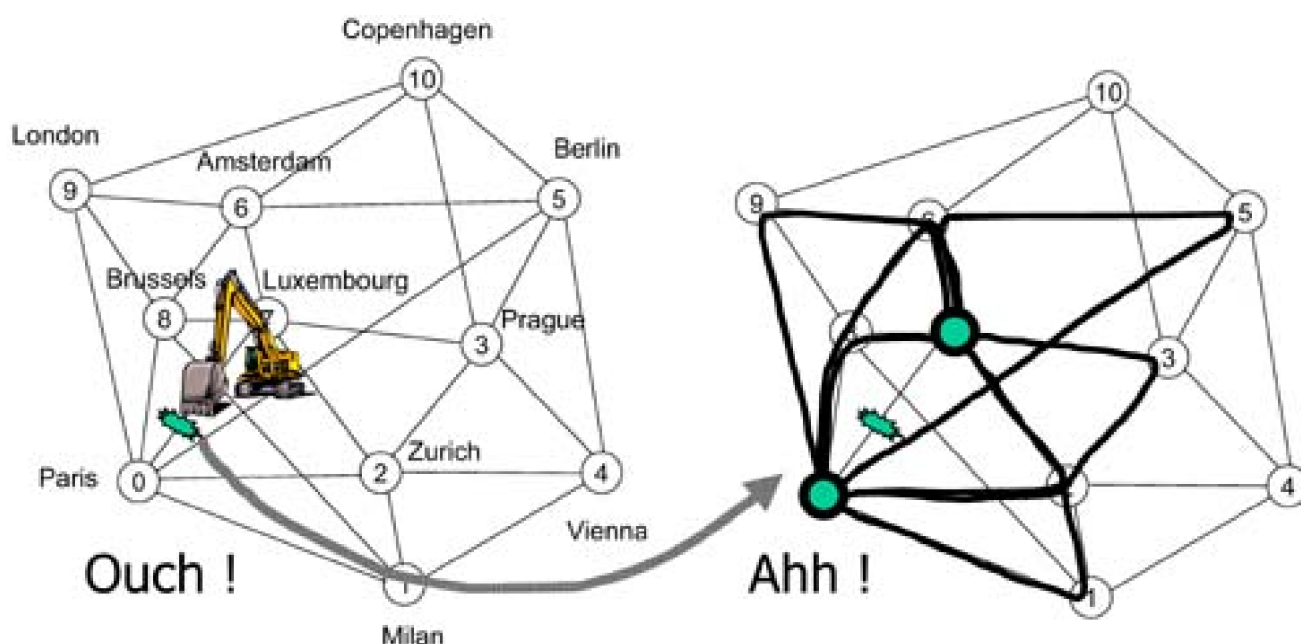
It is the human process of inspecting the network state at the time of the failure, and developing a rerouting or "patch plan," that is the origin for the concept of mesh-restorable networks. The differences are that now we will carefully design-in specific amounts of spare capacity, with various theories for doing so, and embed real-time mechanisms to implement or even develop the "patch plan" itself, in a fraction of second. But the basic concepts are the same as when humans, knowing their network well, would work up an ad-hoc scheme of patching around the fault. The spare capacity would come from a variety of sources—some of it true spare channels, some of it from the spare spans of APS systems, and some of it from bumping low-priority traffic and/or the use of satellite transponders on a pre-contracted basis for restoration.

The Case for Mesh-based Survivable Networks

Through the 1990s the industry vigorously debated ring versus mesh-based principles for survivable transport. Rings greatly predominated in practice, however, because the urgency for a "fix" to the survivability problems (which reached crisis proportions about 1994) was so great, and, in terms of development costs and time, ring systems were relatively easy extensions of existing point-to-point transmission systems with APS. (One of the first commercial ring-protected transport system, a precursor to the BLSR, was actually just existing FD-565 transmission systems each with 1+1 APS, simply connected in a ring with some re-wiring of the protection span inputs at each pair of back-to-back terminals.) Rings offer the advantages of being closed transport subsystems, with unquestionably fast protection switching, and "pay as you grow" cost characteristics. At least initially people thought mesh-restoration was too complicated. One ring looks very simple, and this captivated the industry. But with time it was found that multi-ring networks are actually more complex (and in many ways more brittle) than a single integrated mesh network. With the growing dominance of data over voice, and a merging of viewpoints from people with data-centric backgrounds with those having more traditional telco ("Bell-heads") backgrounds, the pendulum has swung back toward an interest in mesh restoration. The primary reasons are that mesh offers greater flexibility, efficiency, and inherent support for multiple service classes. The greater capacity efficiency comes from the more direct routing of working paths, the need for less spare capacity for restoration, and the avoidance of "stranded capacity" effects in rings (where one or more ring spans may exhaust while other spans of the ring have valuable but unusable remaining working capacity). Mesh-based networks also offer the prospect of fully self-organizing operation in response to time-varying patterns of demand. "Point and click" or fully automated path provisioning is more difficult through a collection of transport rings than through a single integrated mesh network.

Achieving capacity efficiency from the sharing of spare capacity is a central aspect of the design methods in this book. Over 100% redundancy is required with either APS or ring systems. In contrast, mesh restorable networks are based on generalized rerouting as needed for restoration using any or all diverse routes of the network. [Figure I-1](#) conveys the idea of generality and flexibility of the rerouting patterns, and hence the sharing of spare capacity, that is implicit in mesh restoration. Spare capacity on one span typically contributes to the restorability of many other spans. Such networks are called "mesh-restorable" not to imply that the network is a full mesh, but to reflect the ability of the routing mechanism to exploit an irregular mesh-like topology. Anywhere network cost correlates to total installed capacity, or where maximum demand-serving ability is required out of a given transmission capacity base, the low redundancy of a mesh-restorable network is a factor in its favor.

Figure I-1. Mesh restoration involves network-wide sharing of spare capacity.



Capacity efficiency is not the only consideration in a choice of network architecture, but in a competitive environment the combination of

efficiency, ease of growth, and service-provisioning flexibility that a mesh network can offer can provide carriers with a productivity edge over their competitors. The notion that "bandwidth is free" is simply not accurate at the scale of investment faced by transport network planners where incremental capital expenditures of \$US 300 to 400 million for transport equipment are typical. If a 40% increase in capacity efficiency leads to even just 15 to 20% cost savings on a budget like that, then "going mesh" would be a well-qualified project for corporate productivity and profit enhancement.

Efficiency in capacity is also inherently linked to flexibility to cope with forecasting uncertainty, and provisioning productivity in terms of handling high growth rates and/or high rates of customer "churn." Mesh-based networks are more "future-proof" because for the same investment in capacity, one can serve more working demand, and in more diverse patterns than a corresponding set of rings. Sustained rapid growth or churn also creates an environment where there is a premium for capacity efficiency purely because the speed of deploying new capacity or making changes in routing and protection arrangements is the rate-limiting step to earnings growth. Any time new transmission capacity can barely be provisioned fast enough to keep up with demand, then a more efficient architecture will also serve more demand (and hence earn more revenue) given a currently installed base of transmission systems.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

Outline

Chapters are grouped as being either preparatory or specialized in nature, giving the book two main parts. The aim is to provide a new planner, or a graduate student, with a combination of in-depth technical exposures to advanced concepts ([Part II](#)), while also being informed by an overall awareness of the technology and general issues and setting of the field ([Part I](#)). The [Part I](#) chapters not only facilitate access, absorption, and use of material in the later chapters but also provide a backgrounder on the technology directions and network planning issues that influence the ongoing direction and use that a reader can make of the specialized [Part II](#) topics.

[Part I](#) "Preparations"

[Chapter 1](#): Orientation to Transport Networks

Here we introduce the key concepts of *transport* networking which is different in important ways from telephone circuit switching and packet data routing networks that many readers will already know. The concept of "transport networking" has evolved as an extension of transmission system engineering more than anything else. Key ideas here are of service layer networks being clients of the transport layer, and awareness of multiplexing, switching, grooming, aggregation, and carrier transmission system technology basics. This includes an overview of SONET and still widely employed "plesiochronous" digital signals as well as looking at the basic concept of label switching that underlies ATM and MPLS. In closing [Chapter 1](#) we look introduce some aspects of network planning that are specific to transport networks such as modularity of capacity, physical route structures, demand modeling, and shared-risk entity concepts.

[Chapter 2](#): IP and Optical Networking

The dominance of Internet Protocol (IP) packets as the main source of traffic, and the emergence of Dense Wavelength Division Multiplexing (DWDM) technology to carry huge increases in total demand, are two of the greatest technical "discontinuities" affecting network operators, planners and equipment vendors. The material of this chapter is devoted to these important developments—essential for a new student or engineer to join in, in informed participation, on topics related to modern transport network planning. This includes a review of data-centric payloads, gigabit Ethernet, extensions to SONET for enhanced data transport, optical service channel and "digital wrapper" concepts that provide signaling overheads for provisioning and restoration or protection applications. The chapter also familiarizes readers with the concept of adapting and using classical IP protocols for topology discovery and link-state dissemination for control of the optical transport network. This completes the technology backdrop in which the more general design theories and methods of the [Part II](#) chapters are set.

[Chapter 3](#): Failure Impacts, Survivability Principles, and Measures of Survivability

This chapter starts with background on the causes and frequency of transport network failures and their impact on various service types. This includes a discussion on the role and real need for the so-called "50 ms" restoration times. Basic techniques for avoiding or

responding to failures are then covered, recognizing that survivability measures can and should be taken at all levels from the physical layout of cables up to the service layer where performance-related effects are ultimately observed. This is followed by a high-level survey of all known principles for addressing the network survivability problem. This overview of basic survivability principles gives a first appreciation of the schemes that will be covered later in more detail, and sets them in place against other techniques, such as rings and APS primarily, that are not part of the book but to which comparative reference is often made. This "round-up" of all known schemes is also intended to serve as a "quick-reference" resource in its own right. The survey of schemes includes a discussion of the popular tendency to classify schemes as either protection or restoration schemes, and why this is oversimplified. The chapter then covers important technical groundwork for treatment of survivable networks such as the redundancy of a network, measures of outage severity and the concepts of reliability, availability, restorability, unavailability and "network reliability."

Chapter 4: Graphs, Routing, and Optimization

The final preparatory chapter is devoted to mathematical and algorithmic basics needed for someone to reasonably absorb, and then apply and extend, the ideas and methods of [Part II](#). This is an ambitious and fairly technical chapter that serves not only as background for [Part II](#) but also as a self-contained reference for ongoing work to apply, extend, adapt or experiment further with the networking ideas and design models of the later chapters. Everything included in this chapter is material that the author has found that graduate students specifically need to know, or have in their "toolkits," to support further work in the area. The treatment of shortest-path algorithms includes very accessible explanations of Bhandari's modified Dijkstra algorithms and a simplified explanation of "Surballes algorithm." Detailed intuitive explanations for all distinct routes, k -shortest paths, max-flow, shortest disjoint path pair, cut tree, and biconnected component algorithms are all given. The value to many readers will be that in the treatments given here, time is taken to explain fairly fully why and how these algorithms work, not just a specification of each algorithm. These are tested explanations that grad students have said really worked for them.

[Chapter 4](#) then contains a similar grounding on optimization methods. This includes linear and integer linear programming methods, duality, and Lagrangean relaxation techniques. The role of formal Operations Research (OR) methods in practical network planning and research is also debated and defended. We argue that, notwithstanding that exact solutions are "NP hard," such methods provide for a whole range of easily created, highly effective heuristics that are almost always overlooked. A variety of basic network flow problem types are looked at and the concept of unimodularity or "special structure" is explained. The chapter also offers practical tips on formulating and solving LP/ILP models and explains Genetic Algorithms, Simulated Annealing and Tabu Search as additional basic techniques for network design and planning. Whereas an experienced planner may skip [Chapters 1 to 3](#) without creating difficulty in the later chapters, [Chapter 4](#) is more likely to be either useful or essential even to the well-prepared reader. [Chapter 4](#) completes the "preparatory content" part of the book.

Part II "Studies"

As a set, [Chapters 1-4](#) inform and orient the newcomer, or update a current practitioner, and establish a baseline of common concepts, language, and technical preparation for subsequent chapters. Each [Part II](#) chapter is then devoted to in-depth treatment of a specific class of survivable network such as span- or path-restorable networks or new concepts and methods such as p -cycles, hybrids, topology, oversubscription design, dual failure analysis, and so on.

Chapter 5: Span Restoration and Protection

This chapter is devoted to all aspects of *span-restoration* and the closely related technique of *preplannedspan-protection* using shared spare capacity. The basic capacity design models are given, then enhanced with many real-world details such as modularity, nonlinear cost structures, express-route optimization for chains, multi-priority design, and so on. Span restoration was the basis of the first historical proposals for real-time mesh-based restoration and is perhaps the simplest, and so far most theoretically and technically well-developed option for survivability. Span restoration protects working capacity directly, so that any path provisioned over the network is also automatically protected (if desired) without any further considerations. The chapter contains original material and ideas about how

this property of span-restorable networks lends itself to distributed preplanning and to the "working capacity envelope" concept for simplified provisioning of dynamic demands. We also explain how an ultra-high-availability service class can be supported with a "first-failure protection, second-failure restoration" strategy. The chapter also includes the "forcer concept" which is a way of analyzing and understanding the capacity structure of mesh-restorable networks. We use it to give new theoretical understandings about the effect of nodal bypasses on capacity optimization. In [chapter 11](#) the forcer concept is further applied as the basis of ring-mesh hybrid network design. Other original material in this chapter includes explanation and design methods for the "meta-mesh" concept that enhances mesh capacity efficiency on very sparse facility graphs, incremental-growth planning models, and a trio of bicriterion design methods for enhancing maintenance robustness, reducing paths lengths, or maximizing restorability levels of best-efforts service classes in such networks.

Chapter 6: Path Restoration and Shared Backup Path Protection

[Chapter 6](#) provides a corresponding in-depth treatment for *path restoration*. This includes Shared Backup Path Protection (SBPP) which is widely favored for optical networking and MPLS applications. Path restoration is somewhat more capacity efficient than span restoration and—by referring the fault detection and restoration control actions to the end-points of each transport path—SBPP can also simplify real-time operations in an optical network. Among the original material in this chapter are previously unpublished discussion of Lagrangean relaxation methods to solve path oriented capacity design problems and extensive development of new heuristic design methods, new data on SBPP capacity requirements, and new results and discussions of stub-release issues in path restoration. This includes extension of the forcer concept to path restoration, how to bring modularity into the design, and definition of new minimum-cost incremental growth planning models. The chapter also addresses a number of apparent confusions about path-oriented techniques, for instance, why path restoration is not just span restoration with "loopback detection," and why the popular idea of "mass reprovisioning" with GMPLS is not an assured restoration technique. The concept of mutual capacity, which underlies all path-oriented restoration schemes, but is not an issue in span restoration, is also explained with examples.

Chapter 7: Oversubscription-based Design for MPLS or ATM Protection

Oversubscription-based capacity design is a restoration-related planning strategy applicable to MPLS or ATM VP layer networks. [Chapter 7](#) starts by explaining the functional equivalence of MPLS and ATM VP based networks from a logical design standpoint that would be useful to someone already familiar with ATM but not yet with MPLS. It then explains the capacity-saving concept of designing for deliberate, but *controlled* over-subscription of total capacity upon restoration. By treating the problem of design for controlled oversubscription of capacity in a restored network state this chapter also illustrates the potential for complicated and severe congestion hazards that some more ad hoc proposals for "mass-redial" restoration may engender. It is also explained how the oversubscription-oriented design strategy can be extended to implementation of multiple service classes.

Chapter 8: Dual Failures, Nodal Bypass and Common-Ducts Effects

This chapter treats a whole variety of "dual failure"-related issues, including original work on dual-failure design for ultra high availability, the effects of shared-risk link groups (SRLGs) in span-restorable mesh networks, and how to minimize the effect of span maintenance actions in terms of the theoretical risk they may pose of restorability loss should a real failure occur during this maintenance state. Under the umbrella of "dual failure" situations, we consider not only dual-failure design for 100% restorability, but also (and more practically) enhanced dual-failure restorability design for premium service classes. We also consider the effects of express route "bypass" setups, which cause a type of dual logical failure situation. But we clarify the important way this situation differs from other types of dual failures. A novel advance is showing how a "platinum" service class can be created that receives assured dual-failure restorability—and hence *better than 1+1 availability*—with little or no more spare capacity than needed in an ordinary span-restorable network. A generally important finding is that to design against 100% of all dual-failures will require about three times the spare capacity of a single-failure design. But at the same time mesh networks designed for only single-failure restorability produce high average dual-failure restorability levels which can be the basis for ultra high availability service offerings. A design model to accommodate any specific set of known SRLGs is given and it is used to show the capacity penalties associated with survivable design in the presence of SRLGs and how to pinpoint the most "troublesome" SRLGs in that regard.

Chapter 9: Mesh Network Topology Design and Evolution

This chapter is devoted to the challenging and so far almost unaddressed problem of design or evolution of the facility-route topology of a mesh-restorable network. After surveying classic work on topology design for private networks and the theoretical problem of "fixed charge and routing," we see that mesh-survivable networks present a fundamentally new class of problem in topological network design. On the one hand, the routing of working flows wants a sparse, tree-like graph for minimization of the edge costs. On the other hand, restorability requires a closed and preferably high-degree topology on which the sharing of spare capacity allocations over non-simultaneous failure scenarios is efficient. These diametrically opposed considerations underlie the determination of an optimum physical facilities graph for a mesh network. This chapter first looks at experimental data on how nodal degree (overall connectivity of the topology) creates a minimum cost "sweet spot" for mesh-survivable topologies. The chapter then defines a formulation for the complete mesh-restorable design problem and a three-stage heuristic, and other heuristic search strategies, for its approximate solution. The sub-problem of planning a single-span addition to an existing facilities network is given separate treatment. An "edge limiting" criteria for iterative inclusion of candidate graph edges is described which is highly effective in solving the topology optimization problem. Graph sifting and the strategy of "sweep search" through topology space are also described. These are novel heuristics that home-in on the topological "sweet spot" seen experimentally at the start of the chapter. Finally we show how methods to solve the optimal 'ground up' topology design problem can be directly adapted to the more-often faced problem in practice of the evolution of an *existing* transport network facility graph.

Chapter 10: p -Cycles

p -Cycles provide a new option for either optical-layer or MPLS-layer protection. What is interesting and remarkable is that p -cycles offer ring-like speed with mesh-like efficiency. As such, the p -cycle networking concept more or less breaks the long-standing quandary between ring and mesh alternatives that was the status-quo for nearly a decade. Mesh-like efficiency is obtainable without giving up ring-like switching simplicity or speed. This chapter brings together all known advances on p -cycles to date and adds previously unpublished material on extensions of the basic concepts and practical heuristics for solving p -cycle network design problems. We start by explaining p -cycles and how they differ from other "cycle oriented" approaches such as enhanced rings or cycle-double covers. We then give basic design models for minimum spare capacity design of p -cycle networks and show sample results validating the "mesh-like efficiency" claim. p -Cycle design for DWDM networks—including wavelength conversion issues—is treated next. Joint optimization of working routes and spare capacity for p -cycles is also treated followed by discussion of the concepts of p -cycle "score," and several simple algorithms for developing p -cycle based network designs. We also provide a treatment of issues related to "Hamiltonian p -cycles" for homogeneous-capacity networks and show how p -cycles actually inspire a specific new class of "semi-homogeneous" networks that are of theoretically maximum efficiency. We describe how p -cycles can be either cross-connect based, and managed, or based on new nodal device structures that are ADM-like in terms of their capacity and growth characteristics. Also shown is how existing ADMs can be converted for p -cycle operation with addition of a novel "straddling span interface unit." A self-organizing strategy for adaptive p -cycle formation is described and p -cycles in the MPLS or ATM layers are also considered, including the use of *node-encircling* p -cycles to protect against router failure as well as link failure. With roughly 90 pages devoted to the topic, this chapter would almost stand in its own right as the first book to be published on p -cycles.

Chapter 11: Ring-Mesh Hybrids and Ring-to-Mesh Evolution

The book is devoted to advancing the methods and ideas available for mesh-based survivable networking. Ironically, however, many network operators and vendors that are interested in mesh-based planning for their future growth, have extensive existing investments (both intellectual and physical) in ring-based networking. If the appetite for mesh has been whet by this book, then it begs the question of how might we plan a graceful transition from ring to mesh—ideally while reusing and benefitting from the ring legacy as much as possible. That is what this chapter helps provide. The chapter is devoted to the design of ring-mesh hybrid transport networks and to a novel strategy for ring-to-mesh (or ring-to- p cycle) evolution, both of which provide options for evolving from a current situation of an all-ring network. With the concept of "forcer-clipping" we show how selected rings can maximize the capacity efficiency of a residual mesh component in a ring-mesh hybrid. This defines the architecture of a true hybrid transport networks, but it can also suggest which

existing rings to keep while otherwise evolving toward mesh. The options we develop are called "ring-mining" strategies for ring-to-mesh evolution. The main benefit is that significant amounts of ongoing growth in demand can be served before adding new capacity and equipment, by reclaiming the protection capacity and inefficiently used working capacity in existing rings. The recapture of existing installed protection capacity for conversion to service-bearing use would be a one time business strategy opportunity made possible by conversion from ring to mesh. We detail the key ideas and methods of evolving an existing set of rings into either a span-restorable mesh target architecture, or to a target architecture where rings are adapted with straddling-span interface units for re-use as p -cycles.

Notes

The failures described in the opening paragraph are examples of FCC Outage reports at the FCC Office of Engineering and Technology (Internet (2002): <http://www.fcc.gov/oet/outage/>). The estimate of \$500 million in revenue losses was reported as a quote attributed to the Gartner Group by OPNET at their web site describing various traffic flow visualization and failure impact analysis capabilities of the OPNET Flow Analysis Module (Internet (2002): http://www.opnet.com/products/modules/flow_analysis.html)[OPN02]. The data on frequency of cable cuts is also from the FCC as reported by [VePo02].

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Part 1: Preparations

[Chapter 1. Orientation to Transport Networks and Technology](#)

[Chapter 2. Internet Protocol and Optical Networking](#)

[Chapter 3. Failure Impacts, Survivability Principles, and Measures of Survivability](#)

[Chapter 4. Graph Theory, Routing, and Optimization](#)

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Chapter 1. Orientation to Transport Networks and Technology

Many readers will have an initial picture of how the telecommunication network operates that is based on telephony circuit switching. This involves the switching of individual voice circuits through a network of trunk groups. Historically the trunk groups were cables comprising many twisted wire pairs and switching involved metallic connections between wire pairs in these trunk cables. Later, in the first step toward virtualization, point-to-point carrier transmission systems created more than one logical channel per twisted pair, and switching evolved to making connections between these logical channels. But in this context the transmission systems only provided "pair gain" between the switching nodes. In a second important step, cross-connect panels or cross-connect machines began to provide interconnection directly between carrier systems so that the apparent network seen by the circuit-switching machines could be rearranged at will. The same evolution has been repeated with wavelength division multiplexed (WDM) transmission, from point-to-point "pair gain" with coarse WDM, then to optical transport networking using dense WDM and optical cross-connects. In both cases the flexible interconnection of multiplexed carrier signals allows creation of virtual logical connectivity and capacity patterns for client service networks. This is the fundamental idea of *transport networking* regardless of the technology involved.

Through transport networking, an essentially fixed set of multi-channel point-to-point transmission systems are managed to create virtual network environments for all other services. Today, for example, the logical model of voice circuit switching still holds but the entire telephone trunk network is virtual. There are no actual twisted pair cables between voice switches. Nor are there cables or fibers dedicated between IP routers or cables that "belong" to a banking network or private company network. All these networks operate logically as if they did have their own dedicated transmission systems, but they are each just one of several virtual "service layer" networks supported by one underlying actual network; the transport network.

Another widespread concept is packet switching. A compelling but oversimplified notion is that each packet takes an independent route through the network directly over the physical-layer cables. In reality, aggregations of data packets with common destinations or intermediate hubs en route are formed near the edges of the network and then follow preestablished logical conduits without further packet-by-packet inspection until they are near or at their destination. These logical conduits appear to the packet-switching nodes to be dedicated point-to-point transmission systems, but in fact they are cross-connected paths through the transport network, providing the logical network for the packet-switching service.

Both of these familiar types of network (e.g., circuit-switched telephony and packet data switching) have in recent times become essentially virtual. They are only examples of specialized service layer networks operating over a common transport network. It is natural for us also to perceive and interact with other services such as the Internet, the banking network (ATM machines), credit verification, travel booking networks, and so on, as if they were separate physical networks. But they are all just logical abstractions created within one physical network by logical configuration of the carrier signals borne on fiber optic strands. Individual telephone calls, packet streams, leased lines, ATM trunks, Internet connections, etc., do not make their own way natively over the fiber systems. Rather, aggregations of all traffic types from site to site are formed by multiplexing these payloads up to a set of standard-rate digital carrier signals of rates such as DS3, STS1, STS3c and so on, to be discussed. Traffic of all sources, sometimes even from competing service providers, is combined (by statistical multiplexing or time division multiplexing) into composite digital signal streams for assignment to a given wavelength path or fiber route through the physical network.

The transport network operates below these user-perceived service networks, and above the fixed physical network, to "serve up" logical connectivity requirements to such client networks from the underlying physical base of transmission resources. This client-server relationship can sometimes be observed across several layers. For instance, an optical transport network may implement lightpaths to create the logical topology for a SONET OC-192 network. That SONET network may itself provide routing and cross-connection for a mixture of logical paths and a variety of connection rates and types such as Gigabit Ethernet, ATM, IP, and DS3, or DS1 leased lines. An ATM client network of the SONET layer may itself then provide transport for various IP flows between Internet routers, and so on. Thus *transport* can also be thought of as an inter layer relationship, not always a single identifiable layer in its own right. An important technological drive is, however, to reduce this historical accumulation of layers to a single layer-pair: that of "IP over optics." This is an important development to which we will return. "IP over optics" both simplifies and enhances the application of the network design methods that follow.

The mandate of this chapter is to establish familiarity with the basic concepts of transport networking, the technologies used, and some

of the important issues that distinguish transport networking from other far more dominant networking concepts. This establishes context for the design problems that follow and equips a student to understand and explain to others—such as at a thesis defense—how it is, for example, that we don't see individual packet routing considerations in a transport layer study, or how it can be that some nodes of the actual network do not appear in the transport network graph. ("Does it mean you have decided not to provide service to those cities?") These questions, both of which were earnestly posed by committee members to the authors students during their dissertations, are typical of the misunderstandings about, or simple unawareness about, the concept of transport networking. Thus, there is a real need for students and practitioners to understand transport as a networking paradigm of its own, different from telephony switching and packet switching as well as leased-line private network design, and to be able to give clarifying answers to questions such as above. However, space permits only a basic introduction to transport networking; the minimum needed to support later developments in the book. To delve deeper, the book by K.Sato [\[Sato96\]](#) is recommended as a supplement. It gives a more extensive treatment of key ideas and technologies for transport networking without considering network design or survivability.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

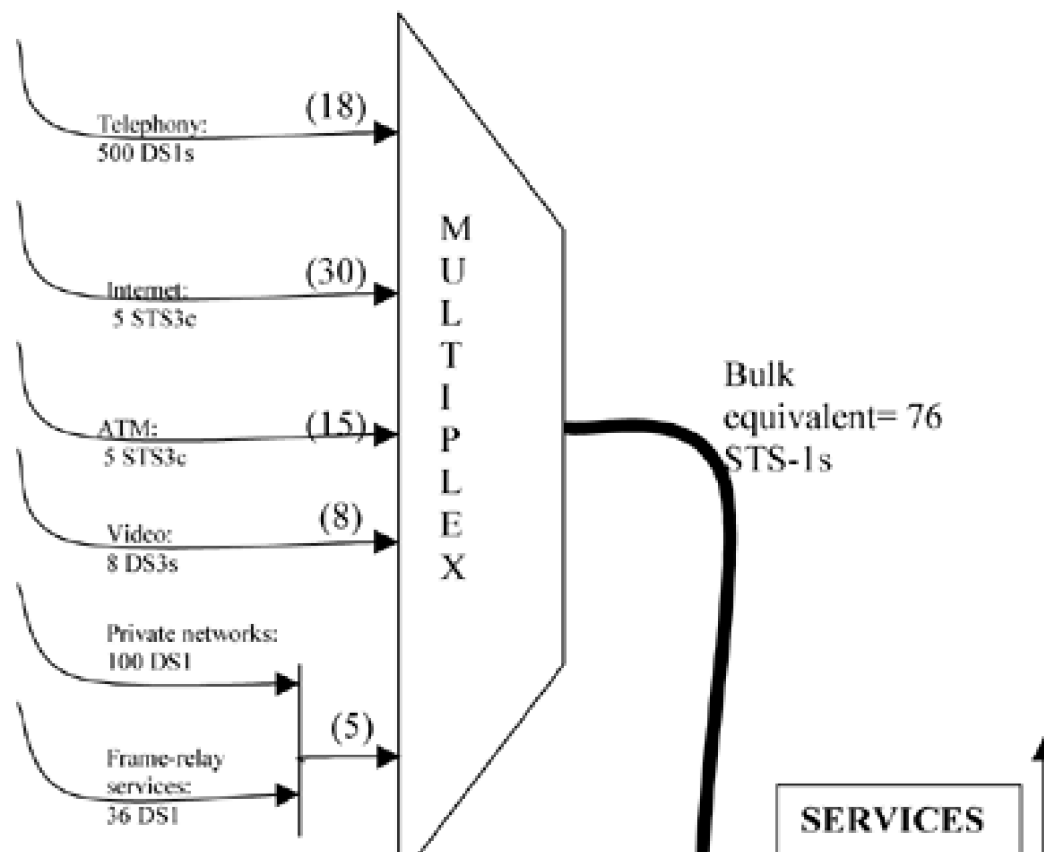
1.0.1 Aggregation of Service Layer Traffic into Transport Demands

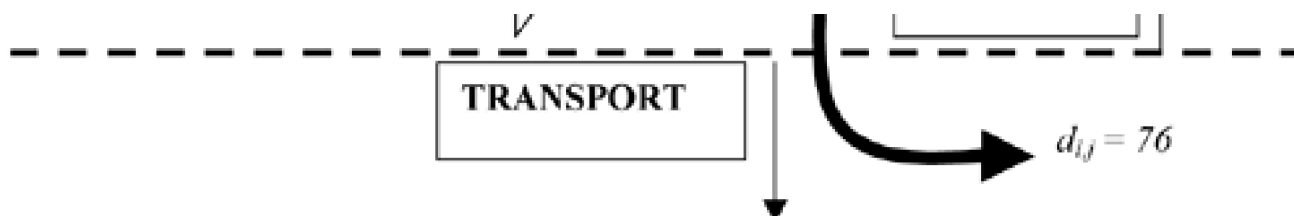
Ultimately, whether through IP over optics or through a stack of DS-3, ATM and SONET layers, the net effect is that a set of user services is mapped onto a set of physical high-capacity transmission, multiplexing and signal switching facilities that provide transmission paths to support the logical connectivity and capacity requirements of all service layer flows. The aggregation of flows between each pair of nodes on the transport network defines what is called the *demand* on the transport network (or the respective transport layer). The term *demand* has a specialized meaning, distinct from the more general term *traffic* [Wu92]. A *demand unit* is a quantum of transmission and routing capacity used to serve any aggregations of traffic flow from the service layers. Whereas traffic may refer to measures of voice, data, or video flow intensities (Erlangs, packets per second, Mb/s, frames/sec, etc.), demands on a transport network are specified in terms of the number of managed units of transmission capacity that the aggregation of traffic requires. Demand may thus take on standard transmission units such as lightpaths, OC-192s, OC-48s, DS3s or DS1s. An optical backbone network may typically be managed at the OC-48 (~2.5 Gb/s of aggregated data) and the whole lightpath (i.e., contiguous optical carrier signal frequency assignments) level. Each lightpath could be formatted to carry an OC-192 that may be structured as 4 OC-48s or to carry a 10GigE aggregation of Ethernet packet frames, or many other application-specific payload formats for which mappings are defined. These are just examples of the general concept of aggregations of payload being matched onto standard units of demand for routing and capacity management in the transport layer.

Figure 1-1 illustrates the basic concept of multiple traffic sources being aggregated based on their common destination (or a route to a common intermediate destination) thereby generating a demand requirement on the transport network. In the example, the bulk equivalent of 76 STS-1s would in practice be likely to generate two OC-48 demands.

Figure 1-1. Traffic sources are aggregated from service layers into transport demand units

SITE *i* traffic sources to: SITE *j*





It is important to note that, as the word "transport" suggests in general language, the individual packets, cells, phone calls, leased lines, and so on are no longer recognized or individually processed at the nodes en route. In effect they have been grouped together in a standard "container" for shipment toward their destination. Only the containers themselves are recognized and manipulated in the transport network. The concept of containerization and transport, although commonly appreciated in trucking, shipping, air transport, mail and overnight courier services, seems sometimes hard for those more familiar with packet switching or call switching to immediately accept. The often strong presumption is of networks where every call or packet is inspected and routed at every node.

An example of a set of transport services that a carrier might offer, and the "bandwidth"^[1] that is allocated for each in transport capacity planning is given in [Table 1-1](#) where the basic unit of capacity planning is an STS-1. The table includes the traditional signal types such as DS-1 through OC-48 as might be provisioned for private line services or for the operator's own inter-switch trunking requirements. Also included are IP-based service offerings where the transport bandwidth allocated reflects the expected average utilization of the access signal and the benefit of statistical multiplexing. For instance, an ISP with OC-12 interface links on its routers may request a "private line" OC-12 as the bit-pipe to another router. Such a "PL" OC-12 will be a true OC-12 circuit for which 12 STS-1s are allocated. Alternately, it may be an "IP" OC-12 service in which case a bandwidth of only about 13% of an actual OC-12 is allocated. The latter, obviously lower-cost, service essentially just provides an OC-12 access interface, from which the IP payloads will be extracted and statistically multiplexed with other flows in the carrier's network. [Table 1-1](#) and [Figure 1-1](#) are just examples. The general point is that in operation and design of a transport network we deal with demand requirements posed in some basic unit of transmission capacity. These requirements are generated by the aggregation of all types of service traffic. But in the transport network we generally never see or have to consider individual packets or phone calls, etc., directly again.

[1] A widespread practice is to use "bandwidth" and "capacity" as synonyms. Strictly, however, *bandwidth* is a physical attribute of a transmission channel. It is the band of frequencies that the channel will pass below some attenuation threshold, with units of Hz. The *capacity* of a channel is the rate of information that can be passed through the channel under specific conditions of modulation, coding, power and noise, and has units of bits/s.

Table 1-1. Transport capacity allocated for various service types (STS-1 equivalents)^[a]

Service	Capacity	Service	Capacity
DS-1 PL	0.036	IP-OC3	0.382
DS-3 PL	1.0	IP-OC12	1.528
OC-3 PL	3.0	IP-OC48	6.112
OC-12 PL	12.0	IP-100T (Ethernet)	0.283
OC-48 PL	48.0	IP-GIGE (Ethernet)	2.830
IP-DS1	0.005	WL-2.5G	48.000
IP-DS3	0.127	WL-10G	96.000

[a] PL = private line service in stipulated format, IP = IP packet service with specified interface rate and format, WL = wavelength service bearing OC-48 or OC-192 container format.

In [\[DoHa98\]](#) the processes of bandwidth allocation based on service types, and overall of aggregation of point-to-point bandwidth requirements to define a demand matrix for transport network design, are described further. That paper also describes an overall framework of a network planning tool—the Integrated Network Design Tool (INDT)—that is relevant as an example of the kind of network planning software tool where many of the design methods in the [Part 2](#) chapters of this book would be employed in future.

[\[Team LiB \]](#)

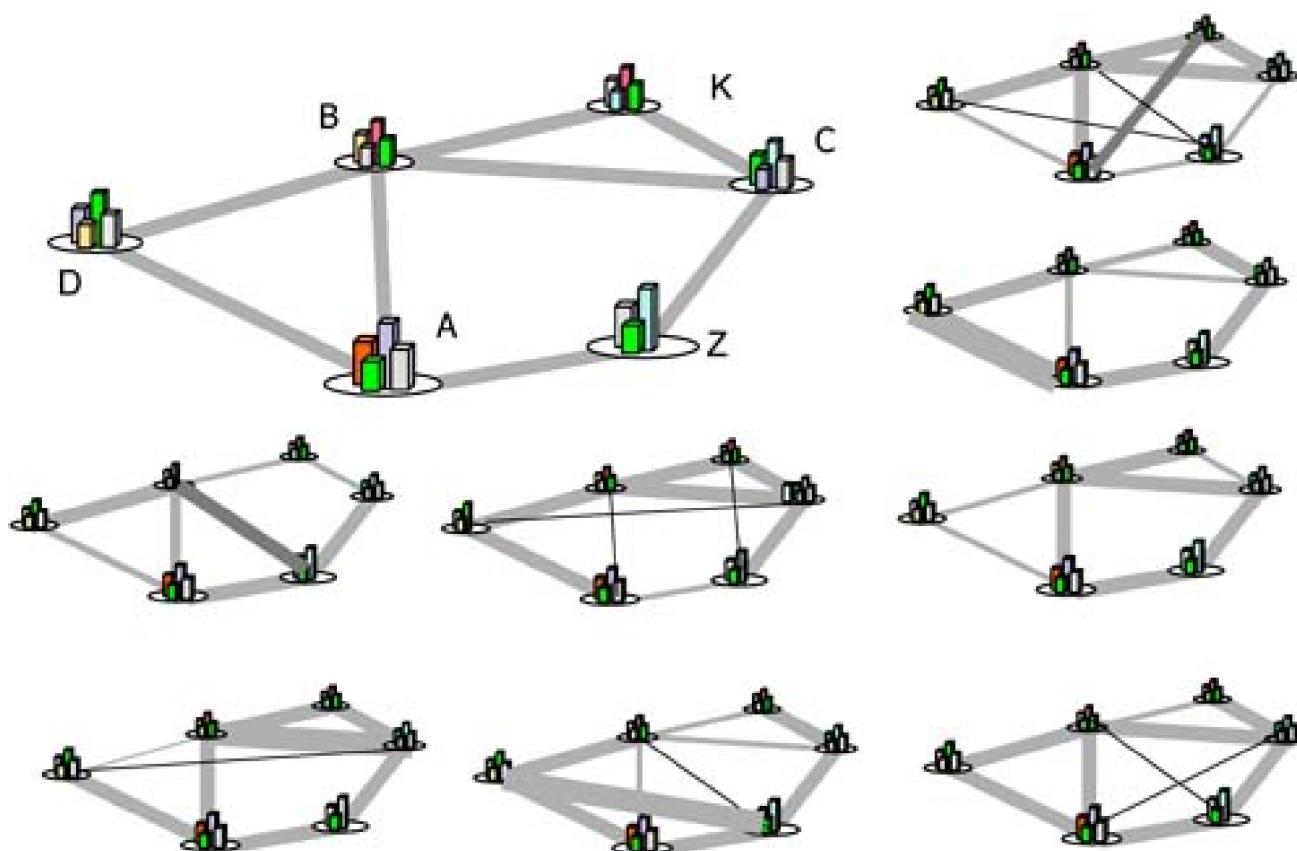
◀ PREVIOUS

NEXT ▶

1.0.2 Concept of Logical versus Physical Networks: Virtual Topology

Each fiber optic transmission system is itself an essentially fixed point-to-point structure that bears whatever set of tributary carrier signals or wavelengths are presented to its inputs, up to its maximum capacity. Changes in the physical layer can be made, but on a much longer time scale than required for purely logical reconfigurations in the transport network. The set of tributary signals (e.g., say STS3s) borne on each fiber system (e.g., an OC-48) can be *cross-connected* to create a vast number of logical transport configurations for the higher service layers. The pattern of logical point-to-point interconnection created by cross-connection of channels in the physical graph is also called the *virtual topology* [RaSi98] [StBa99] [SiSu00]. The virtual topology has the same set of nodes but has an edge between each node pair that have a path established between them. Figure 1-2 illustrates how the set of unit-capacity tributary signals at the input and output of each essentially fixed point-to-point transmission system can be interconnected to provide a vast number of different *logical* transport configurations. The different line thicknesses portray different amounts of point-to-point capacity between nodes. It is these virtual topology configurations that different service layers perceive to be the physical transmission network.

Figure 1-2. A set of fixed point-to-point physical transmission systems and a small number of the virtual networks that higher layer networks may be made to think is present.



The set of ways in which tributary channels on each fiber can be cross-connected to form different patterns of logical connectivity and capacity is a combinatoric space. If the physical network consists of a set of spans S (indexed by j) with individual capacities B_j , then all logical configurations that satisfy the following constraints are feasible:

Equation 1.1

$$\sum C_{v,v'} \cdot \delta_{v,v'}^j \leq B_j \quad \forall j \in S$$

$$\forall(x,y) |x \neq y$$

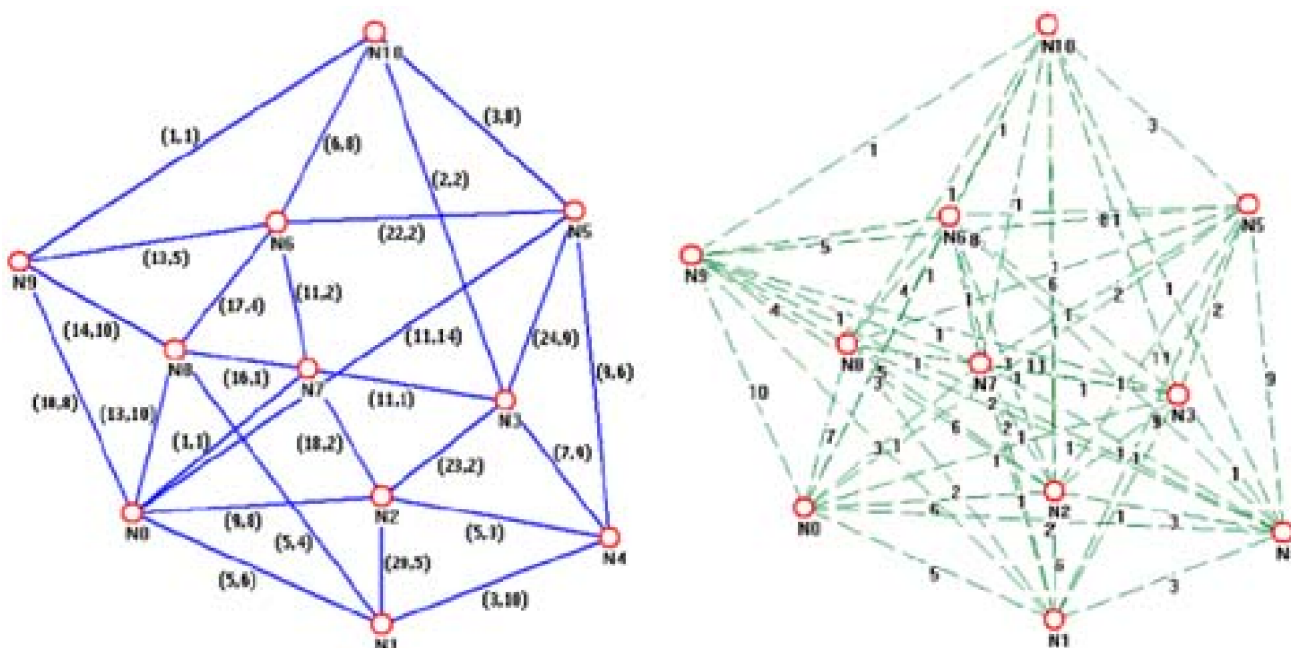
where $C_{x,y}$ is the point-to-point capacity provided between nodes x and y in the logical configuration and $\delta_{x,y}^j$ indicates the route over the network taken to provide the x to y logical pipe: $\delta_{x,y}^j$ is 1 if the route crosses span j , and zero otherwise. (\forall means "for every...")

A frequent analogy for the reconfigurability of the transport network is the routing of trucks over a system of fixed highways with customer payloads inside the containers carried by the trucks. This fits everyday experience to an extent but it belies the circuit-like nature of the actual transport network. A system of point-to-point pipelines, each pipe having a finite flow limit, within which smaller rearrangeable tubes are interconnected would be a more exact analogy. From the discussion, however, there are two important properties of the transport network that are most relevant to the design problems that follow:

- Many logical configurations of the transport network will be functionally indistinguishable by the service layers that use the transport network. (This attribute we use for restoration.)
- The same fixed physical transmission systems can be logically reconfigured to serve many different demand patterns. (This attribute we use for traffic adaptation.)

These and further basic concepts about transport networking are illustrated in [Figure 1-3](#) which is based on actual data for a European inter-city transport network. [Figure 1-3\(a\)](#) shows the physical sets of transmission spans present. It is spans of this network that fail when we postulate a cable cut. Each span in the physical graph (a) has an associated total capacity that is annotated on the figure and represents the number of basic transmission channels or transmission layer links between each physically connected node pair. As shown here, capacity is presented as (working, spare) couplet on each span. The working capacity supports the shortest-path routing of the demands in (b). These are, for instance, individual lightwave channels on DWDM fibers between the nodes. [Figure 1-3\(b\)](#) shows the pattern of logical interconnection ("demand") required in this network. In other words, [Figure 1-3\(b\)](#) is a graphical portrayal of what is known as the demand matrix. The demand quantities are annotated on each dashed line in (b) and would represent, for example, the number of lightpaths required end-to-end between each node pair.

Figure 1-3. A physical transport network and overlying pattern of service layer demand.



Each unit of demand required between nodes in the logical layer has to be mapped onto a route over the physical graph and assigned a transmission path that is a specific sequence of connected channels over the chosen route. The demand between N1 and N10 may be served for example by paths over route (N1-N2-N3-N10) of the physical graph. But from the logical layer view, paths over route (N1-N2-N7-N6-N10) would be equally as good (logically indistinguishable in fact) and thus if span (N2-N3) fails, these paths could be

shifted from the first to the second route. However, the diagram makes it clear that if a span in (a) gets cut in isolation, there will not be an isolated effect in (b). Many demand pairs will be affected by a single cut in (a). Thus, any such shifting over of the paths for node pair (N1-N10) will have to be cognizant of the actual capacity on those other spans and coordinated with the corresponding switchover actions for other affected node pairs that might also use capacity on those spans. In this regard, the spare capacity values shown in (a) are reservations of additional capacity that is sufficient to support end-to-end path restoration (one of the schemes to follow) of all demands whose working paths are disrupted by any single-span failure in (a). Conversely, [Figure 1-3](#) lets us appreciate how the available capacity on the edges of graph (a) can be cross-connected in different ways to support different patterns of demand in (b). This is also the conceptual point about transport networks that [Figure 1-2](#) is conveying.

The two network views, logical and physical, in [Figure 1-3](#) also convey two basic classes of problem, depending on whether capacities or demands are assumed as given quantities. One class of problem assumes that a forecast or planning view of the demand pattern is given. This may actually be a family of possible future demand patterns, or, a stipulated "envelope" of maximum anticipated demands that the network may have to serve. The problem then is to solve for the minimum cost allocation of transmission capacity in (a) (or addition of new capacity to an existing set of capacities) that supports both the routing and protection of all demands in (b). Once a network is built, however, the nearer-term operational problem is one of routing and configuring protection arrangement so that demands that actually arrive are both served and protected within the current as-built capacity. Over the life of a network these two phases are revisited in a constant cycle.

Unlike circuit switching for voice, the lifetime of connections in the transport network is generally much longer, typically days to years because the *aggregations* of traffic do not change as rapidly as individual service connections do. For this reason, transport network connections are often referred to as "semi-permanent" or "nailed-up" connections. The routing environment is also different from routing through networks of trunk groups. First, in making rearrangements within the transport network (especially for restoration) there essentially cannot be any blocking, because "blocking" in the transport domain means hard outage for all the services that the blocked carrier signal would have borne. Secondly, once established, paths in the transport network may be very long-lived so there is much more impetus to try to globally optimize the assignment of transport capacity. In contrast, the routing of any one call only commits the system for a few minutes, after which time the system gets to start over with subsequent calls. In other words, the system state rapidly decorrelates in the voice circuit-switched network (or the state of flows in an IP router-based network) but has a much longer correlation time in the transport network.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

1.0.3 Multiplexing and Switching

Multiplexing is the simultaneous transmission of many separate messages or lower-speed logical circuit connections over a shared medium.^[2] Multiplexing can be via space, time, frequency or code division. Space refers to parallel physically distinct channels, such as the separate fibers of a fiber optic cable or physically separate output ports on a switch or router. Time refers to synchronous time slot allocation, as in SONET, or asynchronous time slot allocation as in ATM. Frequency division multiplexing (FDM) is the basis for conventional AM/FM radio and the pre-fiber generation of high capacity analog and digital microwave transmission systems. When applied to optical frequencies, FDM is the basis of both coarse and dense wavelength division multiplexing (WDM).

^[2] This section, through to [Section 1.2.3](#), is adapted with permission from the tutorial portions of the Ph.D. thesis by D. Morley [\[Mor101\]](#).

There are two basic types of switching that may be used in transport networks: packet switching and circuit switching. In a packet-switched network, the information flowing from one node to another is broken down into sequences of packets at the sending node before being transmitted over the network to the receiving node(s). When a packet arrives at a switching node, it waits in a queue to be transmitted over the next transmission facility en route to its destination. At the receiving node, the packets are reassembled to reconstruct the original information stream. Because the packets occupy the full capacity on a transmission facility only for their duration, the transmission capacity can be shared over time with many other connections (or sessions). This is referred to as *statistical multiplexing*. Statistical multiplexing takes advantage of the strong law of large numbers to obtain efficiency in bandwidth use. In this context, the principle states that for a number of independent flows, the bandwidth necessary to satisfy the needs for all of the flows together stays nearly constant, and much less than the sum of their individual peak rates, even though the amount of traffic in individual flows can vary greatly. Intuitively, the averaging effect is easy to appreciate, especially when delay can be introduced through a buffer to queue up the access to the transmission link. At any moment a few applications could be increasing their traffic while other applications are reducing their traffic. The larger the number of sources, the more that individual uncorrelated changes balance each other out to approximate a near-constant total bandwidth requirement. If a large amount of queuing delay is used, the constant bandwidth approaches the overall average rate of all sources. In contrast with TDM multiplexing, the total bandwidth required is the sum of each source's peak bandwidth. Packet switching is thus particularly well suited for data sessions that are characterized by short bursts of high activity followed by long periods of inactivity. On the other hand a central issue with packet switching is that queuing delays at the switching nodes are difficult to control and packet loss can occur from buffer overflow.

Under TDM each connection is allocated a given amount of transmission capacity (usually in both directions) between the origin and destination nodes. Therefore, once the path (or circuit) has been established, the connection has a guaranteed transmission capacity through the network for its entire duration. Unlike packet-switched networks, the capacity allocated to individual connections cannot be used by other connections during inactive periods. The sum of the capacity allocated to all paths on a given transmission facility cannot exceed its total capacity. Thus, if a transmission facility is fully allocated it cannot accommodate any new connections. If no other paths with the required capacity can be found through the network, a new connection request must be rejected or *blocked*. In contrast, an overload situation in a packet-switched network results in increased queuing delays and potential loss of data due to buffer overflows.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

1.0.4 Concept of Transparency

Service layer networks also differ from transport networks in that they are usually designed for a specific type of service and typically contain user signaling and control functions for setting up and tearing down calls or connections. In contrast, transport networks provide bulk transmission of aggregate information streams independent of the type of end-user services supported. For this reason, they are sometimes referred to as "backbone" networks. One of the ideals of transport networking is that paths in the transport layer would be completely transparent to the rate or format of payload applied to them. In other words, if a transport path was established from A to B, users could put any form of payload they wanted on the path without encountering technical problems such as synchronization failure or high error rates. Conceptually the ideal transport network would thus provide nothing more than "ideal wires"—on demand—to its service layer clients.

As conceptually simple as the idea of a wire is, it is actually quite difficult to even approximate a lossless, constant-delay, infinite bandwidth path that would support any payload signal format at all. This ideal was not achieved in the plesiochronous digital hierarchy (PDH, i.e., pre-SONET), where only completely stipulated tributary formats (DS-1, DS-2, etc.) could be carried. In SONET, transparency was somewhat more closely approximated in that a variety of payload mappings were defined to adapt non-traditional payloads, such as an FDDI or Ethernet LAN signal, into the payload envelope of suitable high-rate SONET OC-n signals. The ideal is closer to being realized with optical networking, where an almost lossless and otherwise low-distortion, near constant-delay, optical path approximates an ideal "wire," up to a certain distance, onto which almost any form of payload can be applied. The distance limit is fundamental because we cannot indefinitely preserve all attributes (phase, frequency, waveshape, amplitude, polarization, and most important of all, signal-to-noise ratio etc.) of an analog signal as it is transmitted over an increasing length of fiber and number of optical amplifiers and possible wavelength-changing transponders. Thus, a more practical notion is that of *digital transparency* [Dixi03], p.105 where any format or rate of digital payload signal, up to some maximum working bit rate is accommodated. Such digital transparency is being provided by recent developments such as digital wrapper and GFP, that we discuss further in [Chapter 2](#).

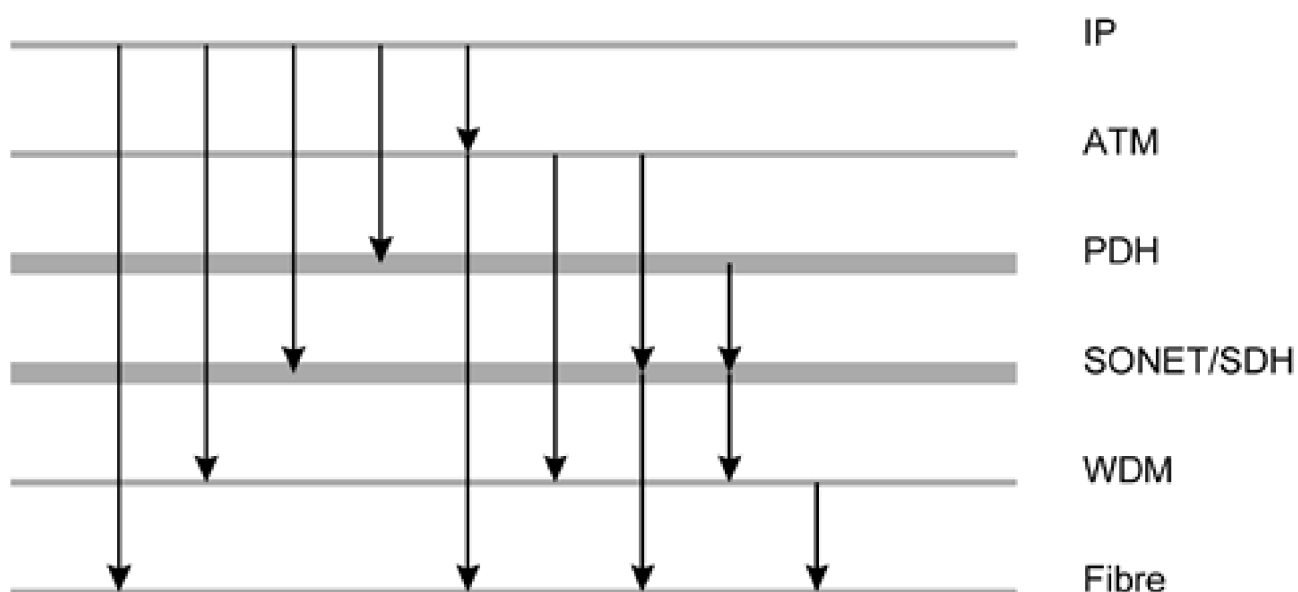
[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

1.0.5 Layering and Partitioning

Over many different multiplexing and transmission technologies, the basic routing and switching functions that they employ are logically equivalent. The main difference is the unit for allocating capacity. In SONET the basic unit of allocation is an STS-n channel, whereas in optical networking it is a wavelength (or to be precise, an optical channel with a specified bandwidth in Hz). There are other important differences but from a functional perspective, these networking technologies can be modeled in a generic fashion for many purposes using the same abstract concepts. This has several implications. First, because the basic functions are equivalent in nature, the same types of network elements and architectures are implemented across the range of technologies. For example, the survivable ring architectures first implemented in SONET can also be implemented in the optical network layer. Similarly electronic digital cross-connects (DCS) for SONET remain logically equivalent in many regards to optical cross-connects (OXC) in the optical network layer. The adjacent layers in the network form client-server relationships, in which transport layers perform signal multiplexing, transport and routing for one or more client layers. For example, the SONET/SDH layer can accept payloads directly from either the PDH, ATM or IP layers. In turn, the resource requirements from the SONET layer become payloads for the WDM layer. [Figure 1-4](#) shows most of the currently possible inter-layering transport relationships.

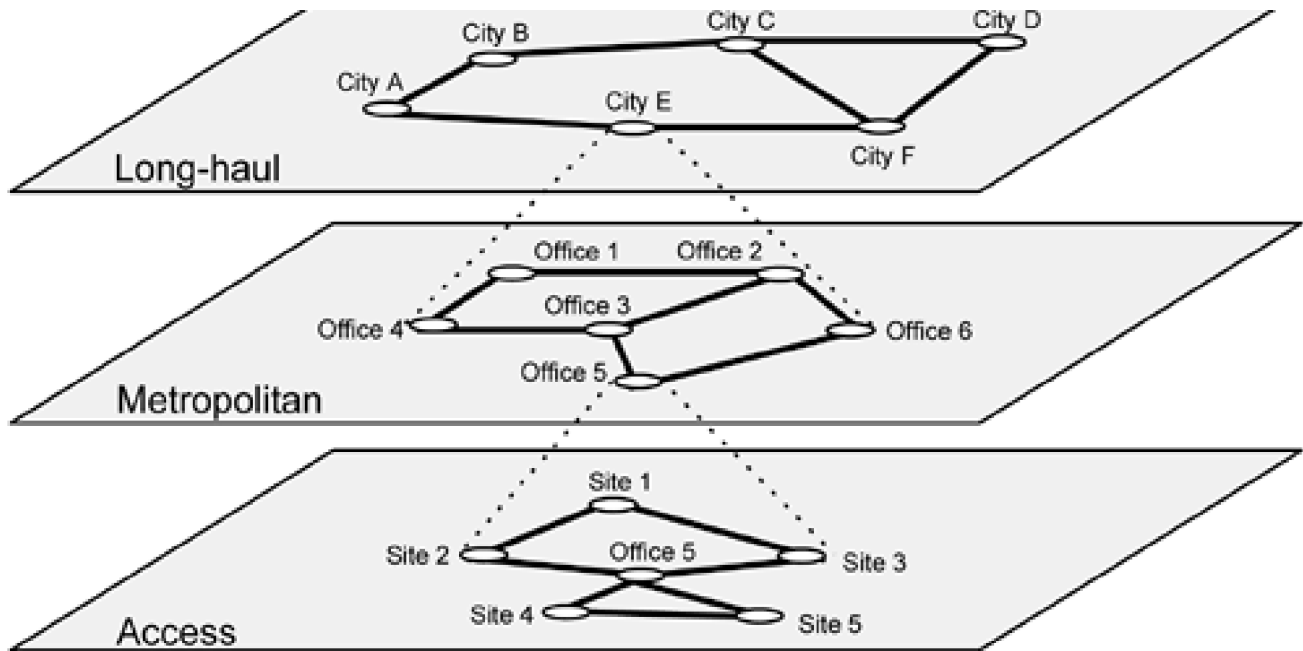
Figure 1-4. Examples of possible client/server associations in a layered transport network.



Each network layer can also be partitioned horizontally into tiers or subnetworks according to geographic and/or administrative boundaries, as in [Figure 1-5](#). This further facilitates design and operation and allows for different survivability schemes to operate autonomously by separation in space. In practice, this partitioning usually reflects the differences in demand distribution, cost structures and topological layout within a network layer. For example, it is common practice to partition a network layer into separate access, metropolitan inter-office (or metro) and core (or long-haul) subnetworks. In an access subnetwork, most demands originate at remote switching offices and customer premises and terminate back at a main switching office (or *hub*). A metro subnetwork connects main switching offices (or other points of concentration) within a metropolitan area and demands are typically more uniformly distributed. Because span distances in access and metro subnetworks are typically less than 25 to 50 km, nodal equipment costs (e.g., ADMs, DCSs) usually dominate total network costs. Long-haul subnetworks, on the other hand, usually connect metropolitan areas on a national or international scale. Span distances are much greater and distance-related costs for cable installation, amplifiers and regenerators can dominate the total cost.

Figure 1-5. Partitioned view of a transport network.





[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

1.1 Plesiochronous Digital Hierarchy (PDH)

The two most common TDM standards for transport networking in North America are the plesiochronous digital hierarchy (PDH) and the synchronous optical network (SONET) standards. These play such an important historical (and ongoing) role in transport networking that an appreciation of them is essential. The first TDM carrier system to achieve widespread use was the T-1 carrier system. Initially, T-1 systems were deployed in metropolitan areas to increase the number of telephone trunks between switching offices over existing copper pairs using pulse code modulation (PCM) to convert speech from analog to digital form. The resulting 64 kbps digital signal is now called a Digital Signal-0 (DS-0). A T-1 voice multiplexer (called a "channel bank") combines twenty-four DS-0s by byte-interleaving the 8-bit codewords from all 24 voice codecs every 125 μ s. A single framing bit is also added to each *frame* of 24 coded voice signals. Frame alignment on the 24 channel structure is done by searching for a known pattern on the single framing bit. The rate of the aggregate signal is 193 bits every 125 μ s or 1.544 Mbps. This is called a Digital Signal-1 (DS-1). A similar PCM carrier system, known as E-1, was developed in Europe. In an E-1 system, the basic signal rate is also 64 kbps, but thirty voice channels are multiplexed by byte-interleaving samples from each channel and adding an additional 16 bits of overhead for framing and signaling. Because the duration of each 256-bit frame is 125 μ s, the data rate of the aggregate E-1 signal is 2.048 Mbps. The traditional DS-1 or E-1 voice payload is encoded via slightly different analog to digital conversion rules.

As higher digital transmission rates became possible, standard rates were developed in North America and Europe for digital signal multiplexing. [Table 1-2](#) lists these "PDH" standard signal rates and the number of lower-speed *tributary* signals multiplexed into each higher-rate digital signal. This family of signals gets the name PDH in reference to the independent timing of each tributary signal and the technique of variable pulse stuffing employed to multiplex such "near synchronous" tributary signals into a composite serial signal. For example, a DS-2 signal is constructed by byte-interleaving four DS-1 tributary signals and adding some additional overhead bits for signaling and to absorb slight frequency offsets between the tributary payloads and the composite line signal. Seven DS-2 signals are then multiplexed to form a DS-3 signal and so on. Initially, these signals were carried over twisted copper pairs, coaxial cable, and microwave radio and the first generations of fiber optic transmission systems up to about 1988. DS-1 and DS-3 signal formats are, however, still widespread. Although almost eliminated from direct use as transport signals today, DS-1 and DS-3 remain the most common and profitable private line service offerings to customers for most network operators. In modern use the internal structuring of both formats has evolved, however, to support various clear-channel and mixed voice-data uses.

Table 1-2. North American and European Plesiochronous Digital Hierarchies

Signal Level	Data Rate (Mbps)	Composition	Signal Level	Data Rate (Mbps)	Composition
DS-0	0.064 (64 kbps)	8000 8-bit bytes / sec " μ -Law encoded"	E-0	0.064 (64 kbps)	same as DS-0, but "A-law" encoded
DS-1	1.544	24 DS-0s	E-1	2.048	32 DS-0s
DS-2	6.312	4 DS-1s	E-2	8.5448	4 E-1s
DS-3	44.736	7 DS-2s	E-3	34.368	8 E-2s
DS-4	274.176	6 DS-3s	E-4	139.264	4 E-3s

A disadvantage of the pulse-stuffing synchronization scheme used in PDH is that any high speed signal must be completely demultiplexed to access any of its individual tributary signals for add or drop purposes. For example, to drop a DS-1 from a DS-3, the entire DS-3 signal must undergo two stages of demultiplexing, first to the DS-2 rate and then to the DS-1 rate. Once the desired signal has been dropped, the remaining DS-1s must be remultiplexed to the DS-3 level. Another limitation is the lack of a uniform set of functions for network performance monitoring, fault detection, provisioning and other network management features and capabilities.

1.2 SONET / SDH

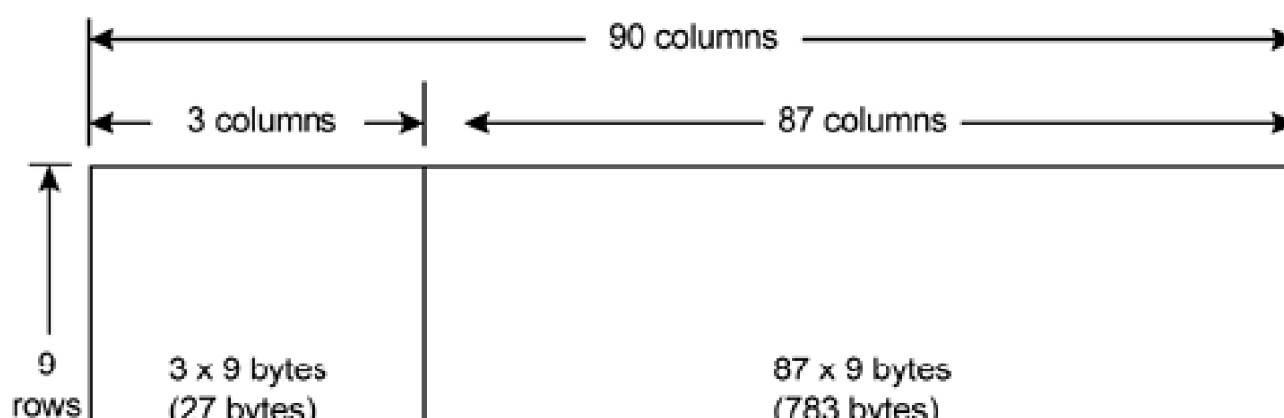
Just prior to the advent of SONET standards, systems based on PDH rates and pulse-stuffing multiplexing techniques had reached 12 x DS-3 (Nortel FD-565) and 24 x DS-3 (Rockwell LTS-1200) rates. But these systems used signal formats, monitoring methods, modulation techniques, laser types, and so on, that were proprietary to each vendor. The emerging "tower of Babel" above DS-3 rates strongly motivated development of SONET standard by network operators. They wanted SONET to standardize all vendors so that transmitters, receivers, regenerators, cross-connects, network controllers, and so on, would all interoperate as separately purchased piece-parts. But SONET also reflected accumulated experience with remote monitoring and network operations: the philosophy, made possible on fiber, was for SONET to be deliberately lavish on overheads for monitoring, fault sectionalization, voice orderwire, protection switching, remote provisioning, etc., confident that the operational benefits would far outweigh any added bandwidth cost.

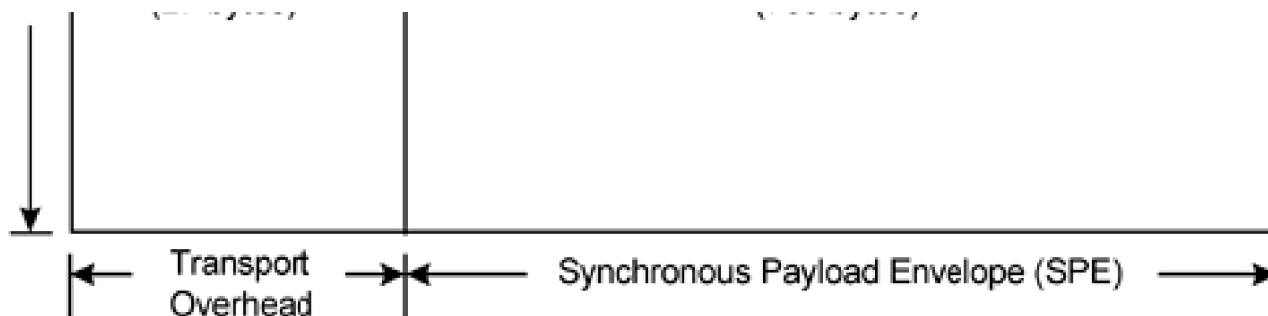
SONET also provides for direct DS-0 visibility within a high speed multiplex format by using synchronous byte-oriented multiplexing at all levels of the signal hierarchy. Any tributary signal in a SONET signal can be accessed without successive demultiplexing steps as in PDH. This requires that all sources of tributary signals must be frequency synchronous. SONET thus requires precise network-wide frequency synchronization. This is achieved through atomic reference clocks at selected nodes and schemes of phase-locked frequency synchronization in other nodes. Precise time from GPS receivers also aids in maintaining synchronization. The international version of SONET is the Synchronous Digital Hierarchy (SDH) [ITU96]. SONET / SDH multiplexing and transmission systems are widely deployed world-wide and evolving to include data-oriented enhancements covered in [Chapter 2](#).

SONET provides an essentially complete example of all the basic logical elements of transport networking. That, plus its general pervasiveness in transport networking, are the reasons we include a limited overview of the topic as basic background. For more in-depth treatments of SONET/SDH, see [Sill96], [SeRe92], [Alle96] or the standards themselves [AN95a] and [AN95b]. Once the basic transport network elements are seen in the context of SONET, the functions of their DWDM networking counterparts are fairly easily anticipated, although not yet standardized. Indeed, recent developments in optical networking, covered in [Chapter 2](#), take on increasingly SONET-like attributes as they develop toward practical networking use, because the same fundamental problems are faced again. An optical layer cross-connect evolves to be almost a functional clone of a SONET DCS, for instance. What the DCS did for timeslots, the optical cross-connect does for lightwave channels, etc. Optical layer rings are similarly almost logical clones of SONET rings, and so on. SONET, therefore, remains relevant both commercially and conceptually as a vehicle for studying the basic logical functions and concepts involved in any transport network.

The basic building block in SONET is the synchronous transport signal-level 1 (STS-1). The STS-1 frame provides nine rows by 90 columns of 8-bit bytes as shown in [Figure 1-6](#). The STS-1 frame time is 125 μ s, reflecting the basic period for sampling speech at 8,000 times a second. With a total of 810 bytes the STS-1 bit rate is 51.840 Mbps. The STS-1 frame provides transport overhead and a synchronous payload envelope (SPE). The transport overhead occupies the first three columns of the frame and is further divided into line and section overheads, that provide signal framing, line identification, performance monitoring, and voice and data channels (used for provisioning and maintenance). The SPE occupies the remaining 87 columns by nine rows. The first column of the SPE is used for path overhead functions such as end-to-end performance monitoring and path identification. The other 86 columns are for payload signals.

Figure 1-6. The SONET STS-1 frame structure.





An STS-1 SPE can carry a single DS-3 (44.736 Mbps) or it may be subdivided into smaller envelopes to provide backwards compatibility with lower bit rate PDH signals. For example, a DS-1 signal can be carried within an STS-1 SPE by mapping it into a SONET virtual tributary—1.5 (VT1.5). An STS-1 SPE can carry up to 28 VT1.5s. Four VTs are defined in SONET: VT1.5, VT2, VT3, and VT6. They are intended to carry DS1 or E1, DS1C, and DS2 payloads, respectively. Each VT has its own overhead bit package, a separate unit within the STS-1 signal. The overhead plus fixed stuffing to permit a fit into the STS-1 frame means the VT1.5 actually has a line rate of 1.728M bps. A VT group carries four VT1.5s, three VT2s, two VT3s, or one VT6. An STS-1 SPE is filled with seven VT groups, a path overhead, and two vacant columns. VT groups with different types of VTs can be mixed within a single STS-1. This gives the capability to accommodate both the DS-1 signal and the E-1s.

SONET signals at rates that are multiples of the STS-1 rate are obtained by byte-interleaving a whole number of STS-1s. Services that require clear-channel multiples of the STS-1 payload (for instance an STS-3c for ATM) can be transported by *concatenating* several STS-1 signals together. The resultant signal is designated an STS-Nc. The difference is that an STS-N is a simple assembly of N intact and separate STS-1s each with its own payload and overheads. The STS-Nc has a single N times payload field and a single overhead stream. Higher rate signals can be comprised of any combination of lower rate individual STS-1s or concatenated signals. For example, an STS-12 signal can be created from 12 STS-1 signals or 4 STS-3c signals or any other combination of STS-1 and STS-3c signals that equals STS-12. Prior to transmission, the STS-N signal is scrambled and converted to a corresponding optical carrier signal (OC-N). Scrambling ensures certain properties for reliable physical layer transmission, such as dc balance and edge transition density for timing recovery. [Table 1-3](#) lists the most commonly used SONET and SDH signal levels and their transmission rates. SDH standardizes only multiples of the 155.35 Mb/s rate of the OC-3 SONET signal, and refers to the resulting level as Synchronous Transport Modules (STM). SONET's concatenation ability is important for carrying single high-bit rate streams of Internet Protocol data packets. The "wider" the single pipe that can be provided, the better the throughput versus delay and statistical multiplexing efficiency is. There are some issues, however, in that some large OC-N concatenations, such as OC-48c (~2.5 Gb/s) do not provide very efficient bandwidth matches to the native formats of IP traffic such as Gigabit Ethernet (GbE). This leads to the technique of virtual concatenation in [Chapter 2](#). Above the top rate of OC-192 defined in the original SONET standards, the industry is developing OC-768 (STM-256) systems as well. The corresponding rate of ~40 Gb/s is considered by many to be the highest serial rate that is technically and economically feasible for electrical processing and switching. Above this rate recourse is made to separate wavelengths to carry multiple 40 Gb/s payloads, and then to multiple fibers when the feasible number of wavelengths per fiber is reached. Thus, time, frequency and space are all dimensions used for multiplexing to realize transport capacity requirements.

Table 1-3. SONET / SDH Digital Signal Hierarchy

SONET Signal Level	Optical Signal	Data Rate (Mb/s)	SDH Equivalent
STS-1	OC-1	51.84	none
STS-3	OC-3	155.25	STM-1
STS-12	OC-12	622.08	STM-4
STS-24	OC-24	1244.16	STM-8
STS-48	OC-48	2488.32	STM-16
STS-192	OC-192	9953.28	STM-64

Table 1-4. Some traditional and non-traditional payloads accommodated by mapping into SONET/SDH containers

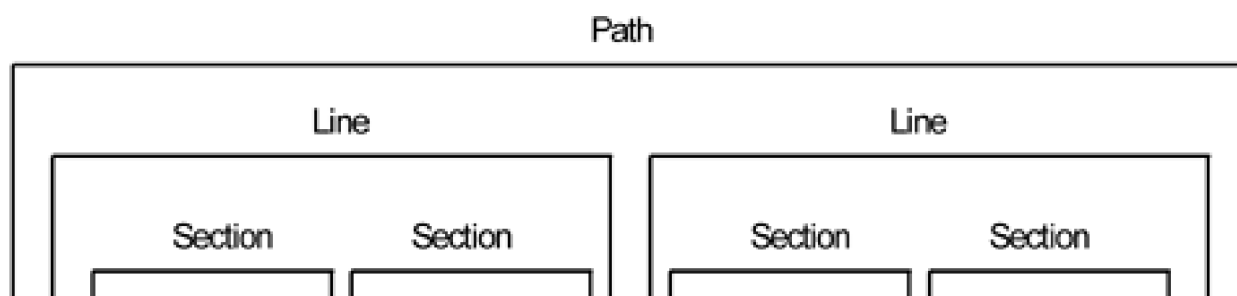
Payload	SONET Structure Used
10 Gb Ethernet (IP)	OC-192c
HDLC / IP (PoS)	OC-48 to OC-192
FDDI	STS-3c
DS1 SF, DS1 ESF	VT 1.5 (28 / STS-1)
DS3 (PDH)	STS-1
DS3 (C-bit parity)	STS-1
ATM (B-ISDN)	STS-3c

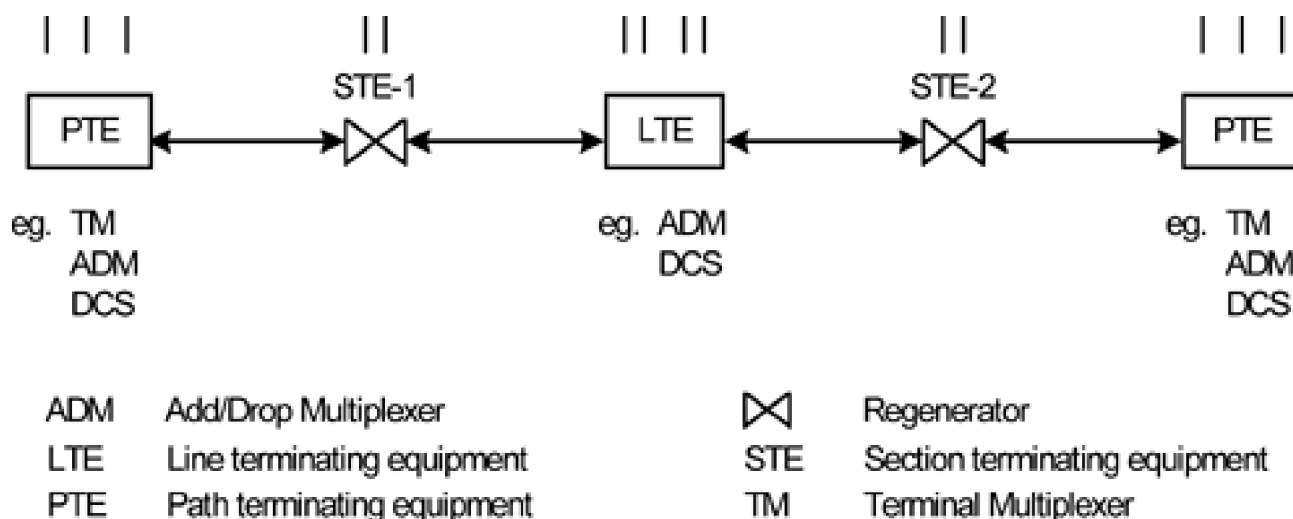
[Table 1-4](#) summarizes some SONET payload mappings that have been so far defined. PoS refers to Packet-on-SONET. The 10 Gb Ethernet to OC-192c mapping is primarily intended for direct wavelength transport for wide-area LAN and data-center interconnect applications.

1.2.1 SONET Overheads

The SONET overhead and transport functions are divided into three logical layers or domains of visibility: section, line and path. The extent of each of these domains is illustrated in [Figure 1-7](#). The section overhead provides framing and performance monitoring for the STS-N signal and local voice "order-wire" and data communications channels. Network equipment that terminates the section overhead is called section terminating equipment (STE). This typically includes regenerators, terminal multiplexers, add/drop multiplexers and digital cross-connect systems. Regenerators are used at intermediate points along the line to extend transmission distance by converting the optical signal to the electrical domain, retiming and reshaping the STS-N signal and retransmitting it in optical form. Optical amplifiers may also be used between regenerators to increase the power level of the optical signal without optical-to-electrical (O/E) conversion and further extend transmission distance. Compared to amplification (where the signals stays in optical domain) regeneration is a much more costly process in general. The line overhead provides performance monitoring of individual STS-1s, SPE pointer adjustment, and voice and data communications channels for operation, administration, maintenance and provisioning (OAM&P) purposes. Pointers are the SONET mechanism for accommodating slight phase wander or frequency offset between the OC-N line signal timing and a given SPE. Network equipment that terminates the line overhead is called line terminating equipment (LTE). The path overhead is transported along with the SPE until it is demultiplexed. A SONET path is a network connection at a given data rate between the point where the SPE is assembled and the point where it is disassembled. SONET equipment that originates/terminates the SPE and the path overhead is called path terminating equipment (PTE). Line and path terminating equipment can be any SONET equipment except a regenerator. Although terminology and some details change, the basic ideas of section, line and path domains within transmission systems is so important that it will likely be replicated in systems for DWDM as well. GFP and Digital Wrapper developments (in [Chapter 3](#)) both have a corresponding set of concepts.

Figure 1-7. SONET section, line and path supervisory domains and equipment types.

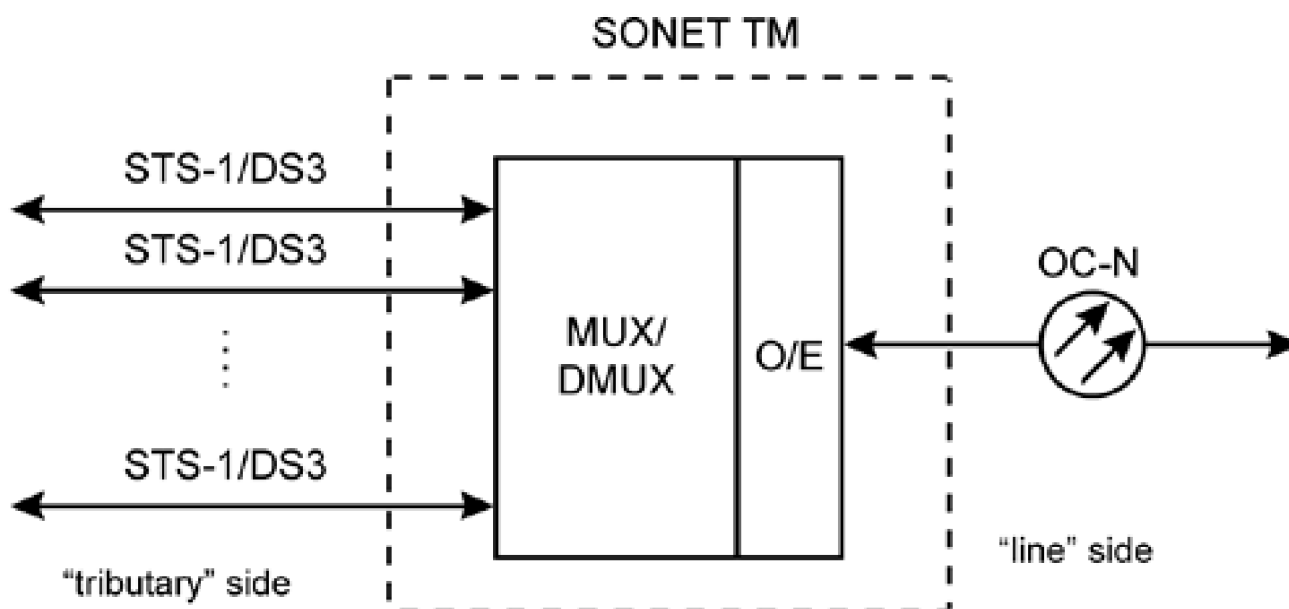




1.2.2 Generic SONET Terminal Multiplexer

A generic *terminal multiplexer* (TM) is a device that terminates several tributary signals (e.g., DS-1, DS-3), assembles them into one or more higher-rate SPEs, and converts the resultant electrical STS-N signal into an optical carrier OC-N signal for transmission. A functional block diagram of a TM is shown in [Figure 1-8](#). Most TMs support a range of electrical and optical tributary types and provide for either single non-redundant or protected (e.g., a 1+1 "hot standby" arrangement) for line interfaces (e.g., OC-3, OC-12, etc.). Terminal multiplexers may be used in traditional point-to-point systems or as part of linear add/drop chains, as suggested in [Figure 1-7](#). TMs are as *line terminating equipment* (LTE) and said to "terminate" the optical line signal.

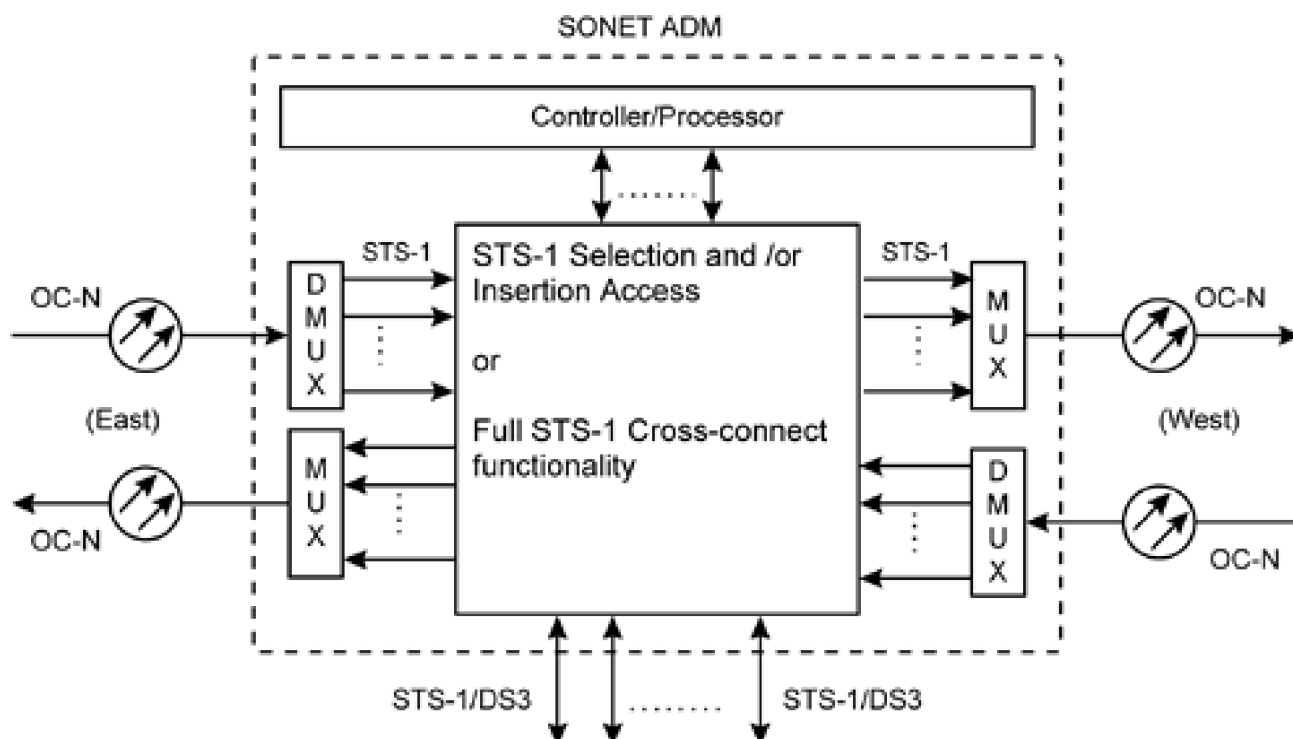
Figure 1-8. Functional block diagram of a terminal multiplexer (TM).



1.2.3 Generic SONET Add/Drop Multiplexer

An *add/drop multiplexer* (ADM) is similar to a TM except there are two line-rate interfaces as shown in [figure 1-9](#). ADMs may be used at intermediate sites along linear add/drop chains to allow tributary signals to be added or dropped from the line signal or to pass through en route to their final destinations. More often ADMs are used in survivable ring architectures where the SONET K1/K2 byte-protocol supports rapid line-level protection switching. Like TMs, most ADMs also accept a variety of electrical and optical tributary (or *low-speed*) interfaces. In a SONET ADM, signals that pass through the site do not need to be demultiplexed and then multiplexed back into the line signal. In comparison with back-to-back TM arrangements, this provides cost savings and improved reliability and is the prime reason that ADM ring-based networking is preferred over point-to-point networking using only TMs. ADMs are sometimes also equipped with time-slot interchange capability to allow flexible assignment of any tributary signal to any eastbound or westbound SPE.

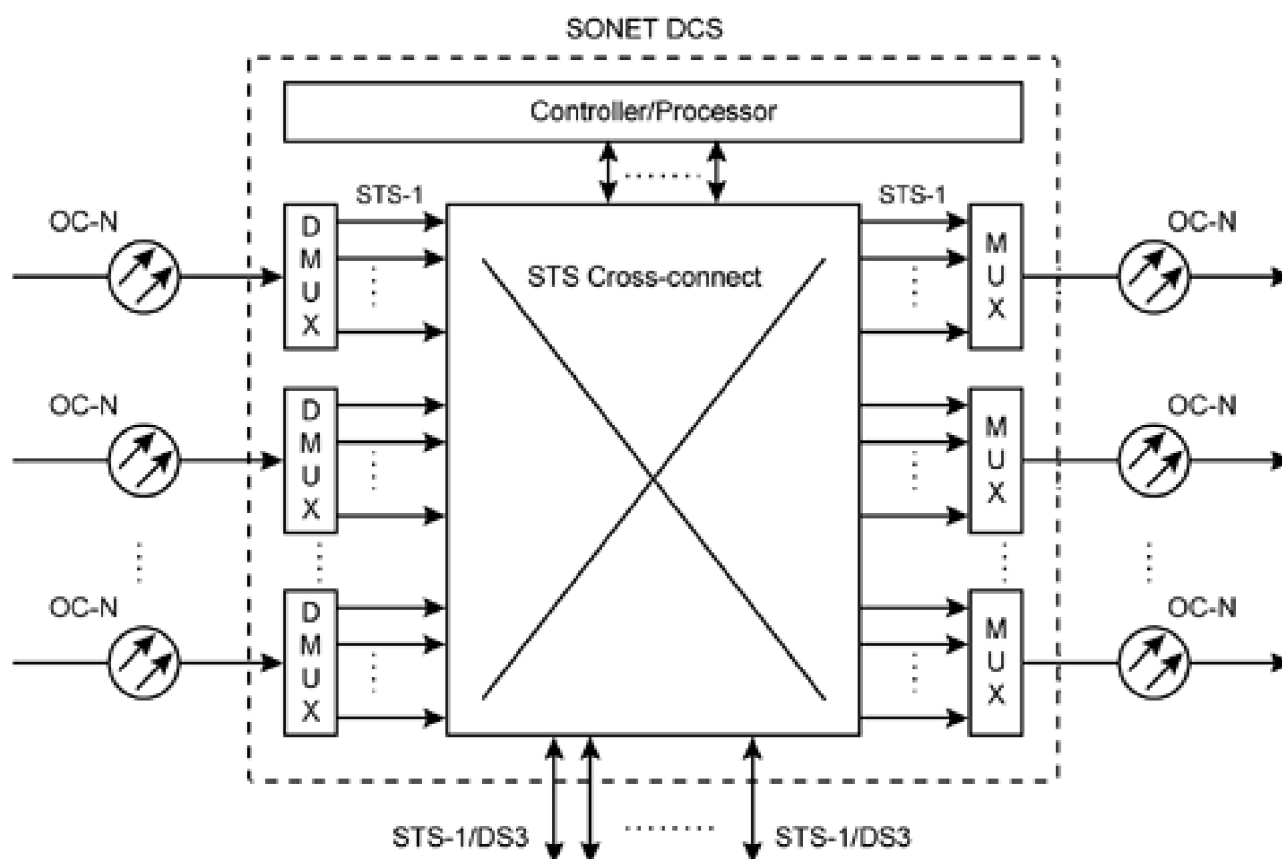
Figure 1-9. Functional block diagram of an ADM.



1.2.4 Digital Cross-Connect Systems

A *digital cross-connect system* (DCS) can be thought of as a circuit switch for digital carrier signals. A SONET (or PDH) DCS accepts various electrical and optical carrier signals, accesses the individual tributary signals (e.g., VTs, STS-1, STS-Nc) in electrical form and switches them from incoming to outgoing facilities. In addition to this switching function, DCS may provide local add/drop and multiplexing/demultiplexing functions. Industry jargon identifies two basic types of DCS: A *broadband DCS* (B-DCS) typically accepts OC-N and DS-3 rate signals and cross-connects them internally at the STS-1 rate. An STS-Nc cross-connection is put into effect through multiple STS-1 connections through the core. A DS-3 connection is made by mapping to/from an STS-1 in a DS-3 interface port card. None of the payloads of these transport signals is accessed, manipulated or altered. Only the routing of the signal is affected. For instance, STS-3c arriving on an OC-96 port may be switched through the B-DCS core to leave on a different fiber system, but it stays as an intact STS-3c rate signal unit. Thus, aside from the fact that multiple types of tributaries may be accepted as inputs, the B-DCS is functionally a pure "space-switch": signals arrive on one fiber system and are switched in space to depart on another fiber system. This simple functionality is important in the transport network backbone because it imposes the least propagation delay on signal paths and, being far simpler than cross-connects that demultiplex each signal for switching (to follow), it is also of higher availability. A functional block diagram of a B-DCS is shown in [Figure 1-10](#).

Figure 1-10. Functional block diagram of a B-DCS.



The major difference between a DCS and an ADM is the total capacity handled. A typical B-DCS may cross-connect up to 8192 DS-3 / STS-1 (or equivalent) ports in a fully non-blocking manner. A primary use of B-DCS functionality in a mesh-based network is flexible provisioning of transport signal paths. For mesh-based survivable networking, the B-DCS (or its optical cross-connect equivalent) is the primary network element to support mesh restoration or protection schemes. If a network is ring-based, B-DCS may also be used for the automated management of signals that transit from one ring system to another en route. The latter B-DCS will usually not have to be as large in terms of total port capacity, however, as in a mesh-survivable network because only working signal flows are handled. If survivability is derived directly from mesh-based rerouting principles the B-DCS are usually larger than for ring-interconnect applications because they must terminate the spare capacity of the network as well. In the SONET era, the cost for suitably large cross-connects, and the fact that a DCS core of suitable ultimate size has to be established as an up-front cost, was a significant impediment to mesh-based networking relative to rings which are based on lower cost ADMs and offer a "pay as you grow" characteristic. Optical cross-connects (OXC) for DWDM-based networking are the logical counterpart of B-DCS. Whereas a B-DCS might terminate OC-48s or OC-192s and switch STS-1s, an OXC will terminate fibers and switch wavelength paths in space without altering the payloads in any way. Optical cross-connects may not always be directly named as such. Terms such as "optical core director" may pertain instead to describe a wavelength path cross-connect that is in all other regards a complete functional equivalent for wavelength channels in the lightpath managed layer to a SONET B-DCS in the STS-1 managed layer.

In a *wideband DCS* (W-DCS) the constituent payloads on STS-1s arriving off of fiber systems are also accessed and manipulated at the DS-1 or VT-1.5 rate. The main use of the W-DCS is for *signal grooming* at hub locations or edge nodes. At an edge node, numerous lower-rate payloads are aggregated and directed onto carrier-rate signal paths through the transport network. At a hub, carrier signals are demultiplexed from incoming facilities and regrouped with other lower-rate signals sharing the same destination (or next hop direction). Both functions are called grooming. Grooming is the process of selecting and consolidating lower rate service payloads into higher-rate outgoing signals to allow more efficient use of the high speed facilities. Grooming is fundamental to creating high utilization of transmission carrier facilities by combining tributaries from partially filled incoming facilities into a smaller number of outgoing facilities. Grooming may also be done to collect together signals of a common type for enhanced service management or monitoring and to support multiple classes of service by segregating demand by service type, destination or protection category. The W-DCS functionality may be standalone or part of a multi-service provisioning platforms (MSPP) that will interface a large range of signal types, both voice and data, and both groom and multiplex these payloads onto a single higher-rate carrier such as an OC-48, and launch it onto a corresponding path to its destination over the transport network, or to a hub node for further routing. Note that grooming is in many ways the circuit-switched counterpart to statistical multiplexing, although without the congestion implications. When data is being both groomed from typically low utilization access pipes, and statistically multiplexed to higher utilization levels, the overall process is often called "aggregation." A typical commercial name for a device that performs the W-DCS function, but adds data aggregation and optical OC-n or GigE output formats would be an "edge director."

The important overall picture is that regardless of the technology, basic architectural principles lead to their being two fundamental types of

"cross-connect" functionality:

- Edge or hub devices that collect lower speed payloads together based on common routing (or common end destinations) and multiplex up to rates suitable for application to an OC-N path or lightwave path through the transport network. Typically in edge grooming the payloads being multiplexed arrive in lightly-loaded access transmission systems.
- Core or backbone devices that interconnect transmission channels at the OC-N or lightwave channel levels to realize the required transport paths and, in a mesh-survivable network, to perform the rerouting functions that protect these paths.

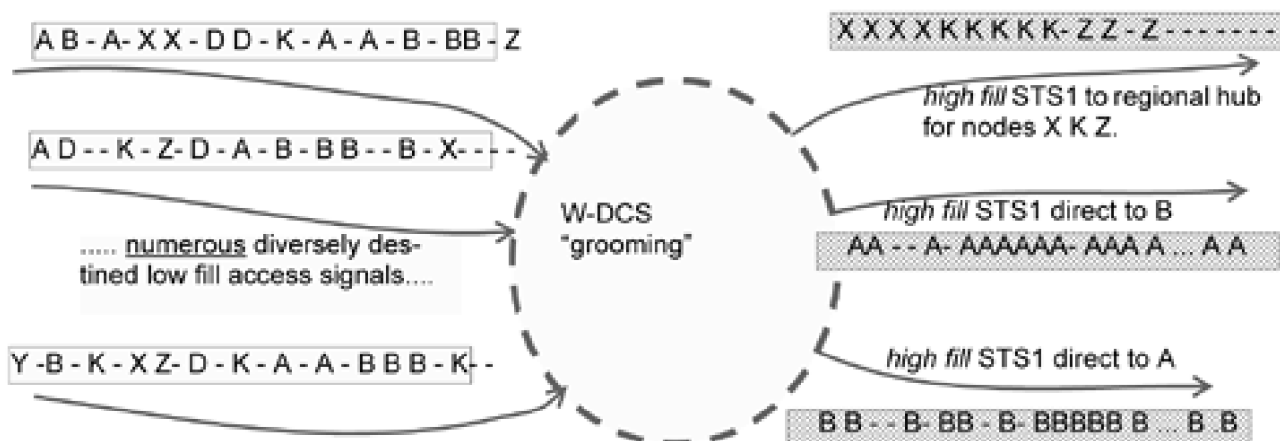
In practice, commercial cross-connect or service management platforms may integrate these two basic functions because almost every node has some local add/drop requirements as well as serving as a transport level switching point. For more background on DCS functions and requirements in general, see [\[Bell88\]](#) and for specifics of the role of DCS systems in mesh restoration see [\[Bell93\]](#).

1.2.5 Hubs, Grooming and Backhaul

Anyone who has taken a trip by air that required them to take a commuter flight to a major hub airport can relate to the concept of grooming and the related role of a *hub* node. What the airlines want to do is make sure that their biggest planes fly well-loaded and over the long distances for which they are optimized. It would not be economic, nor provide desired service quality, to have a Boeing 747 land every 100 miles to let a half a dozen people on or off. Aircraft of this calibre are used on long-haul routes between major "hubs" on routes for which a suitable number of passengers sharing the same destination (or next hub) can be grouped together. The smaller commuter planes, often lightly loaded, bring people from many small centers into the hub. Each commuter plane is carrying people with different ultimate destinations. But over all the commuter arrivals at the hub, everyone who is that day heading to Southeast Asia are grouped together for the hop to, say, Hong Kong. The process of collecting together passengers that can usefully all share the common hop to Tokyo is called *grooming* in the transport context. In an analogy to communication networks, the lightly loaded commuter planes are the access systems, the grouping together of everyone going to S.E. Asia is grooming, the combining into a well-filled 747 is multiplexing (and the 747 itself is a high-capacity transport path).

[Figure 1-11](#) illustrates this role of a hub node with a notional W-DCS. In a typical application the W-DCS may be programmed to pick and choose individual DS1s out of all the incoming DS3s and assemble them into STS-1s that are efficiently filled with co-destined DS1s or DS1s that all share the need for transport to the same intermediate node in another region. The nodes that perform this grooming function are called *hubs*. Hub sites are important factors influencing the (apparent) demand patterns seen in the logical and system layers and are a consideration in the design of both ring- and mesh-based survivable networks. With effective grooming, transport signals such as a SONET OC-n, or a wavelength bearing several OC-n, can be efficiently filled with lower-speed connections near their points of origin, all of which share a common destination or a common intermediate hub. This allows optical (or in general transport-layer) bypass at intermediate nodes with considerable savings in add/drop costs and electronics costs at those nodes.

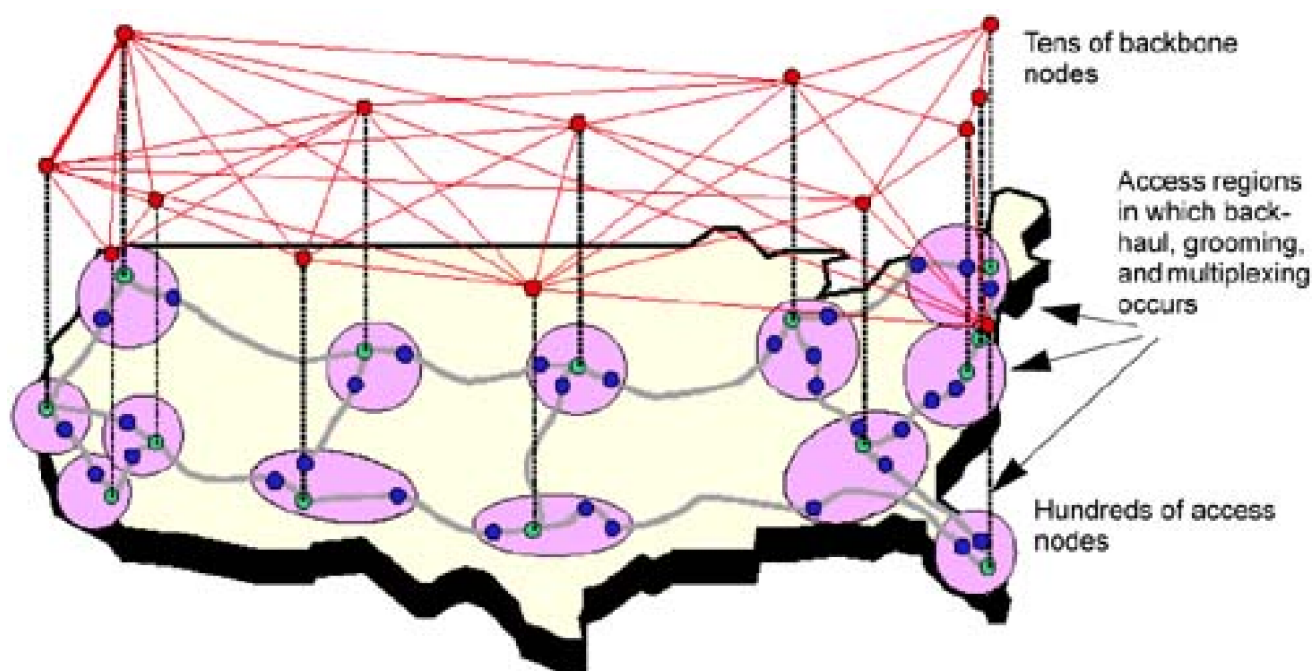
Figure 1-11. Grooming diverse demands from lightly loaded access facilities into well-filled transport signals to other regional hubs or destinations.



Backhaul refers to the process of bringing lower speed signals to the hub site to be groomed into the more efficient long-haul carrier

signals leaving the hub. Viewed solely from the lower-speed payload standpoint, going to the hub may actually route it away from its ultimate destination. Again the airline analogy gives an essentially exact explanation of the process, and the trade-offs involved. One's individual destination may be Anchorage, Alaska, say. But if you want to get there from Edmonton, Canada you are first routed south to Seattle (directly away from the destination). The reason is that it is only at the Seattle hub that enough individual travellers going to Anchorage can be collected from the surrounding region to fill a 767-class plane from Seattle to Anchorage at a suitably high fill factor. The backhaul is illustrated by the fact that Edmonton is partway north between Anchorage and Seattle, so there is a significant backhaul component in this routing. The identical concepts explain why an individual DS-1 may be routed on a lightly loaded regional DS-3 via a path involving a backhaul over local access transmission systems to a hub where it may then join an entire OC-48 going to its destination (or to another hub in the destination region). The corresponding view of how access regions, hubs, and backbone transport are interrelated in a network as a whole is illustrated in [Figure 1-12](#).

Figure 1-12. Concepts of access, hubs, grooming, backhaul, and transport. (source: [FrSo01](#))



Historically, the practice seems to have been one of establishing cross-connect hubs at the largest regional central offices. Once such hubs are given, grooming has been primarily treated as the problem of finding aggregations of lower-speed signals that can economically justify establishment of certain larger outgoing high speed signal containers to each other hub. Each such establishment decision is made independently. Typically, 60 to 75% fill is a criteria that turns out to be economic for establishing a new high-speed signal to a corresponding hub. If sufficient traffic cannot be aggregated to justify a direct higher-speed connection from the current node, the lower-speed signals are routed indirectly via another hub that must handle them again at their lower speed. In a future optical mesh network there is considerable research still to be done on globally optimized strategies of hub placement for access-grooming, backhaul routing, and optical bypass savings.

To appreciate the issues and tie into this related research, see [\[ZhMu02\]](#), [\[ZhBi03\]](#), [\[MoLi01\]](#) and [\[SiSu00\]](#) (p.286). In this book we consider problems of survivable transport design where the set of high speed signal path requirements is already determined as output from the grooming decision process. In reality, however, details of the transport layer design determine certain costs that will influence the grooming strategy and the grooming decisions define which wavelength paths are needed in the transport layer. This should be kept in mind as the methods of this book may be iteratively solved in conjunction with grooming methods developed elsewhere to approximate a joint optimization of both grooming and transport architectures.

1.2.6 Fundamental Efficiency of Edge Grooming and Core Transport

The three basic functions of edge-grooming, hub-grooming, and core cross-connection are in a sense fundamental specializations that are bound to emerge under almost any technology. SONET recognizes it with B-DCS and W-DCS generic elements. IP (and ATM) data

networking reflects the specializations of edge and core routers and under MPLS (to follow) we find with label edge routers (LERs) and label switching routers (LSRs). In optical networking, similar distinctions between access and core optical switches are also emerging. But the concept of transport networking, the fundamental efficiency it offers, and the fact that it emerges in all other real-world contexts involving transportation is not completely accepted in telecommunications, perhaps because "disruptive" technologies come so frequently. A tendency is for major advances in switching or routing technology to be coupled with proposed network architectures that would, in our prior analogy, amount to landing the 747 every 100 km to add and drop a few passengers. ATM switches to handle every cell of OC-192s, terabit IP packet switches, and optical packet switching are examples. The general point is that it is fundamentally unnecessary and unadvantageous to ever handle every cell, packet, or call at every core switching node. Any device that *can* handle every transiting packet or cell, at a node will inevitably require more space, power, complexity, cost and maintenance than a corresponding transport node handling only containers, coupled with a smaller local access node. A well-groomed lightpath routed and protected in the optical transport layer is ultimately a far more cost-efficient and scalable means to pass data through a node than to inspect and reroute every packet. The fundamental architecture of access, grooming and transport will therefore always offer efficiency and cost advantages as long as there are any limits to router (or switch) speed, power, space, cost, or reliability.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

1.3 Broadband ISDN and Asynchronous Transfer Mode (ATM)

The Broadband Integrated Services Digital Network (BISDN) was developed to support demand-switched, semipermanent, and permanent broadband connections for both point-to-point and point-to-multipoint applications. Channels operating at STS-3c and OC-12 were seen as the main bearer channels for BISDN. The foundation of BISDN is the *label-switched* routing of fixed-size Asynchronous Transfer Mode (ATM) cells using SONET/SDH transport.

BISDN was a precursor to the Internet in its intent, but IP-based applications and networking have supplanted it. The main legacy of BISDN / ATM, however, is the proven concepts of virtual circuits and label switching, which are now the basis of MPLS for IP traffic. BISDN recognized that compared to operating several dedicated networks, network service integration can provide advantages in terms of economics, implementation, operation, and maintenance. While dedicated networks require several distinct subscriber access lines, BISDN access can be based on a single optical fiber. A variety of interactive broadband services can be supported with BISDN: broadband video telephony; corporate videoconferencing; video surveillance; high-speed digital transport of medical images, artwork, and advertising; high-speed clear-channel data transmission; file transfer; high-resolution fax; color fax; video retrieval; TV distribution; LAN interconnection, etc.

Instead of reserving time slots as in STM networks, information in ATM is packetized and placed in short 53-byte cells that are multiplexed and transmitted asynchronously on the transmission medium. The 53 bytes of an ATM cell consist of a 48-byte information field and a 5-byte header. Two of the fields defined in the header are the *Virtual Path Identifier* (VPI) and the *Virtual Circuit Identifier* (VCI). VPI is a 12-bit field and the VCI is a 16-bit field that together define the routing information of a cell. As with any other packet-switching network, routing of cells is performed at every node for each arriving cell. ATM can support a variety of services (e.g., telephone, image, video and data), with a guaranteed *Quality of Service* (QoS). A *Virtual Circuit Connection* (VCC) in ATM is analogous to a virtual circuit in data networks, such as an X.25 or a frame relay logical connection. The VCC is the basic unit of switching in ATM networks. For ATM, a second tier of labeling defines a virtual path. A *Virtual Path Connection* (VPC) is a bundle of VCCs that have the same endpoints, e.g., switching systems, LAN gateways, etc. Thus, cells flowing over all of the VCCs in a single VPC may be switched together. Because VP switching is inherently more efficient than VC switching, it is advantageous to switch a cell in a VP. This Virtual Path concept was developed in response to a trend in high-speed networking in which the control cost of the network is becoming increasingly high in proportion to the overall network costs. The Virtual Path technique helps reduce the control cost by grouping connections sharing common paths through the network into a single unit. Network management actions can then be applied to a small number of groups of connections instead of a large number of individual connections.

For more detail on ATM, readers are referred to [\[Onvu94\]](#) and the first five chapters of [\[WuNo97\]](#). The last two chapters of [\[WuNo97\]](#) describe ATM VP-based protection switching mechanisms, although without an emphasis on network design. We return to ATM VP-based protection switching mechanisms as the starting point in [Chapter 7](#), which develops network design for controlled oversubscription of capacity upon restoration in ATM or MPLS networks.

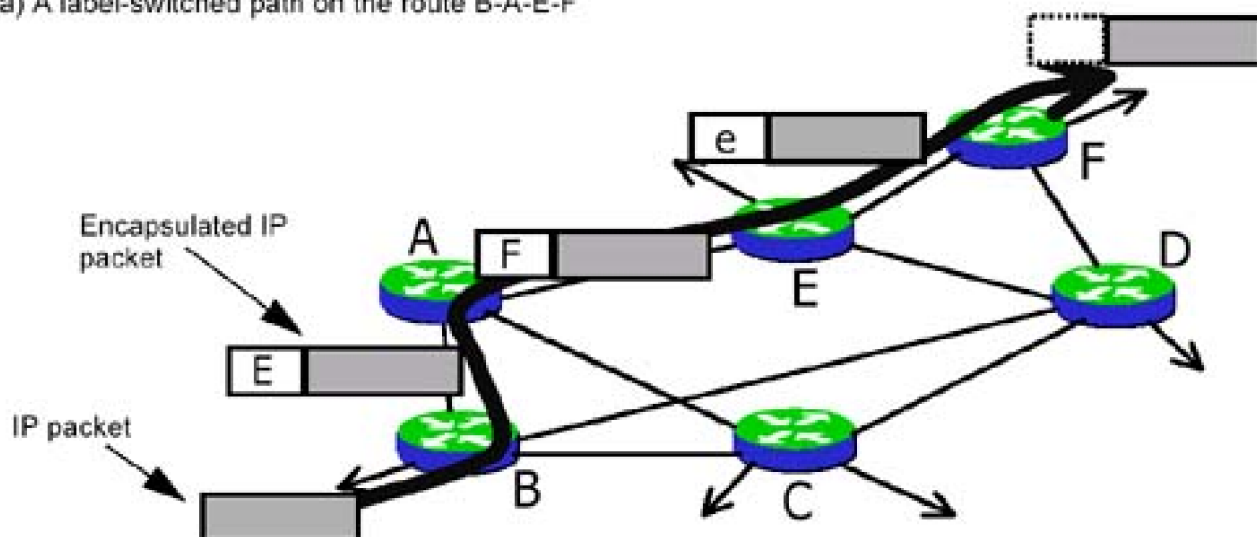
1.4 Concept of Label-Switching: The Basis of ATM and MPLS

ATM and multi-protocol label switching (MPLS) are both based on "label-switching." In ATM, fixed-size cells arrive in one port of an ATM switch and the label they bear in the cell overhead is used to index a routing table at the node that indicates the outgoing port and new label number to be assigned to the cell. In MPLS, variable-length IP packets arriving at a label-switching router (LSR) are similarly redirected based on incoming port and label, to a next-hop outgoing port and new label. In both schemes, a connection setup protocol creates the sequence of {out-port, next-label} entries along a desired path so that a logical circuit is established between any desired origin node (and application or user) and destination host (and application). ATM uses the virtual circuit (VC) construct to connect individual application-level connections. MPLS (over IP) uses logical port numbering schemes to distinguish between packets for different applications running on the same host or sharing the same label-switched path.

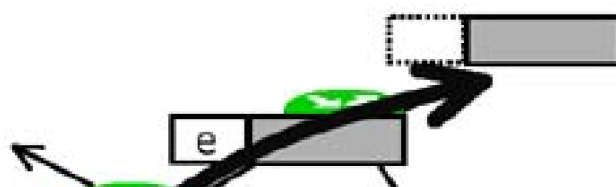
The concept of label-switching is illustrated in [Figure 1-13](#) where two label-switched paths are shown. In [Figure 1-13\(a\)](#) node B receives an unlabeled packet (or cell) at the entry to the path with an indication that its destination is node F. A separate path setup process is assumed to have previously laid down a set of label-swapping rules at each node involved. If the intention is to create the path shown in [Figure 1-13\(a\)](#), then the label swapping entries at each node to create this path are shown in the top four rows of [table 1-5](#). The corresponding entries to create the path in [Figure 1-13\(b\)](#) are in the second half of the table. The example defines two different paths that could be established between nodes B and F. The example packet initially arrives at node B but does not bear a label at that point. We use the label field in the table to indicate, however, that its destination is known (node F). At node B, the routing table entry says in effect "if anything comes in going to node F, send it to node A with a label E." When node B receives a packet on the input port from node A, on label E, the table tells it to send the packet to the output port connected to node F, on the new label "e". In this context "e" is just a code for path termination (i.e., decapsulate the packet and terminate it, or forward it on under other normal routing means). The label-switched path is then completed. Note in these examples that the "new label" assignments are identically the name of the next node to which the packet is forwarded. This particular labeling scheme can be thought of as each node saying to the next: "When you get this packet from me, pass it on to *this* next node for me, and look up who they should pass it on to following that as the new label."

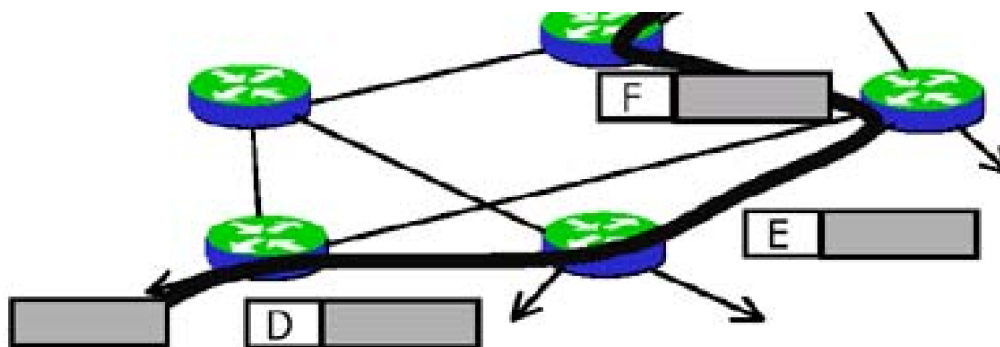
Figure 1-13. Examples of label-switched paths.

(a) A label-switched path on the route B-A-E-F



(b) A label-switched path on the route B-C-D-E-F





Labelling schemes can alternatively be completely arbitrary. As long as the sequence of relays represented by {in-port, label} to {out-port, new-label} at each node is written coherently, the label values themselves can be completely arbitrary tags. The special significance of the labeling scheme used for the example will become more apparent later when we begin to consider actual SONET/SDH timeslots or DWDM circuit establishment under generalized MPLS (GMPLS). In those cases the label sequence becomes a specification of literally what time slots, wavelengths or fibers we want cross-connected to realize a hard physical circuit, rather than strictly just a packet or cell relay function such as a label switched path.

Table 1-5. Label swapping table entries to effect the paths in Figure 1-13

node	role	in port	label	out port	new label
B	entry	n/a	(dest = F)	to A	E
A	forwarding	from B	E	to E	F
E	forwarding	from A	F	to F	e
F	egress	from E	e (=egress)	n/a	n/a
B	entry	n/a	(dest = F)	to C	D
C	forwarding	from A	C	to D	E
D	forwarding	from C	D	to E	F
E	forwarding	from D	E	to F	e
F	egress	from E	e	n/a	n/a

Another important point to observe is that from the point of view of the entry node B in the first example, the label E on port A is completely equivalent to a single tag for "the route to node F." Since the mapping between labels is constant at each node, the complete path is determined by the initial label (and port) value. Node B need not know any other details; anything it puts on port A with label E will egress at node F. This illustrates the sense in which a label-switched path truly establishes a dedicated circuit or tunnel through the network between the respective end-node pairs. In fact in the example with two different paths between nodes B and F, node B could consider that it has two separate permanent circuits that it can use to send traffic to node F: route 1 = {port A, label E}, route 2 = {port C, label D}. A pair of such virtual circuits or label-switched paths can be used in practice for load balancing over the two routes, or as a form of protection switching.

Label-switching was adopted for both ATM and MPLS out of a fundamental need for a circuit-like logical construct for many networking purposes. Pure datagram flows cannot be controlled enough to support QoS assurances and effective and fast schemes for restoration are greatly enabled by the manipulation of either physical or logical circuit-like quantities, rather than redirection of every single packet or cell through conventional IP routing tables. The reason failure recovery is faster within a circuit-oriented paradigm is ultimately the same reason that basic delay, throughput and loss rates of cell or packet transport are improved by switching as opposed to routing.

Routing involves determining a next-hop decision at every node based on the absolute global destination (and sometimes source and packet type) information of the packet in conjunction with globally-determined routing tables at each node. This is fundamentally a more time-consuming and unreliable process than label-switching. A conventional IP router has to perform a maximum length address-matching algorithm on every packet and rely on continual topology updates to ensure a correct routing table entry for every possible destination IP address in the administrative area. The maximal length aspect of the address-matching refers to the fact that a packet's destination address may produce no *exact* match in the routing table. In such cases the packet is forwarded toward the subnetwork with the maximal

partial address match.

In label *switching*, however, the next hop decision is made with entirely local information established previously in an explicit connection setup process. The local information is the label swapping tables in an LSR, or simply the physical connection state between input and output timeslots or wavelengths in a cross-connect. In a physical fiber or metallic switch, or in a timeslot-changing switch, or a wavelength cross-connect, the switching information (i.e., the output and next timeslot or wavelength) is stored in the physical connection state. In an LSR or ATM switch, the forwarding port and next-label are stored in tables that are typically much smaller than a full IP routing table, and operate on the basis of an exact label match which is fundamentally much faster, autonomous, and reliable than IP datagram routing.

1.4.1 Multi-Protocol Label Switching (MPLS)

In MPLS, a relatively short "label" replaces the full interpretation of an IP packet's header as necessary under conventional forwarding. In traditional IP forwarding, packets are processed on the basis of the global destination address at every router along the path. This is also known as hop-by-hop routing. In contrast, MPLS encapsulates IP packets with a label when the packet first arrives into an MPLS routing domain. The MPLS edge router will look at the information in the IP header and select an appropriate label for it. The label selected by the MPLS edge router can be based on QoS and explicit routing considerations, not just the destination address in the IP header. This is in contrast to conventional IP routing, where the destination address will determine the next hop for the packet. All subsequent routers in the MPLS domain will then forward the packet based on the label, not the IP header information. Then, as the labeled packets exit the MPLS domain, the edge router through which it is leaving will remove the label. The value of the label usually changes at each LSR in the path as the incoming label is looked up and swapped for another. Thus labels only have meaning between neighboring nodes. With this, MPLS can be used to forward data between neighboring nodes even if no other nodes on the network support MPLS.

MPLS uses two categories of label-based routers: label edge routers (LERs) and label switching routers (LSRs). An LSR is used in the core of an MPLS network and participates in the establishment of LSPs and high-speed switching of flows over existing LSPs. Notably many LSRs are implemented with high capacity ATM label-switching switch hardware as their core with a conventional router added to participate in the LSP establishment protocol and to perform routing functions for the residual traffic of a normal IP datagram nature (i.e., packet traffic that does not constitute significant flows per se to warrant establishment of an LSP). The LER supports access to the MPLS core network. Edge routers are classified as either ingress or egress. An MPLS ingress router performs the initial routing lookup and LSP assignment for launching the packets into the fabric of logical pipes through the MPLS core. The egress LER removes labels and performs a further routing table lookup to continue the packet on its way under conventional routing. LERs can have many types of access interface (frame relay, ATM, Ethernet for example).

MPLS allows a packet flow to be fully "routed" only once, at the entry to an IP network, and thereafter label-switched all the way to its egress router. Aside from the fact that the packet flows remain asynchronous and statistically multiplexed, in all logical regards, once the IP packets for that destination are encapsulated with a label and put in the appropriate outgoing queue from the egress node, it is like they have been dropped into a dedicated pipe or tunnel direct to the destination. The train of switching relationships written in the label-swapping tables en route creates a hard-wired sequence of relays that is fully amenable to hardware implementation at each node, and referred to as a *Label Switched Path* (LSP). This improves router network throughput and delay performance compared to routing every packet at every node en route based on its ultimate destination IP address.

In assigning IP packets to LSPs, an LER makes use of the concept of forwarding equivalence. A Forwarding Equivalence Class (FEC) is a set of IP destination addresses that, from the standpoint of the given entry node all share the same local routing decision. In conventional IP forwarding, each router needs to maintain FEC tables to indicate for each output port the set of IP destination addresses that should be sent out that port. In MPLS the FEC tables need be kept only at LERs and record for each IP address what LSP to put it on (i.e., what initial label and outgoing port to assign it to). FECs at the LERs are based on class of service requirements and on IP address. The table at the LER that lists the initial label, and outgoing port for each FEC is called its label information base (LIB).

For more detailed background on MPLS the book by Davie and Reckhter [\[DaRe00\]](#), and references therein, are suggested. The relevance here is twofold. In [Chapter 2](#) we see how MPLS is extended into Generalized MPLS (GMPLS), with applications in the automatic provisioning of service paths through a transport network at any level such as DS-1 through to whole lightpaths. As with ATM, MPLS is also a transport technique that involves circuit-oriented logical constructs for the transport of stochastic packet flows, and hence is also amenable to the controlled oversubscription design strategy of [Chapter 7](#).

[\[Team LiB \]](#)

1.5 Network Planning Aspects of Transport Networks

1.5.1 Modularity and Economy-of-Scale

SONET provides an example of a hierarchy of standard rates and formats on which multiplexing and transmission system products can be based. Some industry experts expect that an SDH-like progression (4, 16, 64...) of wavelengths per fiber may also arise for DWDM transmission. The adoption of a discrete set of modular capacities aids both users and system designers. It allows specific technology choices (such as frequency spacing, carrier generation, EDFA noise, gain, bandwidth, etc.) to be combined and optimized to realize a product offering at each price-performance point that the market needs. Some vendors may specialize in the market for 16-l systems while another may use quite different technology to specialize in, say, 768-l systems. As a result, the actual installed capacity in a SONET (or an optical network as well) is inevitably *modular* in nature. The relevance to network design is that although we may often compare design approaches on the basis of total unit-capacity requirements, we have to keep in mind that the real capacities and costs will be stair-step functions of cost versus capacity. Moreover, cost is rarely in linear dependence on the capacity; so there is an *economy of scale* effect. In detailed studies it can be important to consider the modular and nonlinear cost structure of the actual equipment to be specified. As an example, consider a bidirectional 48-l DWDM system. The cost structure will reflect all the following physical items.^[3]

^[3] This discussion is adapted for use here from [DoGr00](#).

- a. One fiber pair and pro-rated cost of right-of-way acquisition, duct and cable, installation, and repeater/amplifier housing costs
- b. 48 electrical (transmit) channel interfaces
- c. Generation and modulation of 48 optical carriers
- d. Optical WDM multiplexor
- e. In-line optical amplifiers with bandwidth and power capabilities suitable for 48 wavelengths, every 60 to 100 km typically
- f. An average distance-amortized cost for 48 regenerators, every 300 km, say
- g. Optical WDM demultiplexor
- h. 48 electrical (receive) channel interfaces
- i. Redundant common power, maintenance processor, rack, cabling, and equipment bay installation costs

Items (a), (d), (e), (g), (i), and a large part of (f), are one-time costs required for the system's existence, even if just one channel is operated. Such "get started" cost items are traditionally called the "common equipment" in telecommunications and are a major contributor to the nonlinear cost structure. To turn up the first channel, all the common equipment must be present. In principle, all subsequent electrical and optical per-channel circuit packs can then be provisioned on a one-by-one basis as needed, but if growth is high or on-demand path establishment is desired, then all of these may also be preprovisioned. In the limit of full preprovisioning there is a large step in cost to establish the system, but no further cost depending on how many channels are actually in use. This is the modularity aspect. The economy of scale aspect is that, in this example, a 192-l system may be only two or three times the cost of the 48-l system (not four times).

If the per-channel costs are actually equipped only on an as-needed basis, then we actually have two levels of capacity-cost modularity. Say for example that for current working requirements a given facility requires 37 wavelength channels. Then the common equipment for a standard 48-l system may be required, but only 37 of the per-channel costs need actually be incurred. Thus the cost structure for transport capacity actually has a least two scales on which it is modular. A third sense in which capacity cost is extremely modular is if we recognize

the "get-started" cost of having the facility right-of-way in the first place. This is usually only a factor in considering topology design itself, however.

There are several cases where modeling a single scale of modularity can be justified, however, corresponding to the full capacity of each basic module size. First, if the growth rate is high and the cost of dispatching maintenance crews to populate new channel cards one-by-one is also high, then it may make more sense to fully equip the system when installed. Secondly, to support truly dynamic path provisioning the systems may be *fully preprovisioned* when installed. Additionally, in any case where the average system utilization is high following some planning design study, then the "fully equipped" model also tends to be justified in its own right. Another way to limit the complexity to a single modularity scale while recognizing only partial channel equipping is to approximate the average fill factor of transmission systems and lump the total per-channel costs of this average fill level into the module cost.

A notation used to represent economy of scale is "*N times Y times*." For instance 3x2x means that a tripling of capacity results in a doubling of cost. "3x2x" is fairly characteristic of mature SONET transmission technology over the OC-24 to OC-192 range. Similar or even stronger economy of scale effects may characterize mature DWDM transmission systems.

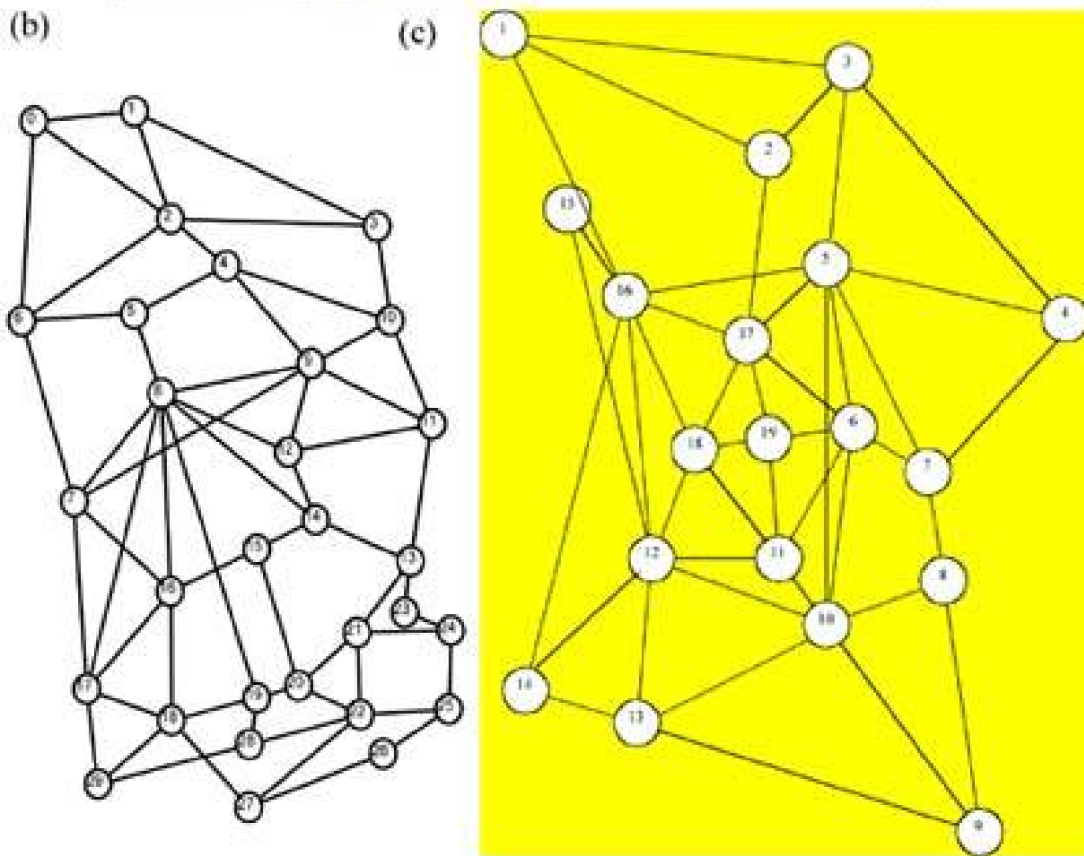
1.5.2 Fiber Route Structures

In transport network planning we refer to the fiber route structure on which physical facilities can be built as the network graph. This is the actual set of right-of-ways on which permission to lay fiber cables is in hand. Several recently built networks are based on rights-of-way obtained from a cooperating, co-owned or defunct power, gas, or railway utility and so inherit some of the topological properties of those networks. Typical transport network graphs tend to have an irregular mesh interconnection pattern with an average number of geographically diverse rights-of-ways at junction sites that varies from two to at most about seven at any individual node. This measure of nodal route-diversity is called the span-degree or just the "degree" of a node. The corresponding average over all nodes is also referred to as the network average nodal degree, \bar{d} .

In some North American networks \bar{d} may be as low as 2.3, indicating a quite sparse topology. A degree of two at every node is the minimum possible in any network that has a fully restorable topology. In contrast some European networks have \bar{d} as high as 4.5. [Figure 1-14](#) allows visualization of these two extremes, using examples from published data on real networks. Note, however, that even the most richly connected transport graphs are quite sparse relative to truly random graphs of full mesh graphs. Transport network graphs tend to look neither like fully connected networks nor like random graphs where any node may with equal probability be connected to any other. In addition to the characteristic \bar{d} in the range of 2 - 4.5 real networks tend to show a high "locality" in that nodes tend to be connected to other nodes that are near them in the plane. Related to this locality characteristic is planarity. Typical transport graphs are often strictly planar or have only a few non-planar spans. (Planarity is treated further in [Chapter 4](#).) Metropolitan area networks in both North America and Europe tend to be similar to the more richly connected European national backbone example in [Figure 1-14\(b\)](#), whereas long-haul networks in North America tend to be the most sparse. In contrast to the service layer networks realized over the transport network, the transport graph is typically much sparser. Sometimes statements about nodal degree cause confusion because the reference is to different layers. For instance, viewed at the STS-3c connectivity layer, a given location might logically have a degree of 160. But the same node could be a degree-2 site in the facilities graph. Unless explicitly stated otherwise in this book, the default understanding about any reference to the network graph or to a nodal degree, we are referring to the physical layer facilities graph.

Figure 1-14. The range of fiber network topologies: (a) a sparse North American long-haul route-structure ($\bar{d} \approx 2.2$), (b) a more richly connected European fiber backbone route structure ($\bar{d} \approx 4.4$), (c) another European regional network ($\bar{d} \approx 4.1$).





1.5.3 Network Redundancy

Another important characteristic of transport network and a measure of architectural efficiency for a survivable transport network is its *redundancy*. Redundancy is always a consideration in conjunction with survivability design because it indicates how "lean" or efficient the design is in its use of spare capacity. Redundancy is in a broad sense the price one has to pay to achieve a target level of restorability or other measures of survivability. *Capacity efficiency* is defined as the reciprocal of the redundancy. It is ultimately cost that matters and this is not always directly proportional to capacity-efficiency, however. High redundancy in ring-based metro networks does not always mean they are economically inferior to a less redundant mesh network design, for example, because nodal costs may be lower with ring ADMs than with mesh cross-connects. But what is always true is that to serve and protect a common base of demands, where all other factors are the same, a more redundant design requires more capacity and hence does cost more. Especially in long-haul networks, cost does correlate with capacity.

Often in research or basic comparative planning studies, exact cost data is also not available, so redundancy is used as a surrogate for cost in comparing alternatives and as an intrinsic measure of network design efficiency. It should also be noted that although redundancy is usually computed from capacity quantities, each unit of capacity on a span has related nodal equipment at its ends that is either a per-channel cost to start with or is a common cost that can be pro-rated to each channel of capacity. So redundancy is perhaps the single most generic surrogate that we have for actual cost effectiveness of an overall survivable network design.

The *logical redundancy* of a network is defined as the pure ratio of spare to working channel counts and does not take any distance effects into account.

Equation 1.2

$$\mathfrak{R} \equiv \sum_{m \in S} s_m / \sum_{m \in S} w_m$$

where S is the set of spans in the network, w_m is the number of units of working capacity on span m and s_m is the corresponding number of spare capacity units on span m . Spare capacity may be the protection fibers of APS or ring systems or designed-in spare channels for restoration in a mesh network. Note that the definition of redundancy that we use differs from the other possible definition which is the ratio of spare to *total* capacity present, i.e., spare / (working + spare). It is sometimes important to clarify which definition of redundancy is being used. The *geographical redundancy* is of the same form as [Equation 1.2](#) but each s_m , w_m term is weighted by a coefficient representing either the distance of a span or the cost per channel on the span.

Both of these redundancy measures are somewhat idealized in that they allow working and spare capacities to be present in any assumed quantity. But as discussed, the capacities of real transmission systems are modular. In such cases the *modular redundancy* is defined as:

Equation 1.3

$$\mathfrak{R}_{mod} \equiv \sum_{m \in S} (C_m - w_m) / \sum_{m \in S} w_m$$

where C_m is the total installed (modular) capacity on span m . In some contexts the non-working capacity of modules on a span is sometimes viewed as containing two sub-parts: $(C_m - w_m) = s_m + o_m$ where o_m is called the *provisioning overhead* due to modularity. The *modular geographic redundancy* follows the same pattern as above: it is given by [Equation 1.3](#) respectively, but every capacity term is again weighted with the distance or cost of span m .

Note that when using redundancy as a figure of merit for comparison of survivability techniques, it is only meaningful to compare alternatives in which the working demands are routed in an identical, or at least capacity-equivalent, way. In many problems this is the case because demands follow shortest paths in all alternatives and the only comparative questions are about the spare capacity design to protect such working flows. (These are later called "non-joint" design problems and redundancy is always a meaningful comparative measure.) In cases where the working routes and working capacity may be manipulated or altered as well as spare capacity as part of the overall design strategy, the simple redundancy measures above can be misleading. The reason is that if one increases working route lengths, more total capacity may be required, but it will appear that redundancy per se is lower. For this reason the *standard redundancy* of a network is also defined as:

Equation 1.4

$$\mathfrak{R}^* = \left\{ \sum_{m \in S} (w_m + s_m) - \sum_{m \in S} w_m^* \right\} / \sum_{m \in S} w_m^*$$

where $\sum w_m^*$ is the total working capacity needed to support *shortest-path routing* of all demands (i.e., the minimum possible working capacity of a network) and $\sum w_m$ is the actual amount of working capacity used to support working paths in the network. Any excess of working capacity above the minimum for shortest path routing is thus equivalent to additional spare capacity used in the design. This allows meaningful comparison between more general design strategies (later called "jointly" optimized designs). Under standard redundancy measures any two designs serving the same demands on the same graph can be fairly compared.

1.5.4 Shared-Risk Entities and Fault Multiplication

Generally, network operators who own and operate their own physical facilities ("facilities-based operators") pay considerable attention to the physical separation or "diversity" of cable routings. Particular efforts are made to ensure that important backbone fiber cables are routed over physically disjoint ducts, pole lines, or buried trenches, or at least on opposite sides of the same streets. The point is to mitigate the impact that any one physical structure failure might have in terms of the number of transmission paths it brings down. It is tempting to think that this is a new concern only associated with recent emphasis on automatic network-level rerouting for survivability. Physical cable diversity has, however, been a general historical concern. In some senses diversity was even more important without an embedded network restoration mechanism because the outage would last as long as the cable repairs. Care would be taken then that important services, backbone transmission paths, and key trunk groups would be split and routed over diverse physical cable or microwave routes. Telephone central office buildings typically have two to four separate underground cable vault entry points on opposite sides or all compass points of the building footprint, precisely for diversity.

In more recent practice, where spare capacity is explicitly designed into the network for survivability, it is important to take the existence of any "shared risk entities" into account in the capacity design. A shared risk group (SRG) or shared risk link group (SRLG) defines a set of transport channels or paths that have something in common: they all fail together if a certain single item of physical equipment fails [StCh01] [LiYa02] [SeYa01]. The most common ways in which multiple different paths share a risk of simultaneous disruption from a single common physical failure are:

1. The paths share the same cable (or fiber if on WDM carriers).
2. The paths traverse separate cables that share a common duct or other structure such as a bridge crossing.
3. The paths are routed through a common node.

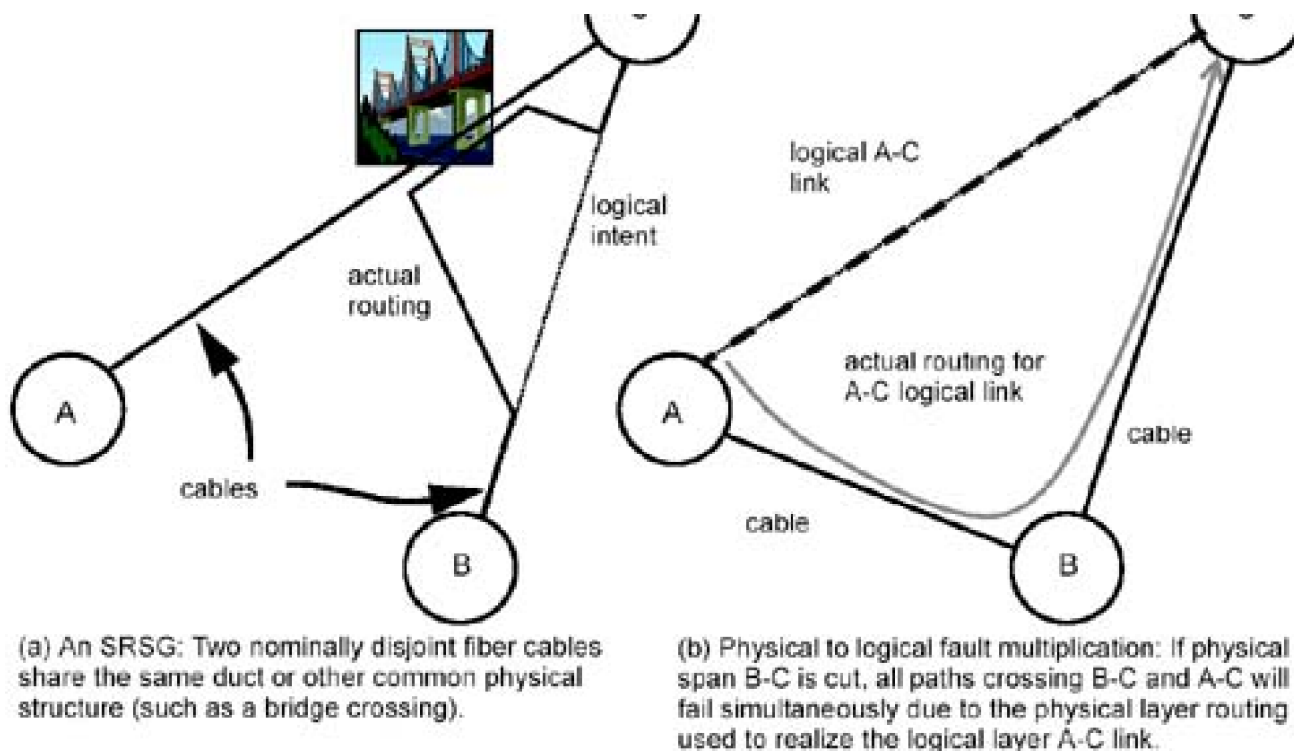
That paths should share the same cables extensively throughout the network is inevitable and expected. But in certain types of path protection, the set of end-to-end paths that share each cable must be grouped together for failure planning purposes. Such groups of paths are called *shared risk link groups* (SRLGs). The importance to path restoration schemes is that each SRLG that exists defines a specific set of end-to-end path failures that must be simultaneously restored. More generally any failure of a cable, a duct, or a node defines a set of network paths that are simultaneously disrupted. Each such physical element failure defines another SRLG. The SRLG concept is thus the most encompassing category of shared risk failure scenarios. In practice, because single cable cuts are still the most frequent failure event, the set of end-to-end paths affected by single cable cuts is also called the "default set of SRLGs."

Another class of restoration schemes are span-oriented, however, and do not need to consider or know the end-to-end details of all the paths affected, because these schemes recover from the failure by replacement routing directly between the end nodes of the failed span (i.e., the nodes adjacent to a failed span or node). In this context, isolated but complete failure of an individual cable span connecting adjacent nodes is the basic single-fault model. The aspect of shared risk that is of concern under span-oriented restoration moves up from planning oriented on individual links within paths to planning that is oriented to shared risk between cables that define the spans between cross-connect nodes. We refer to these as *shared risk span groups* (SRSGs). Whereas SRLGs are strictly quite numerous (because each cable segment defines at least one SRLG), there may be many fewer SRSGs in a network because each SRSG represents an unintended lapse in physical layer cable or duct diversity. In other words, an SRSG is a situation where entire cable spans that are at least notionally separate are actually routed together over some common structure. This is different than the routine and numerous forms of shared risk between all end-to-end paths that traverse the same cable together. The classic case of an SRSG is that of a bridge crossing near a node that is common to two spans. All of the transmission channels in both spans share a common risk of failure on the bridge crossing, although to network planners the logical view may be of two separate cables, for which survivability planning would normally consider only as separate individual failure scenarios. This type of physical layer diversity lapse may be relatively rare, but the importance of an SRSG in network planning is that it allows for the simultaneous failure of what are otherwise nominally independent spans of the transport graph. In other words, an SRSG sets up a situation where one physical failure can multiply into two or more logical span failures.

But certain fault multiplication effects can also arise without involvement of an SRSG. [Figure 1-15](#) illustrates a true SRSG on the left, and a nodal bypass arrangement, often referred to as express routes or glass-throughs, on the right. The nodal bypass sets up a particular type logical dual-failure situation. In the nodal bypass one or more paths bypasses node B but follows the same ducts as other channels on spans A-B and B-C. Such bypass arrangements are usually established when the demand (in the example, between nodes A and C) warrants its own dedicated fiber(s) but a separate geographical route is not available. In this case the A-C express fibers can be well enough loaded that they can be dedicated to the A-C direct logical span, saving OEO termination costs at the intermediate node B. But this sets up certain logical dual failure situations. For instance, if duct A-B is cut, spans A-B and A-C appear to fail simultaneously.

Figure 1-15. Shared risk span group in the physical layer and closely related concept of physical-to-logical fault escalation.





Perhaps because it might be considered sensitive data, or because it is hard for the network operator to determine, it is difficult to obtain network models that are complete with data on SRLGs. Getting rid of span SRLGs has been a historical preoccupation of network planners regardless of the survivability strategy because SRLGs inherently complicate and undermine any scheme for protection or restoration, even manual patch recovery or 1+1 APS. It is an embarrassment when it is found out (now and then) that some ultra-important service such as a Defense Department backbone link, planned on a diverse 1+1 basis, in fact shares the same bridge crossing (for example) at some unexpected locale in the middle of the country. An interesting anecdote is about a pair of "mission critical" telemetry links used to monitor high pressure gas pipelines in the Canadian North. For 3,000 km these 1+1 redundant signal paths were indeed physically diverse and created at considerable expense to establish and audit the path realization. The story goes, however, that after some years of operation it was discovered that 100 meters from the company operations center in a city 3,000 km to the south both signal feeds crossed a bridge together. So the problem of SRLGs is not new, although the term is recent. The issue has existed for decades and has traditionally haunted special-services efforts to establish 1+1 APS arrangements with confidence in the intended diversity in the physical layer. Even when operators lease carrier services from other service providers to try and assure diversity, SRLGs can manifest themselves. An outcome from Hurricane Hugo in the US was the apparent discovery that up to five different network operators had transport spans crossing one particular, seemingly unimportant, small bridge in the rural southwest. This SRLG was discovered when the bridge was washed out by the flood surge!

The effect of SRSGs and the logical dual failure scenarios that arise from bypass are both studied in further in [Chapter 8](#). In [Chapter 3](#) the related but more general concept of *dependent edges* is discussed in terms of its implications for the choice of level—physical, logical, or services layer—at which various survivability principles can be applied.

1.5.5 Demand Patterns

We previously described how various service layer traffic types are combined into standard "demand units" for delivery via the transport network. Thus, different service layer architectures and traffic handling strategies influence the pattern of demand requirements seen by the transport network. When many nodes have a strong demand flow to (from is always implied as well) one or two specific nodes the pattern is described as a "hub-like" demand pattern. In a metropolitan network this propensity can arise when telephony traffic from many sites is trunked to just one or two major centers for switching. In addition there may be only one or two toll-connecting offices to access the long-distance switching network from each metro area. Within the metro area network it therefore appears that there is a strong demand to these "hub nodes." The establishment of regional hubs for grooming also causes a hub-like demand pattern to arise from bringing many DS-3 or STS-1 access connections back to a W-DCS for grooming into well-filled OC-48s (for example) for long haul transport. In the limiting case, this leads to the notion of a perfectly hubbed demand pattern in which all nodes of a region exchange demands only with the hub site(s).

As illustrated in [Figure 1-12](#), such hub sites represent each region on the backbone inter-regional transport network. At this level the regional hub sites are typically exchanging demand at the OC-48 or OC-192 level in a much more uniform random mesh-like pattern. In practice, there may be detailed historical, demographic and economic explanations for the exact amount of demand exchanged between each regional hub node, but the overall effect is that of an essentially random distribution of demands on each node pair. The random mesh type of demand pattern may also arise in dense metropolitan area networks where the community of interest between all nodes is essentially almost equal and a hubbing architecture is not used. Callers and other forms of communication from each node are as likely to go to one node as any other in the rest of the network.

A historically recognized effect for long haul demand is referred to as a "gravity-like" inverse-distance effect on demand intensities. The intensity of voice communication in particular depends proportionately on the size of the two centers and (at least historically) inversely on the distance between population centers. There is a classic anecdote of the Severn Bridge in England that illustrates this tendency. The Severn Bridge was a major construction effort to span a large estuary that previously separated two cities. The effect of the bridge was to reduce the travel distance between the two communities from hundreds of miles to only a few. Telephone calling volumes between these centres went up markedly after the bridge was built, giving experimental evidence for the inverse-distance effect when all other factors were equal. When the bridge brought the two nodes closer together, their mutual attraction (and hence mutual traffic) rose more than proportionately.

Of course, any real situation will reflect the individual historical and demographic circumstances affecting demand. In general, there will always be some blend of uniform, hubbing, and gravity type effects in a real demand pattern seen at the transport layer. Recognition of these three constituent pattern types can allow us to numerically model and systematically vary demand patterns for a variety of purposes in network planning or research studies.

Numerical Modeling of Demand Patterns

Often network planning or research studies have to be conducted without the luxury of a known or reliably forecast demand matrix. Obtaining accurate forecasts of the real demand is a whole discipline of its own. In other contexts, however, one is conducting comparative network planning or research studies and needs only a representative demand model on which to base comparative design studies. While there is an infinite number of demand patterns that could be tested, the practical researcher or planner has to use only a few test-case demand patterns to try to reach generally true comparative conclusions about alternatives. A variety of different test-case demand patterns can be made up from the following models:

1. Pure logical mesh with equal number of demands on all node pairs
2. Random distributed demand: uniform random number of demands on each pair
3. Hub-like demand pattern
4. Dual hub demand pattern
5. Mutual attraction demand models
6. "Gravity" demand (mutual attraction with inverse distance dependency)
7. Weighted superpositions of the above component patterns

The first two models are self explanatory. The hub and dual hub patterns often characterize metro or regional area networks involving one or two main hub sites. In general, the demand between a hub and a non-hub node may also be inversely proportional to the distance between them. All other node pairs (i.e., between two non-hub nodes) exchange either a constant or uniform random number of demand units. A numerical model that can be used is:

Equation 1.5

$$d_{i, hub} = int(d_0 - round[(I_{i, hub})/I_0])$$

Equation 1.6

$$d_{i, non-hub} = d_0 l_i$$

where d_0 and l_0 are demand and distance scale constants, respectively, and $d_{i, hub}$ is the distance from node i to the respective hub, $d_{i, hub}$ is the (integer) number of demands generated. The second part of the model generates a constant mutual demand between all other non-hub nodes.

A principle that is believed to underlie real traffic intensities at long-distance scales is the concept of mutual attraction, independent of distance. For example, New York City and Los Angeles are two of the most important centers in the U.S. New York City also exchanges traffic with Ottawa, but it is lower because of Ottawa's lower population and economic status compared to Los Angeles. Population data can be one measure of this notion of importance in the real world, but to generate test-case demand patterns with the same overall characteristics, "importance factors" can be assigned to nodes at random. Another approach is to view nodal degree as a surrogate for the presumed demographic importance. The argument is that large centers will tend to have higher nodal degree. In either case, once the "importance factors" l_i are assigned to each node i , the mutual attraction demand model is:

Equation 1.7

$$d_{i,j} = int((I_i \cdot I_j) \cdot d_0)$$

The pure mutual attraction model is independent of distance between centers. The "gravity" demand model adds an inverse distance dependency, partially offsetting the mutual importance effect. For example, extending our prior example to include Paris, we might agree that Paris, New York and Los Angeles are all of the same "importance" class, but "distance" (including geographical, cultural, time-zone offset, etc.) tends to attenuate the Los Angeles–Paris demand relative to the Los Angeles–New York demand. The fully general "gravity" model produces a number of demands for each (i,j) node pair that responds to both importance and distance as follows:

Equation 1.8

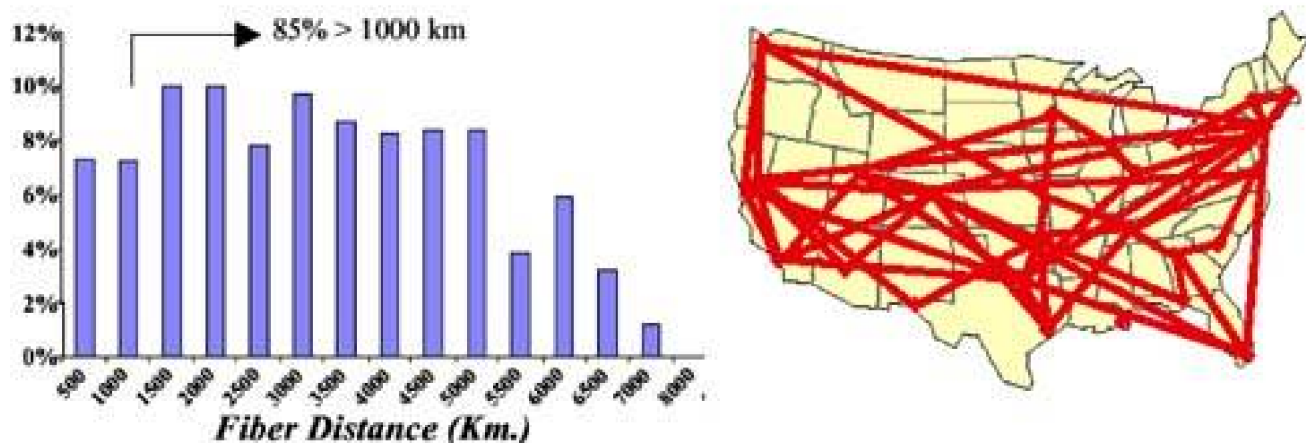
$$d_{i,j} = int\left(\frac{I_i \cdot I_j}{(L_{i,j})^a} \cdot d_0\right)$$

where a is an exponent modeling the inverse-distance effect and d_0 is a scaling constant set so that the individual demand quantities are in the range intended. Of course the name "gravity" suggests $a = 2$. But in practice where this method has been compared to actual demand patterns, a much closer to one (e.g., 1.1 to 1.3) have been typically found to be more accurate.

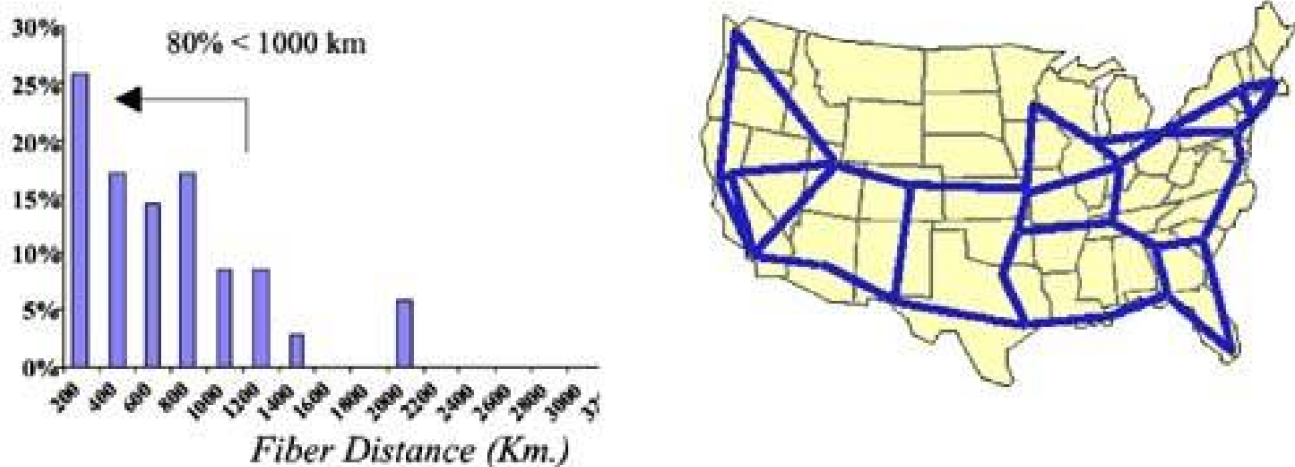
There is evidence that as IP data volumes outpace voice, demand in real backbone networks is tending even more toward distance independence (i.e., $a \rightarrow 0$). This may also be due in part to flat-rate (distance independent) voice calling plans. A 2001 study of North American long-haul demand evolution showed that pair-wise total demand between nodes was almost uncorrelated to geographical distance. This can be seen in [Figure 1-16\(a\)](#). In a sense, with the Internet, all points are now an insignificant "distance" apart from a social or commercial standpoint. This gives increasing justification for network planning studies to use a distance independent demand model.

Figure 1-16. Evolution of demand patterns to be almost independent of distance within a national network implying that most nodal flow is also transiting traffic (adapted from [[FrSo01](#)]).

(a) Logical network distances of demand flows



(b) Physical network span distances



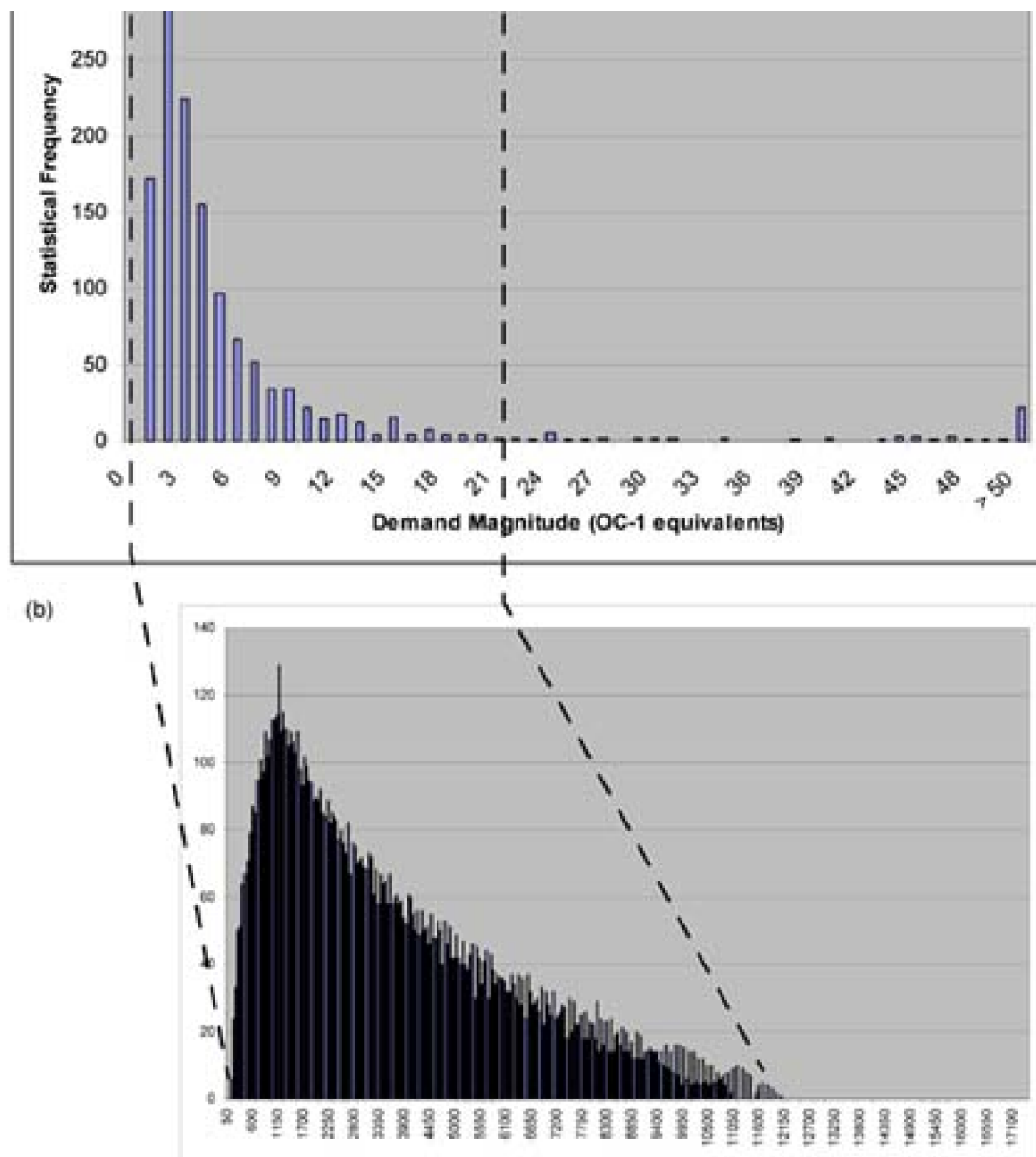
An implication of distance-independent demand is that most traffic at backbone network nodes tends to be transiting traffic. To illustrate, [Figure 1-16\(b\)](#) shows the distribution of physical span distances between adjacent (i.e., directly connected) nodes in the same network to which the data of [Figure 1-16\(a\)](#) pertain. Together [Figure 1-16\(a\)](#) and (b) imply that in the average backbone node, 70% of the flow at a node *transits* the site. On average only 30% originates (or terminates) at a node. (This is another compelling argument for the role of cross-connects in a transport layer to pass such flow through at low cost and high speed, without any electrical payload manipulation at transiting nodes.)

Note that distance-independence does not imply that all demands are also equal in their magnitude. [Figure 1-17\(a\)](#) illustrates this with data from an inter-exchange carrier network. The histogram shows the statistical frequency of various numbers of STS-1 equivalents of demand between node pairs. (Data is only plotted for those node pairs exchanging a non-zero demand quantity.) About 90% of all possible demand pairs exchange at least one STS-1. What the data show is that there tends to be three components to the overall demand model:

- A fraction of node pairs exchanging no transport-level demands.
- A large number of demand pairs exchanging demands that are distributed consistent with the pure attraction (product of importance factors) model.
- A small number of demand pairs exchanging relatively huge demand values lying far out on the tail of the overall distribution.

Figure 1-17. (a) Histogram of demand magnitudes in an inter-exchange carrier network, (b) Histogram of the product of uniformly distributed "importance" numbers for comparison.





Let us comment on each of these three components. First, for clarification, when a node pair exchanges no demands in the transport network, it does not mean that there is no traffic between these centres. It just means that the traffic they exchange is being routed via regional access networks through different transport hubs. At the other extreme is the set of super-demands that seem to be completely independent of the rest of the distribution. In this particular data there were 22 values above 50 STS-1s, with individual values of 89, 113 and so on, irregularly up to as high as 210 STS-1s. The number of such super-demands is under 10% of all node pairs and is distributed in no recognizable way in terms of the histogram, but in the full data they clearly correspond to the largest city pairs in the country. Although <10% in number, they contribute almost 50% of the total demand over all node pairs.

The middle grouping of the overall demand distribution (corresponding to the second bullet above) produce the most prominent characteristic of [Figure 1-17](#). This includes about 80% of all the possible demand pairs and amounts to about half the total demand. What we mean by the most prominent characteristic in [Figure 1-17](#) is the overall binomial-like shape in [Figure 1-17](#). This distribution is remarkably consistent with the mutual attraction model in which a uniform random "importance number" is assigned to each node and the product of importance values determines the demand for each node pair. To illustrate how characteristic the attraction model is of the actual data, [Figure 1-17\(b\)](#), shows the distribution of the products of uniform random importance values assigned to 50 nodes on the range [10...110].

For more discussion and useful data about demand patterns, and the breakdown of demand into voice and Internet transactions, etc., [\[MaCo03\]](#) is an excellent reference. It also provides several examples of European inter-city transport network graphs in particular.

1.6 Short and Long-Term Transport Network Planning Contexts

One of the dividing lines between different planning contexts is short-term versus long-term. Long term studies are sometimes also called fundamental planning or technology planning studies. A primary distinction between these two contexts is whether the network capacity is already placed, or where capacity assignment is the goal, typically with a minimum cost objective. In the short-term context where specific capacity is already deployed, the concern is typically with maximizing the amount of demand the as-built network can support. In a future with truly dynamic random demand arrival this will mean minimizing the probability of blocking new service path requests. Algorithms for the best routing of paths and/or logical reconfiguration of existing routes and capacity are the primary focus in this context. The focus in the short-time scale is on operational aspects of the network and on trying to maximize its performance in terms of blocking of new service requests, service provisioning delays or other dynamic performance measures. [\[Dixi03\]](#) (p. 305) presents a somewhat complementary discussion to what follows in that he recognizes a *connection-level* time scale and a separate *capacity-planning* time scale of "months or longer" that includes consideration of both working capacity and protection techniques and capacity.

Capacity Planning Time Scale

The connection-level issues of routing and blocking avoidance in the short-term, with already-deployed capacity, are not the primary focus for this book. Our emphasis is instead on issues surrounding the basic choice of network architecture and different basic survivability principles and strategies that are options for long term planning. This is the context of fundamental network planning or technology and basic network architecture selection studies. These questions are often focused on some form of cost or resource minimization to satisfy a forecast demand. It is accepted that the forecast on which different basic architectures and strategies are compared will probably be in error. But what will be more lasting is the basic understanding and insights into one architecture or strategy relative to the other options. [Table 1-6](#) recognizes these two essentially different planning time-scales and indicates some of the important differences.

Table 1-6. Comparison of short and long-term planning contexts

	Short term (operational)	Long term (planning and design)
Capacities	These are given.	These are to be decided upon.
Architecture	Given	To be decided
Demand Models	Random arrivals/departures or slowly varying demand.	A forecast demand matrix, possibly with different representative demand scenarios.
Typical Problems and Goals	Routing to minimize blocking probabilities and/or rearrangements to routing and protection capacity configuration.	Minimum cost of capacity and equipment to serve forecast demands (or to provide a working capacity envelope within which to support dynamic demand operations).
Channel Assignment	Detailed time-slot or wavelength assignment.	"Capacity only" model assuming full cross-connection.

Connection Level Time Scale

In the short term, network architecture and capacities are given. In long term planning, these are basic decisions to be made. In the short term context, the demand may turn out to be different from the forecasts used in long term planning and "customer churn" is apparent. A kind of limiting case of churn is the notion that lightpath demands may come and go as frequently and unpredictably as individual telephone calls. Many recent papers fall in this category where fixed capacities apply and simulation is used to model blocking probabilities under various lightpath routing and protection schemes where lightpath arrivals and departures are modeled as independent memoryless Poisson processes. The claim is that such "dynamic demand" studies get away from the assumption of a specific "static demand" matrix. And yet the Poisson arrival processes have static mean values that completely characterize the demand between node pairs, indirectly assuming just as accurate a forecast as in the static demand cases. It is in fact, quite difficult to predict or simulate short-term blocking performance or to plan the capacity of a network without ultimately requiring some kind of forecast, or at least a representative demand model accompanied with uncertainty range estimates.

Forecast Uncertainty

The increasing inability to forecast accurately, long enough ahead to deploy networks or augment their physical capacity, is a real issue. Modeling demand arrivals as memoryless Poisson processes of known mean intensities is, however, really no more dynamic than planning capacity to static average-demand forecasts. Even in the most dynamic future environments, however, demand at the transport level will arise from changes in the aggregation of many smaller traffic flows, even when routers are themselves the source of the lightpath demands. Such demand combinations are not subject to complete decorrelation after just a few holding times (which is an implication of ordinary traffic theory), rather they evolve more slowly. In other words, the envelope of an aggregation of flows is inevitably less dynamic in time than the individual flows being aggregated.

What seems, therefore, to be more useful in the face of forecast uncertainty is a means of designing network capacity to provide for an *envelope* of operational states in which a vast number of actual demand patterns is accommodated and protected. In the face of evolving demand patterns that are unpredictable but have considerable correlation times (not completely independent states every holding time), successive batch provisioning and incremental growth and reconfiguration approaches can be developed to manage the dynamic demand problem. All of the design problems treated in later chapters lend themselves to the latter approaches to handling dynamic demand. In particular, [Chapter 5](#) explains the concept of a *working capacity envelope* (WCE) and incremental optimal capacity design models to dimension and adapt network capacity to support dynamic traffic and unpredictable evolutions of the demand pattern. The WCE operational concept also applies directly to p -cycle based networks in [Chapter 10](#). Strategies that attempt in various ways to take forecast uncertainty directly into account in the economic design of a survivable network are essentially in their infancy, but several initial approaches are being studied [[KeOl03](#)], [[LeGr02](#)], [[GeAn01](#)].

A final difference between short and long term planning contexts, especially for optical networks—where full wavelength conversion may not be available at every node—is that of channel assignment. When actually provisioning a new path, or restoring a path, specific time-slot or wavelength assignments must be given. Except for a few cases in the chapters that follow, we consider the "capacity only" problem, which leaves detailed time slot or wavelength assignment as a secondary problem to be solved. The reason for this is that in the long term context, costs and basic architectural efficiencies and other attributes do not depend on the detailed channel assignments, as long as a suitable channel assignment is feasible under the capacity available. This is the case for SONET mesh networking where the B-DCS can assign any STS-1 to any free time slot in an outgoing OC-N. In a lightpath managed optical network the corresponding feasibility of wavelength assignment depends on the availability of wavelength conversion. If the optical cross-connects have an electronic core, then a wavelength conversion capability is implicit and the problem is logically identical to the SONET case. Where all-optical nodes are involved we need only assume that each has a small pool of wavelength converters sufficient to reduce any wavelength blocking to low levels.

Chapter 2. Internet Protocol and Optical Networking

As if in recognition of the turn of the century, the volume of data traffic finally surpassed that of voice in the year 2000. Forecasts over the last 15 years had often predicted this to happen sooner. What made it happen was the Internet and the revolutionary applications that it enabled. The Internet and all its applications use Internet Protocol (IP) packets, making IP the now-dominant form of all traffic. With the growth rate of Internet traffic between 100% to 200% per annum, and 3 to 5% for voice, IP data will only continue to dominate. As voice and other TDM service connections become a smaller fraction of the total traffic, they too will be converted to IP packets for switching and transport. These developments—and parallel advances in DWDM for optical networking—lead to an "IP over optics" paradigm that corresponds directly to the view of "services over transport" in [Chapter 1](#). This is important background for transport network planning as it defines the technological framework for most architecture and design problems in the future. It also introduces new aspects to network design and operational problems, such as wavelength conversion, service-dependent survivability requirements, dynamic transport demands, and the need to plan in a way that accommodates uncertainty about future demand patterns.

Existing transport networks are not particularly efficient at handling huge amounts of data traffic or on-the-fly provisioning. Worse still from a network operator's standpoint, revenues from data are low compared to those from voice, especially given the investments the operator has to make to keep up with data growth. Operators are looking for ways to support data transport with less equipment and with more flexibility and provisioning speed than offered by a full stack of IP, ATM, SONET and DWDM layers. They also question the need to have ubiquitous SONET-layer ring protection, that at least halves the use of installed capacity. Each existing layer is, however, designed for particular functions. IP is best for end-user applications and the huge base of existing Ethernet LAN traffic. But native IP networking lacks reliability. ATM is good on quality of service and reliability, but is not widely used in the end-user environment and is relatively intensive in terms of cell processing and overhead. SONET provides low-delay, low error-rate transport, with protection ring mechanisms defined, but at the same time is rigidly channelized for TDM applications. DWDM provides a way to multiply the physical layer fiber infrastructure to cope with the growth, but lightpaths are expensive and need to be used flexibly and efficiently. Thus the conventional situation for IP data transport is a stacking of usually four layers:

- IP— for user applications and LAN environments
- ATM— for virtual circuit / virtual path capacity engineering, flow control, performance monitoring, virtual networking and QoS guarantees in the data networking layer
- SONET— for high quality transport of payload over the fiber physical medium, error monitoring, OA&M, TDM synchronization and protection switching
- DWDM— for sheer capacity, effectively multiplying the number of fibers in the ground

A problem with so many layers is dealing with the overall complexity of interlayer interfaces, configuration details, and management aspects of each layer, as well as the total amount of space, power and equipment spares and repair costs involved to sustain all the layers. There is also an inevitable loss of capacity efficiency from so many layers.

[\[Team LiB \]](#)

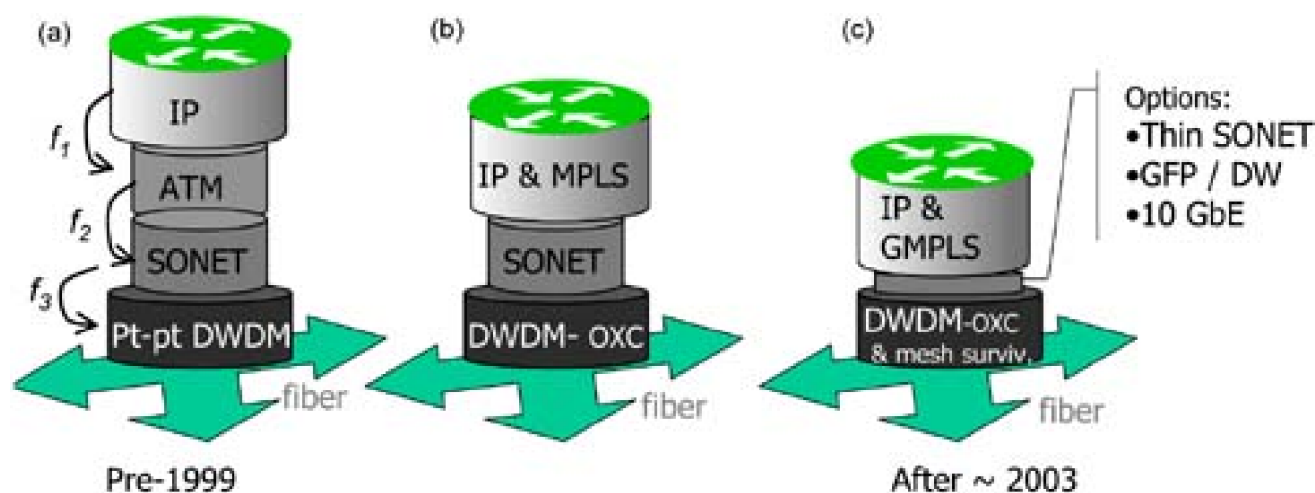
◀ PREVIOUS

NEXT ▶

2.1 Increasing Network Efficiency

One problem with any stacking of layers, is that the grooming of each layer's traffic into demands on the next lower layer inevitably fails to yield 100% fill. The net utilization multiplies all these fractional fill levels and becomes poorer with increase in the number of layers. For instance, the assignment of logical IP links between routers onto ATM VCs or VPs may typically result in 80% utilization (or less) of the bandwidth actually allocated to those VPs. Similarly the STS-3c fill from the ATM VP cell flows may be only 80%. And the fill of the OC-192s on wavelengths in the optical layer may itself be only another 80%. The net utilization of an OC-192 lightpath with customer payload is the product of each individual trans-layer efficiency, or, in this example: $0.8 \times 0.8 \times 0.8$ which is ~51%. [Figure 2-1\(a\)](#) illustrates this cascading of fractional fill levels down the four-level stack. The overall inefficiency is greater still if we also have ring-based protection at the SONET layer. At its most efficient, SONET ring protection adds another multiplier of 0.5 because working and protection channels are exactly matched.

Figure 2-1. Evolution toward "IP over optics:" four layers down to about "2.5 layers."



Achieving higher overall efficiency is one of the central aims of the mesh-based approach to survivability. Despite what commentators sometimes say, capacity is *not* free and capacity efficiency *is* important. Even if capacity was thought to be free (and/or a "glut" of dark fiber abundant in the ground)—planning, installation, testing, and management to bring on additional "lit-fiber" capacity, and integrating it into operational use is certainly not free—so efficiency in its use always pays dividends. An efficient network is inherently more flexible and able to absorb more demand growth or churn before physical additions of new plant and capacity have to be installed. With the Internet driving growth at up to 200% per annum, the problem has often been that capacity simply cannot be laid down fast enough and precisely where it is needed to fully keep up with the growth. In these cases efficiency in the first place, by design, is as good as extra capacity on the ground.

But in fact the simple cost of equipment for added capacity—in the quantities needed is by network operators—is also by no means free. Since much of this book is aimed at the design of capacity-efficient mesh-survivable networks, it is important to address the often remarked view that "capacity is free."^[1] If so, why would one worry about optimizing capacity efficiency? The basis for this is typically a calculation that assumes a lightpath or fiber bearing, say, a fully-loaded OC-192 and takes the ratio of total cost to total Mb/s carried. The numerical ratio is indeed small. Notwithstanding this, an inter-exchange carrier may have annual budgets for *incremental* transport equipment of \$300 to 600 million. It then depends on the point of view as to whether the cost of capacity, and hence the benefit of capacity efficiency, is significant. If mesh efficiencies can take 20% or more off of the latter kind of annual budget, it should be well-worth the CEOs time and attention.

^[1] The point about capacity cost is that it is similar to the notion that "MIPS are free" with computers. Price per MIPS may be asymptotically low, but applications always use more. RAM or disk storage is similar. Price per bit is vanishingly small, but the quantity required is always rising. As a result neither is ever insignificant in practice.

But with the four layer stack it is also complex and labor intensive to provision new services. Each layer tends to have its own management systems. And any one layer can act as a bottleneck to throughput and/or provisioning speed even if all three other layers are

generously provisioned. Thus a simple two layer model for services and transport is attractive: the services layer is IP-based with MPLS and the transport layer is based on a DWDM optical path switched network.

However, in getting to the two layer model, we do not want to abandon the important functions of ATM and SONET in the four layer model. The role of ATM will be assumed by the IP layer, with MPLS and GMPLS to replace ATM. Similarly, many roles of SONET can be referred down into the optical cross-connecting and transport layer. Certain functions of SONET cannot be eliminated—such as formatting bit streams for physical transmission, framing, error monitoring and so on. They can, however, be realized by "lightweight" implementation of the full SONET standards, or by digital wrapper that puts SONET-like overheads on optical channels (to follow). SONET itself is also being extended in terms of flexibility and payload types by developments such as GFP, virtual concatenation, and LCAS. And there is every expectation that—especially in long-haul networks—SONET ring protection can be replaced by a true mesh-based optical transport network (OTN) using optical cross-connects (OXC) and mesh-survivability schemes. The outcome is expected to be a "two and a half" layer model conceptualized in [Figure 2-1\(c\)](#), evolving through the intermediate step, (b). This brings the initially idealized view of "services and transport" layers introduced in [Chapter 1](#), much closer to reality. It also makes all of the capacity planning and design models that follow in the book even more directly relevant to industry practice. Let us now look at some of the specific technologies by which this almost-two-layer model of IP services over optical transport can be realized.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

2.2 DWDM and Optical Networking

In principle, wavelength division multiplexing (WDM) is the same as FDM except that it occurs at optical frequencies—wavelengths in the 1310 and 1550 nm ranges. Several optical carrier signals occupy the same fiber at non-overlapping center wavelengths (or *ls*). A different payload can be simultaneously modulated onto each optical carrier as long as their laser center frequencies are suitably spaced apart. The main drivers for the deployment of WDM multiplexing are the depletion of fiber capacity in long haul networks due to demand growth and favorable economics of WDM systems relative to alternatives such as installing additional fiber or further upgrading single fiber TDM systems to bit rates beyond 10 Gb/s (OC-192). Although the capacity of early point-to-point WDM systems was limited from 2 to 4 *ls*, systems with over 1,000 *ls* have been demonstrated in the laboratory and optical ADMs (OADMs) and optical cross-connects (OXC) allow wavelengths to be routed and switched in much the same way as timeslots are in SONET networks.

This section is intended to give basic background on WDM concepts, issues, and network elements—to provide a self-contained "setting of the stage" for our subsequent studies. To delve deeper, a range of books on DWDM and optical networking are available. Starting from the most accessible, in the sense of the technical content, [\[Kart00\]](#) is a popular primer on optical transmission and technology. [\[Gora01\]](#) covers WDM but also SONET, GbE, LAN and WAN networks in a practical high-level way. [\[Free02\]](#) covers optical fiber, lasers, detectors, amplifiers, regenerators, and WDM multiplex devices somewhat more extensively than [\[Kart00\]](#) but at a similar level of technical depth. [\[StBa99\]](#) and [\[RaSi98\]](#) are more comprehensive textbooks on optical networking including theoretical treatments of topics such as wavelength routing. [\[Rama01\]](#) is a more specialist volume addressing transmission impairment analysis, optical amplifier placement and wavelength conversion technologies. [\[SiSu00\]](#) and [\[Stav01\]](#) are edited compilations of chapters based on specialist research papers on optical network performance issues. Finally, [\[Dixi03\]](#) is a volume that spans both IP and WDM layer considerations with chapters from a variety of experts in both layers. These sources primarily address the challenging basic problem of "working" signal transmission. A few include overviews of protection switching and signaling, or ring concepts for survivability, but generally all are complimentary to the present effort in that they emphasize the technology and basic network planning, leaving the domain of mesh-based survivability strategies to be explored in depth in this book.

2.2.1 Coarse and Dense WDM

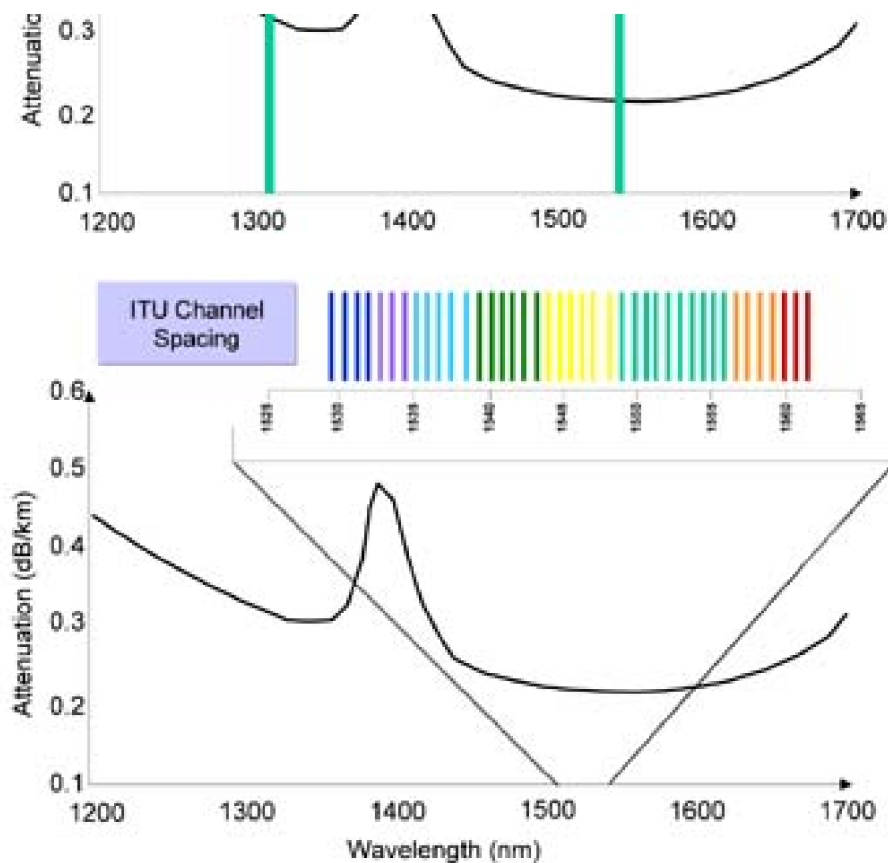
In *dense* wave-division multiplexing (DWDM) numerous laser sources are operated in a closely spaced frequency plan, defining a bank of precisely offset optical carrier frequencies. It is called dense WDM to contrast with the prior technology of *coarse* WDM. In coarse WDM (CWDM) two to four lasers operate independently at widely separated frequency ranges in the 1550 nm and 1310 nm low-loss windows of the optical fiber. CWDM was intended primarily to achieve pair-gain on a point-to-point basis, effectively doubling or tripling the fiber count between two nodes. CWDM lasers could have a much broader spectral occupancy and looser center frequency stability and relative power levels than DWDM laser sources that must be of extremely narrow linewidth and are usually under active feedback control for absolute center frequency stabilization. In DWDM maintaining relatively equal power levels over all carriers can also be an important and difficult challenge not present in CWDM. [Figure 2-2](#) contrasts coarse and dense WDM wavelength plans and illustrates the concept of DWDM based on tightly spaced and controlled optical carriers in the 1500 nm range where optical fiber attenuation is at its lowest.

Figure 2-2. (a) Coarse and (b) dense wave-division multiplexing.

(a) Coarse WDM
(CWDM)



(b) Dense WDM (DWDM)



DWDM systems require precise standardization of the carrier frequencies to use. Current ITU standards define a grid of 81 wavelengths in the "C-band" starting from 1528.77 nm incrementing in multiples of 50 GHz (0.39nm). Commercial systems with 16, 40, 80 and 128 wavelengths per fiber have been announced based on this frequency grid. ITU standards so far only define the 50 or 100 nm spacing grids but laboratory work at 25 nm spacing is occurring and up to 1000 wavelength have been reported on a single fiber [Bell99]. The number and selection of channels implemented on this grid depends on the application requirements such as reach, data rate on the carrier, fiber type, optical filter technologies used, etc., so there is no one set number of wavelength channels associated with all DWDM systems per se.

2.2.2 Optical Amplifiers

The single most important invention that enables DWDM networking is the optical amplifier (OA). An optical amplifier counteracts fiber attenuation over a complete band of lightwave channels at once, without demodulating or in any other way individually processing each optical carrier. In the most common type of OA, an erbium doped fiber amplifier (EDFA), an optical pump laser creates a photon population inversion in a section of erbium doped fiber. The much weaker multi-channel optical input signal from the fiber causes a higher power output of stimulated emission, as in a laser itself, which tracks the input signal thus directly amplifying it in the optical domain. Without this key technology there would be little impetus for DWDM-based optical networking per se because each channel would have to be individually detected and regenerated or amplified each time fiber loss accumulated above threshold. There would still be multiplexing gain in the use of the fiber itself, but no corresponding relief in the amount of physical equipment needed at each regenerator or amplifier station along the fiber route. The EDFA has been so successful that the term is often used as a synonym for OA itself. But there are other OA technologies, such as the semiconductor optical amplifier (SOA) and erbium doped waveguide amplifier (EDWA).

2.2.3 Regenerators

The optical amplifier is an analog device. It functions as a broadband linear amplifier of incoming light power to offset fiber attenuation or other insertion losses in the optical path. In practice, OAs are neither perfectly linear nor noise-free. They add noise as well as boost any

incoming noise with the signal. Other fiber and amplifier related impairments accumulate in the optical path such as chromatic dispersion, polarization mode dispersion, and types of crosstalk. The important point here is that these impairments add up to a point where *regeneration* (as opposed to amplification) is required. In regeneration, more specifically in a "3R" regenerator, each channel is individually demodulated from its optical carrier and converted to an electrical form where its bit-timing is extracted and used to sample the binary state of each symbol. Jitter is removed from the recovered clock, new 1/0 symbols generated, and these are remodulated onto a new outgoing carrier wavelength. "3R" thus refers to retiming, regenerating and retransmitting.

Regeneration is a per-channel process involving electro-optical conversion and high speed electrical processing of each channel individually and is therefore more costly than optical amplification. Often the regeneration function needed along an optical path would be provided by optical-electrical-optical (o-e-o) cross-connects along the route of the path. The need for regeneration is one of the reasons that "transparency" (optical-only processing without wavelength conversion) is likely to be limited in optical networks. As long as there is a need for regeneration, it makes sense for at least some of the cross-connects in the network to be electrical-core devices as opposed to purely passive optical (o-o-o) space switches. Any time the signal has been demodulated to the electronic domain for regeneration or switching, we can also change wavelengths as needed because the signal is applied to a new transmitting laser to return it to the optical domain. Less often a bank of stand-alone regenerators may be needed on long-haul facility routes, contributing considerable extra cost to the signal path. With present OA spacings of around 60 to 80 km, regenerator spacings are typically around 550 to 600 km. Note that in the design of the DWDM link these spacings are not independent. If OAs are more frequent, optical signal to noise ratio will be better preserved allowing for a longer distance between regenerators (and the converse).

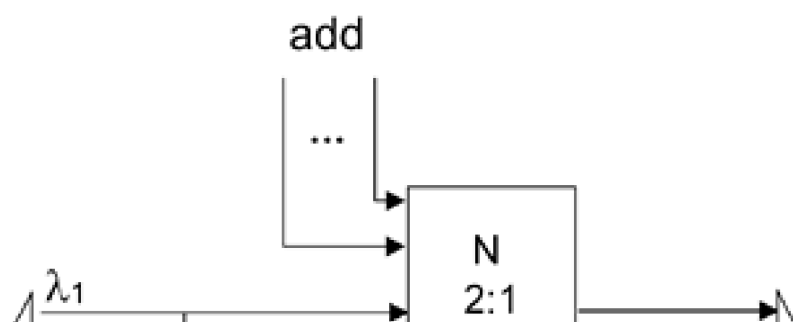
For a variety of network planning purposes, the detailed design of a transmission span between nodes is usually summarized into a single incremental cost for adding a channel to the span as a whole or a set of costs associated with modular capacity addition options along the entire span. Such cost coefficients summarize the net effect of the detailed transmission design efforts, taking into account each span's different length and exact equipment types and spacings and each of the technology options available to the planner. The per-channel cost may or may not include a pro-rated allocation of all the "get started" costs associated with the acquisition of the route, installing ducts, cables and equipment huts, etc., before the first channel is turned up.

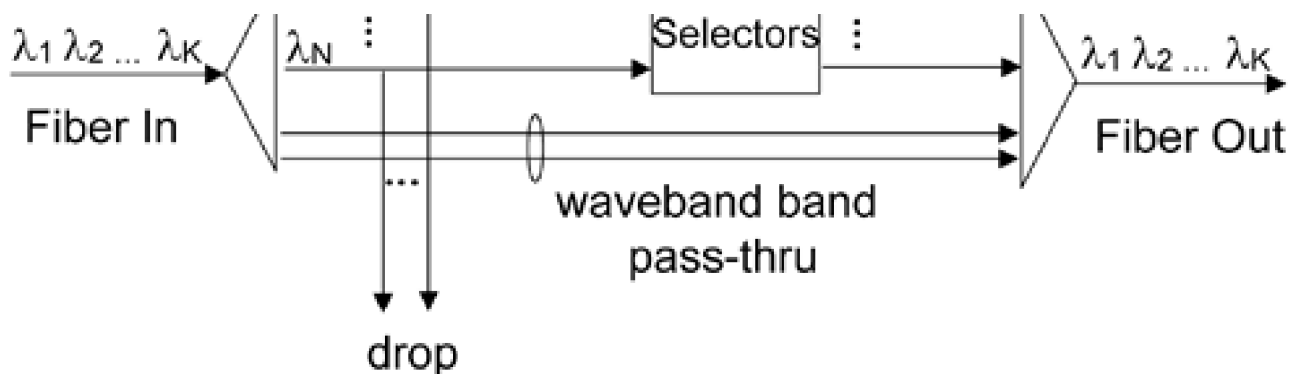
2.2.4 Optical Add/drop Multiplexers (OADMs)

The structure and function of an OADM almost completely follows from its SONET ADM predecessors, although a variety of different technical implementations exist for selecting the optical channels to add or drop. In principle an OADM can be based on electronic detection of payloads, electronic add/drop, and remodulation of pass-through and add signals onto new outgoing wavelengths. Such an OADM is capable of full wavelength conversion as well as add/drop and pass-through functions. But especially in an OADM it is desirable to leave pass-through channels in the optical domain. Many OADM designs therefore tend to be based on passive, purely-optical add/drop/through channel filters without any wavelength conversion. Only add/drop channels need to go through e/o and o/e conversion respectively.

In addition, because of the technical challenges of selecting or inserting single wavelengths, OADMs may be based on a *waveband* add/drop principle, where groups of 4 or 16 wavelengths are handled as a whole for add or drop purposes and all other wavelengths pass through the OADM to the other fiber. Also, because of the technological challenges involved, an OADM may or may not be reconfigurable under program control. A fully reconfigurable OADM (called a ROADM) provides complete wavelength routing functionality, selecting any incoming channel from one fiber to either drop locally (on the same wavelength) or pass on through to the outgoing fiber. Similarly the outgoing wavelength multiplex or can be fed on a per-channel basis either from the incoming line or the local add port. [Figure 2-3](#) illustrates some of these key aspects of an OADM.

Figure 2-3. An OADM multiplexer where N of K channels can be added or dropped.





The OADM handles one fiber pair (of which only one direction is shown) on which there may be K wavelength channels. Generally some number N , less than K , channels can be selected for add/drop functions. If the incoming WDM demux is programmable in terms of the wavelengths selectable, or if an optical switch is included (not shown), then the OADM is reconfigurable in that any N channels may be added or dropped. Otherwise only a specific block of N channels is accessible for add/drop processing at the OADM, based on the optical filter sets physically installed in the OADM. Optical interference filters based on thin film coatings and fiber Bragg gratings are examples of technologies used to separate channel bands by reflectance outside the passband and transmission in the band. The same filter techniques can be used to select the drop channels as well as insert the add channels and remultiplex all the outgoing channels. The incoming optical channel demultiplexer can be based on diffraction gratings, waveguide interferometers or series connected single-channel reflectance filters to separate each wavelength into a separate local fiber for processing in the OADM.

2.2.5 Transparent, Opaque and Translucent Optical Networks

In its purest form "all-optical" networking refers to transport networking in which the logical connections between node pairs are realized by assigning them a specific DWDM wavelength path (of suitably low loss, noise and distortion) end-to-end. Each lightpath must be routed from its source to destination without any electronic processing at intermediate nodes. Thus resulting path is said to be "transparent" not (ironically) because it is optical all the way, but because any payload can be modulated on the respective optical carrier wavelength and received at the far end. In other words, transparency refers to the fact that there is no dependency on the payload being in a specific format in terms of framing, bit-rate, line-coding, power level, jitter and so on, as is usually the case when electrical circuits have to handle the signal en route. In practice today, transparency requires a wavelength assignment that must be uniquely reserved for the path on each fiber en route. But in principle a transparent network will allow a lightpath to occupy different wavelengths on its links when an all-optical wavelength conversion technology is available. Such an all-optical path that does not change wavelength is also called a pure *wavelength path* (WP). The main advantages of WP networking are that wavelength converters (WCs) are not needed and the path is transparent to payload. In practice, a pure WP network would also employ only all-optical (o-o-o) cross-connects that generally take less power and space than a corresponding o-e-o cross-connect. Disadvantages associated with pure WP networking are, however, as follows:

- Wavelength assignment is much more complex because lightpaths must use the same wavelength end-to-end. To assign a wavelength then requires that it is free on all spans en route. Logically this is like operating a SONET network in which payloads cannot change their timeslot assignments in successive spans. The shortest path routing problem for working demands is replaced by the routing and wavelength assignment (RWA) problem, which is NP-hard (a measure of computational complexity covered in [Chapter 4](#)).
- To avoid *wavelength blocking*, where a path cannot be routed because a single wavelength is not available on every span of any route, more total capacity (lightwave channels) is needed on each span and the DWDM technology employed must also support more wavelengths per fiber than a corresponding network where wavelengths can be re-assigned at each node.
- The benefit of regeneration is not obtained to "reset" the accumulation of optical transmission impairments at each node. Each optical path must therefore be individually transmission engineered or limits imposed on the length and routing of end-to-end paths to contain transmission impairments and loss.
- For the same reason, fault isolation becomes more difficult and there are no signal-associated overhead channels to support numerous operations and maintenance functions that employ payload-associated adjacent node signaling.

Current technological limitations, and issues in network control and management, also make such all-optical networks difficult to implement. Performance assurance is especially an issue if switched all-optical paths are to bear 10 to 40 Gb/s payloads reliably. At these rates the point-to-point DWDM link design problem is so dependent on dispersion, noise, polarization, nonlinear and relative-power effects

in fiber and OAs that detection and electronic regeneration at each OXC node is almost required to retain assurances of end-to-end BER under arbitrary patterns of cross-connection of carriers between multi-channel DWDM links.

The opposite of a pure WP or "transparent" network, that consists only of transparent paths, is (not surprisingly) called an "opaque" or *virtual wavelength path* (VWP) network. A VWP is defined as an end-to-end optical path that may use more than one optical wavelength along its route. Wavelength assignments to VWPs can be changed as needed, through wavelength conversion (WC) at cross-connect nodes. In the future, all-optical wavelength conversion may be possible, in which case a VWP could remain all-optical end-to-end, but at present wavelength conversion technically implies optical to electrical conversion and remodulation onto a new laser (i.e., o-e-o processing). There are practical advantages, however, to being "forced" to do o-e-o conversion in the OXC. Once the payload is returned to the electrical domain it can be accessed for performance monitoring, regeneration, and overhead signaling functions as well. Provisioning operations, capacity design, fault-location, transmission engineering and capacity requirements are all more favorable with VWP networking. In particular wavelength blocking is eliminated so that problems of routing and capacity design are all logically similar to that of a SONET environment. In addition, Digital Wrapper (DW) and Generalized Framing Protocol (GFP) developments (to follow) both address the issue of retaining *payload transparency* without requiring optical transparency. In other words, payloads can have digital rate and format transparency through o-e-o conversion and regeneration stages.

However, a fully opaque network implies O/E and E/O transponders at each node and large electronic switching cores, which are expensive and power-consuming operating at 10 to 40 Gb/s. Thus, using large electronic core optical switches to form an entirely opaque network may not be feasible. Studies have shown, however that—from a capacity and blocking standpoint—a small pool of WCs associated with each optical cross-connect, or a small number of nodes that have full WC capabilities [StDo00], can effectively eliminate lightpath blocking.

This leads to the concept of *translucent* optical networks that strike a practical balance between transparency and opaqueness in an optical network in terms of either "transparent islands" or a "translucent" optical network. In the former, a large-scale optical network is divided into several domains (i.e., islands) of optical transparency. The idea is that within a domain, transmission engineering factors are suitably controlled or assured so that a lightpath can transparently reach any node without intermediate signal regeneration. But for communications between different domains, electronic switches are used at the domain boundaries, acting as 3R regenerators and wavelength converters while relaying the lightpaths crossing the domain boundaries.

A translucent optical network is a somewhat more general option in which the idea is to have a relatively small number of strategically chosen opaque nodes (VWP-OXC) at which wavelength conversion and regeneration is possible, with all other nodes being transparent WP-OXC. This is called sparse placement of the o-e-o switches. Rather than being dedicated to routing lightpaths only in and out of transparent islands, these switches are shared by all paths of the network as a whole. Studies have shown that with about one in three nodes being VWP-OXC nodes, wavelength mismatch blocking can be essentially eliminated from the network. Another highly pragmatic approach is to use PVWP OXC nodes everywhere with the pool of WC resources dimensioned so as to essentially eliminate wavelength blocking. The number of WC resources may often be no more than needed to provide for unavoidable signal regeneration needs in any case. Typically, if there is a small pool of wavelength converters at each OXC node, lightpath blocking performance is close to that of a network with full wavelength conversion on each port at every node. A translucent network may, therefore, either have subset of OXC nodes that have a full WC capability (i.e., o-e-o nodes) or be comprised of all-optical OXC nodes that have a limited pool of wavelength-changing transducers to avoid lightpath blocking situations or to apply regeneration for an optical signal reaching its limits of noise accumulation. For further reference see [RaSa97], [YaLa96], [SuAz96], [SuAz98].

Industry terminology refers to either type of path (WP or VWP) generically as a *lightpath* when the WP or VWP distinction is not crucial. Similarly, all three types of optical network, transparent, opaque, or translucent are generically called an *optical transport network* (OTN).

2.2.6 Routing and Wavelength Assignment (RWA) Problem

In an optical network where there is no wavelength conversion or limited conversion abilities (insufficient to make lightpath blocking negligible) one encounters the *routing and wavelength assignment* (RWA) problem. In general, there are two ways that blocking can occur, either through capacity blocking or wavelength mismatch blocking.

Capacity blocking is a simple insufficiency of available channels to support any routing of an additional lightpath between a desired pair of nodes. This is logically identical to the blocking that arises in attempting to route telephone connections through a network of trunk groups. In a data network, or a SONET DCS-based mesh network, for example, the routing of demands that minimizes cost is generally a shortest path route. Total flow may be balanced over a few shortest routes, and if survivability is concerned the working route choices may change somewhat, but the route is chosen independent of the individual channels that may be employed, because a SONET DCS can always connect any input timeslot to any output timeslot. Blocking on the desired route would arise only from a shortage of total capacity on some span.

If, however, the OXC nodes in a DWDM network cannot change wavelength, or are limited in doing so, the choice of route and the assignment of wavelength(s) must be a combined decision if resource requirements are to be minimized. This gives rise to the intensively studied RWA problem and algorithms for RWA. In this context, blocking may arise even if there are available channels, because an end-to-end assignment of a single wavelength that can use the available channel(s) is not possible.

RWA problems can be either on-line or off-line in nature. An off-line RWA problem assumes all lightpath requirements are known and we seek a single overall solution for the assignment of routes and wavelengths to meet lightpath requirements. A typical objective is to serve all requirements using the fewest number of wavelengths in total or if the number of wavelengths is given, to minimize the total number of unserved demands. The wavelength assigned to each path must be free on at least one fiber pair on each hop of the route assigned to the same demand. Thus off-line RWA problems usually take the form, or are a built-in part of, overall network design problems. Interestingly, while routing as a standalone problem, and wavelength assignment as a separate problem (given a route) are individually simple polynomial-time problems, their combined global solution for a set of demands is of exponential complexity for exact solution (more on these concepts in [Section 4.2](#)).

On-line RWA algorithms are the operational counterpart intended for handling one demand at a time in a dynamic demand arrival/departure environment. On-line RWA algorithms for mesh networks are much simpler than off-line RWA problems because they necessarily handle each demand individually (in a "greedy" algorithmic sense). A useful basic concept underlying on-line RWA algorithms is the *layered graph* approach. The layered graph is a copy of the actual topology instantiated in each wavelength that the network employs. In a pure WP network all these graph layers are interconnected only at the access nodes where initial wavelength assignments can be made. If certain other nodes have full or limited wavelength conversion capabilities, this is also represented by links connecting the corresponding wavelength planes. Depending on the technology of the nodes, the inter-layer links can be either zero-cost or reflect the cost of wavelength conversion. (Note that in the limit of a full conversion at each node, the layered graph simply becomes the multigraph expansion of what we otherwise normally treat as a simple capacitated graph.) Any links currently used in any layer are marked with infinite cost. Once the layered graph (plus links in use) construct has been built or updated, the on-line RWA problem can be solved as a shortest (i.e., least-cost) path finding problem through the overall graph. The overall graph has many more links than the one physical graph, but shortest-path problems can be solved quickly at large sizes, so this is not necessarily onerous. A lightpath request is blocked only if no path then exists for it at all in the overall layered graph. Otherwise the path found specifies both the route and the wavelength (or wavelengths) assigned based on the layer (or layers) traversed by the end-to-end shortest path.

Additional heuristic principles are usually also built into RWA algorithms to "hedge" on the exhaustion of capacity over edges through strategies where the link routing cost in each layer inversely reflects how much capacity remains on the graph edge as a whole (e.g, the distributed relative capacity loss algorithm (DRCL) [[ZaJu00](#)]). Another general principle that improves overall blocking (and speeds execution) is similar to the concept of "link-packing" in multistage switching networks. That is, to systematically consider each wavelength layer in sequence, always placing new paths into the first layer at which a route is feasible (e.g., the adaptive first fit (FF) algorithm—also in [[ZaJu00](#)]). Over a network as a whole, link-packing or first-fit principles tend to keep the incoming and outgoing links on each wavelength at a node either both in use (together) or both free, ready for a new path establishment. Without a link-packing bias, blocking is more likely because the free links that exist at a node have no tendency to be available as pairs on the same wavelength, facilitating transparent optical connections.

The RWA problem and RWA algorithms are already extensively covered in the literature. See for example [[StBa99](#)], [[RaSi98](#)], [[Rama01](#)], [[SiSu00](#)], [[Stav01](#)], [[Dixi03](#)] and the surveys [[ZaJu00](#)], [[AsSh01](#)], [[RaSi95](#)], [[ShBo00](#)]. It is important here only that we understand the issue and nature of RWA. Some of the design models that follow in later studies inherently involve solutions of (off-line) RWA as constituent subproblems. But because the book is not primarily focused on the operational time-scale we do not consider on-line RWA algorithms further.

[[Team LiB](#)]

◀ PREVIOUS NEXT ▶

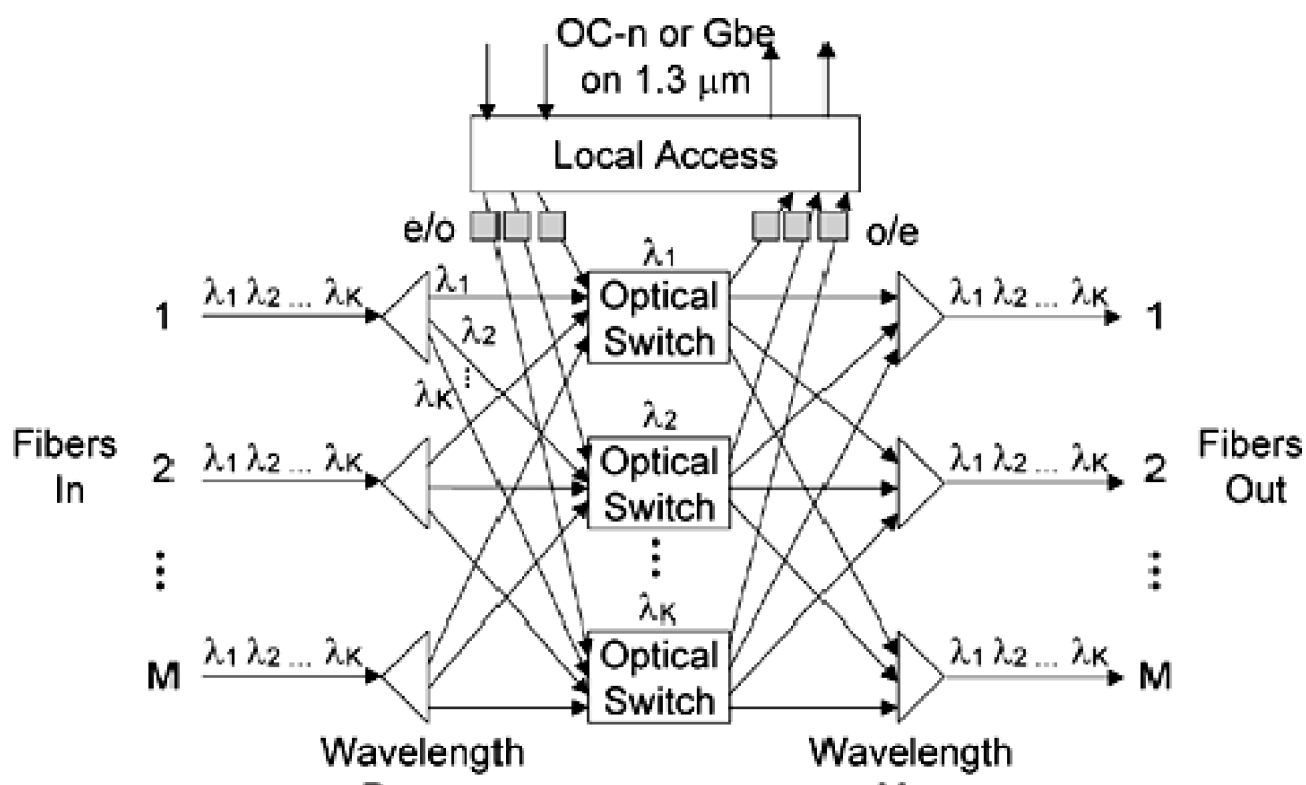
2.3 Optical Cross-Connects (OXC)

Optical cross-connects (OXC) play a role in DWDM-based networking that is analogous to that of the B-DCS in a SONET mesh network. Both are non-blocking space switches that can connect from any input fiber to any output fiber and perform switching in a secondary dimension as well. For the B-DCS this is time-slot position switching between the incoming/outgoing OC-n frames on each fiber. For the OXC, it is switching from one wavelength to another, as well as from one fiber to another in space. Generally, a B-DCS would always be capable of switching STS tributaries from any input fiber to any timeslot on any output fiber, whereas OXCs may not have a full wavelength switching capability.

2.3.1 Wavelength Path Optical Cross-connect (WP-OXC)

In wavelength path (WP) optical network, the nodes cannot perform any wavelength conversion. As a consequence, every path must have the same wavelength on each fiber. The respective OXC architecture is depicted in [Figure 2-4](#). The wavelength channels on each incoming fiber are demultiplexed (but remain optical) to separate local jumper fibers. Then all copies of the same wavelength from each incoming fiber are directed to an optical switch module where they can be routed to any outgoing fiber and multiplexed with other wavelengths. Note that because wavelength channels are already separated when demultiplexed, the middle-stage optical switches do not have to be frequency selective or frequency specific; they need only redirect the entire band of light off each incoming fiber to a specified outgoing fiber. In other words, the central optical switches can be essentially a set of mirrors. [Figure 2-5](#) shows a 2-D MEMS (micro electro-mechanical system) implementation of an 8x8 switchable mirror array, a prime candidate for this type of pure optical space-switch.

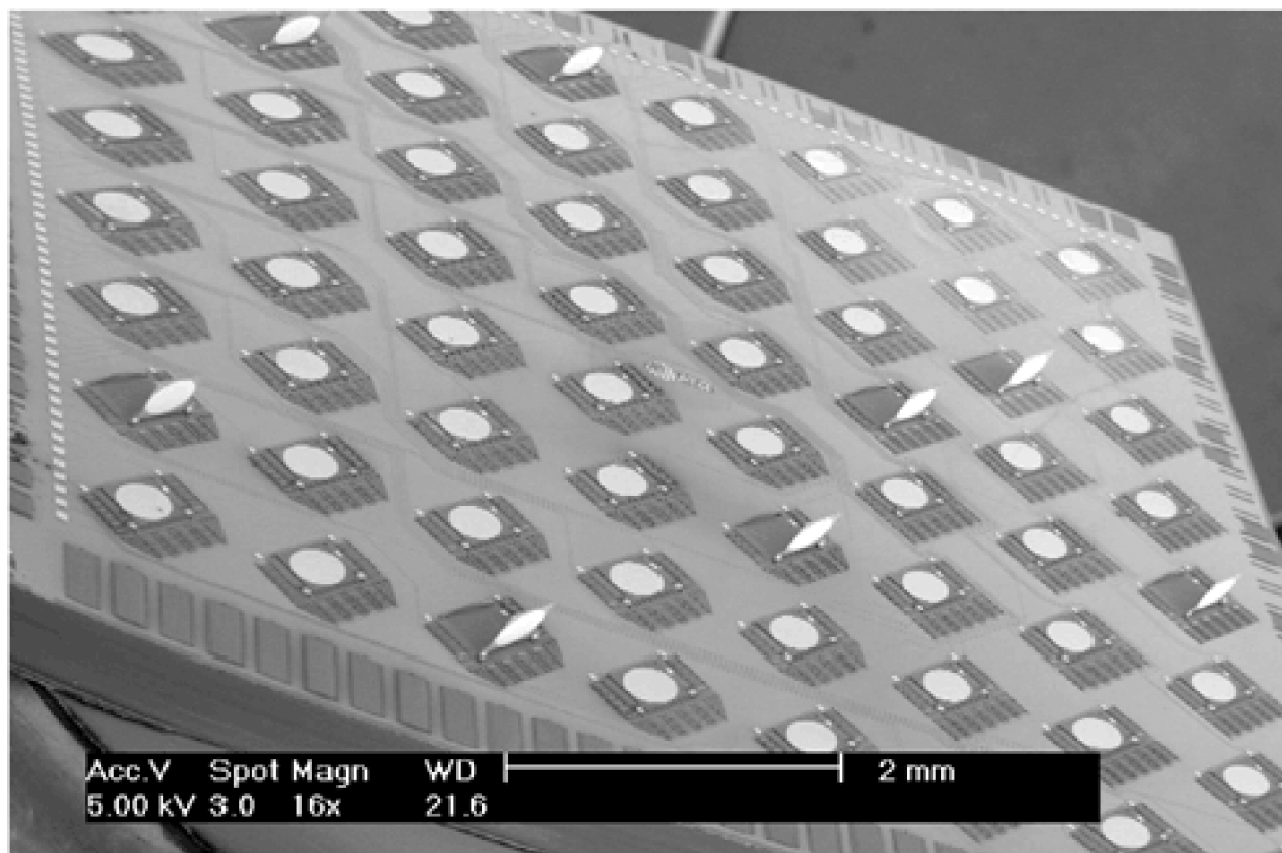
Figure 2-4. Pure WP-OXC with no wavelength conversion or regeneration (except add/drop transponders).



Demux

MUX

Figure 2-5. An 8x8 optical layer switch based on 2-dimensional MEMS (source [Lin98](#)).



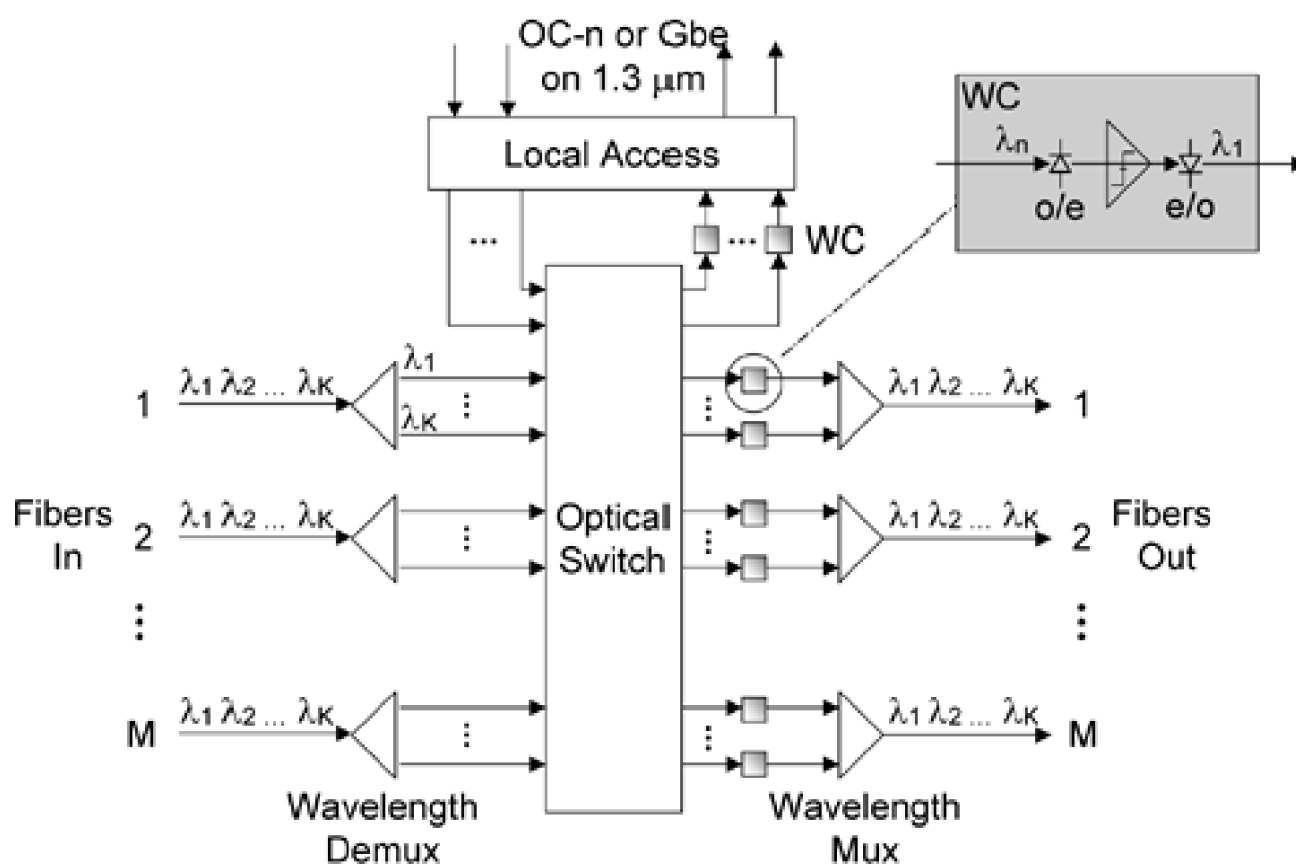
The overall structure of the OXC shown in [Figure 2-4](#) makes use of the fact that each wavelength can be used only one time per fiber. For M incoming and M outgoing fibers, each with K wavelengths, K separate $(M+k) \times (M+k)$ optical space switches are used where k is an allowance for the desired number of local add/drop channel opportunities, managed by the local access interface. Electro-optic conversion is provided at the local add/drop interface so that drop signals may be demodulated from their DWDM transport wavelength (in the 1550 nm range), regenerated electrically by a corresponding interface card (for example, a SONET OC-n receiver or a Gigabit Ethernet (GbE) receiver). The regenerated drop signals may then be delivered electrically to terminating equipment if close by, or, more commonly delivered up to a few km over a "short reach" fiber link using less stringent lower cost single-fiber-per-carrier optics in the 1.3 μm range. The latter short-reach fiber link is sometimes referred to as "uncolored" or "gray" optics because it is now outside of the DWDM domain and there is no highly precise requirement for the single 1.3 μm laser wavelength used on such links (in contrast to when the signal was borne on a precisely "colored" DWDM carrier in the 1550 nm range). In the "add" signal path at the local access interface, OC-n, GbE, or other payload formats are received (also on short reach optics) and either further prepared for transmission (by addition of GFP or DW overheads, for instance) or directly applied to modulate a laser source at the assigned DWDM wavelength. The circuit pack that interfaces to a payload in its native format and/or further preparing it for transmission and applies it to an assigned DWDM wavelength (and vice-versa) is often called a *transponder*.

2.3.2 Optical Cross-Connect for Virtual Wavelength Paths (VWP-OXC)

In VWP (or "opaque") optical networks, the OXC nodes are capable of full regeneration and wavelength conversion. Any incoming lightpath can be transformed to any other wavelength on any other outgoing fiber link through optical-to-electrical conversion and remodulation on a new laser wavelength. The VWP-OXC is therefore also often called an *o-e-o* switch. (To effect the VWP-OXC function without electro-optic conversions will eventually require a pure optical wavelength translation technology.) [Figure 2-6](#) shows one architecture for a VWP-OXC using a single $(M-K+k) \times (M-K+k)$ optical space-switch core. Following the core space switch each lightpath can change its wavelength before entering the next fiber. This is why the core switch must be able to access the whole set of fibers

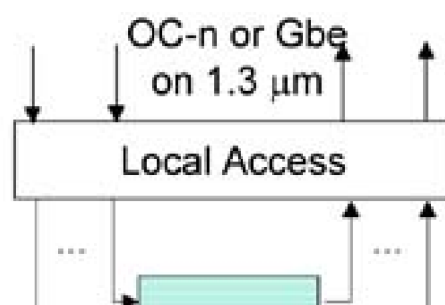
independently in each wavelength. Wavelength converters are also available to reassign drop channels to the desired local wavelength, or, more likely, to simply demodulate the selected channels and reassign their payloads to short-reach optics.

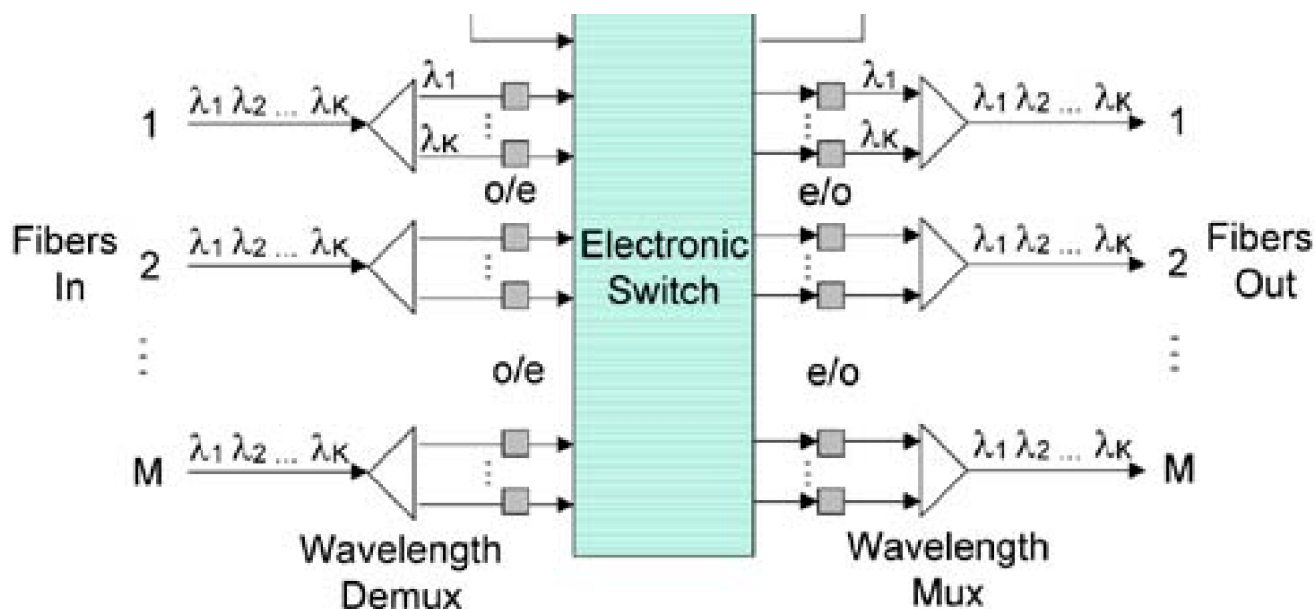
Figure 2-6. VWP-OXC with optical switch core followed by separate wavelength converters.



An alternative implementation of the VWP-OXC based on use of a purely electronic switch core is shown in [Figure 2-7](#). This is the actual architecture of the first o-e-o cross-connects. The central electronic switch core is essentially the same as that for a SONET B-DCS. The core operates at some standard synchronous rate and may itself be implemented in several stages (typically a 3- to 5-stage CLOS non-blocking configuration, not shown). Around the core each wavelength channel is demodulated and regenerated with a receiver circuit pack specialized for the payload rate and format on the given channel. The payload is adapted to the format and rate of the core for switching. Note that to keep the clock speed of the electronic core within limits of available circuit technologies, this can require parallel paths or inverse multiplexing techniques where, for instance, four paths would be needed through a 2.5 Gb/s core to switch a 10 Gb/s payload. More commonly, incoming OC-192s are already channelized with 2.5 Gb/s tributaries or GbE so each of these is demultiplexed and cross-connected individually. Ultimately, however, the bandwidth requirements through the core drive the OXC technology to an optical switch core followed by o-e-o conversion after switching as in [Figure 2-6](#). The $M \cdot K$ o-e-o modules following the optical switch in [Figure 2-6](#) represent much less high speed electronics than in [Figure 2-7](#), but is only possible if much larger all-optical switches can be built. (This is the arena for 3-D MEMS where many more input/output ports can be realized in the same space but require variable angle deflection and control of each mirror. Stacked 2-D MEMS arrays are also being developed to enable the OXC architecture of [Figure 2-6](#).)

Figure 2-7. VWP-OXC with electronic switch core followed by fixed or tunable laser sources.

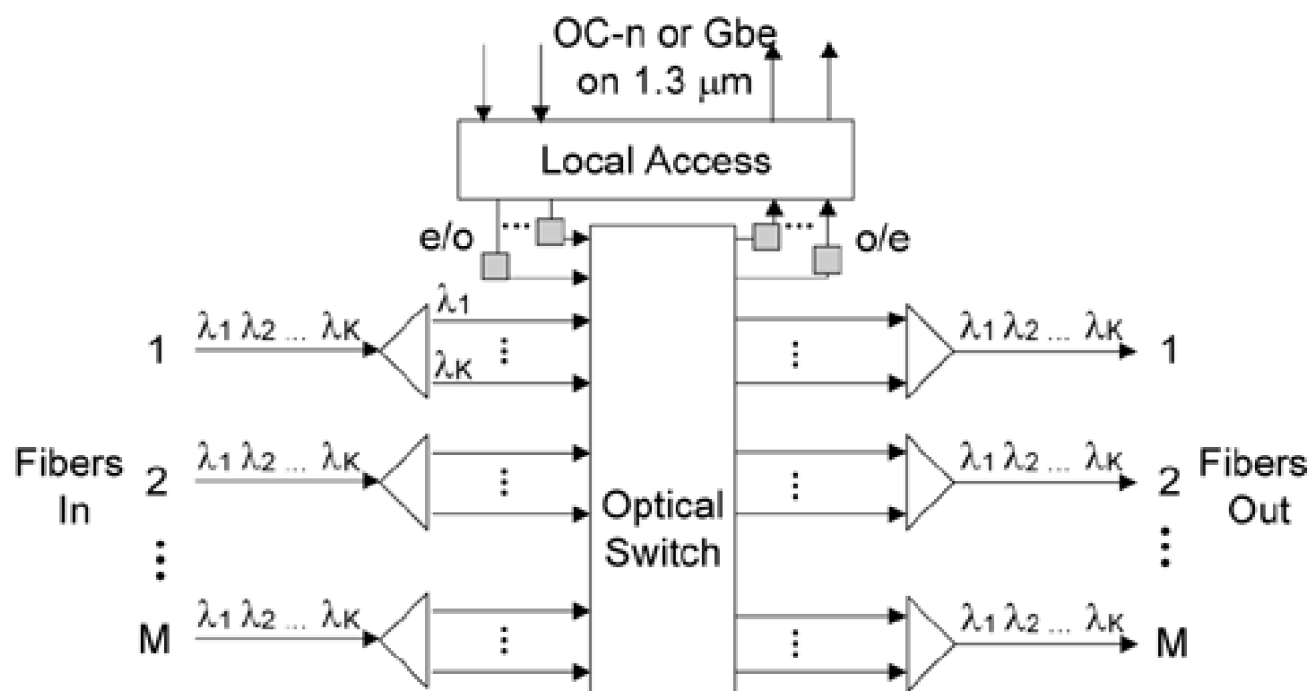


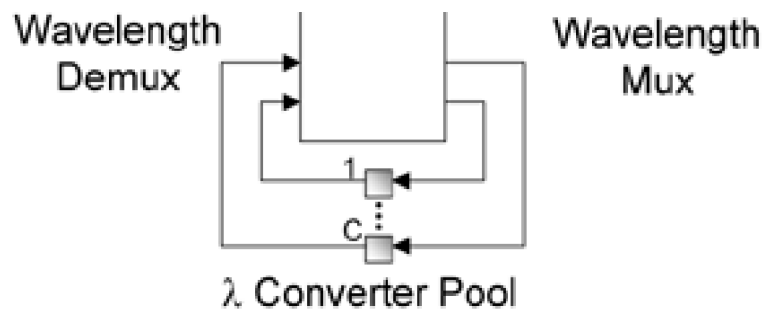


2.3.3 Partial Wavelength Converting OXC (PVWP-OXC)

An important type of translucent or partially VWP network in practice may consist of OXC nodes that can perform a limited number of wavelength conversions using a shared pool of WCs, as in [Figure 2-8](#). If there is no wavelength conversion required for a given lightpath, the optical carrier is space-switched to the appropriate outgoing fiber, or the local access. Otherwise the lightpath can be locally switched to a free port in the pool of C o-e-o converters. From there, the WC output is space-switched on its new wavelength to the desired output fiber. The signal path thus circulates once through the space switch core to change its frequency to one that is free on the required outgoing fiber. If wavelength conversions for the local add/drop channels are needed they can also be performed this way, eliminating the need for separate WCs between the optical switch and the local access. The optical switch core has a dimension of $(M \cdot K + k + C) \times (M \cdot K + k + C)$ for full interconnection of the pass-through traffic, converter pool access, and add/drop.

Figure 2-8. A PVWP-OXC with a small pool of wavelength converters can approximate the function of a full VWP-OXC.





The PVWP-OXC architecture may be the most important and practical approach for optical networking because the wavelength blocking of a pure WP network can be eliminated in practice with a small number of WCs at what is otherwise a WP-OXC node. This implies that the lower capacity requirements of a completely VWP network are realizable with most of the lower cost benefits of WP-OXC nodes. The extra cost for the relatively small pool of WCs may also be offset by the fact that regeneration has to be supported for a certain number of paths through a typical node in any case.

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

2.4 Data-Centric Payloads and Formats for the Transport Network

Let us now familiarize ourselves with the IP or services side of the "IP over optics" theme. One of the aspirations of ATM and B-ISDN was to move the signal formats and technologies used previously in the wide-area transport network (WAN) into the local area network (LAN) arena. The vision was of desktop computers interfacing at OC-3 and generating and processing ATM cells. However, due to the huge embedded base, lower cost, and familiarity with Ethernet and the Internet Protocol, ATM lost out to IP/Ethernet in the LAN. In fact, it has turned out that where the WAN could not invade the LAN, the opposite has happened: LAN signal formats and payload, and those native to storage area networks (SANs), are now being adopted for long-reach interconnect. These provide a range of new signal types for the transport network to convey.

2.4.1 Computer Interconnect and SAN

Fiber Channel is a computer interconnect protocol and format for high performance short-reach information transfer. It is commonly used to connect peripheral devices such as a disk drive, printer, tape drive, etc. to a workstation using SCSI or HIPPI higher-layer protocols. Fiber Channel is also used in storage area networking. Being able to access mass storage devices quickly and from greater distances is attractive to applications such as multimedia, medical imaging, and scientific visualization. Fiber Channel is also used for mass storage backup at off-site locations.

The Enterprise Serial Connection (ENSCON) is an IBM technology for replacing numerous copper-based I/O channels on large mainframe computers with a high capacity fiber-based interface and circuit-switched access of peripherals (printers, disk storage, tape, etc.) to the mainframe. The High Performance Parallel Interface (HIPPI) is an 800 Mb/s parallel electrical I/O interface standard limited to 25 meters reach due to parallel clock skew limitations. For longer reach HIPPI includes a serialized 1.2 Gb/s format for short-reach fiber optics. HIPPI was a precursor to Fiber Channel and is used in the same type of super-computer/mainframe interconnect type of applications. "Infiniband" is a multi-purpose high-speed computer interconnection architecture optimized for IP traffic. Its applications emphasize large enterprise data centers and ISPs server clusters where high speed serial interconnections reduce cabling and connectors and physical space requirements extensively. Each 2.5 Gb/s pair of wires or fibers is called a link. High capacity and scalability is supported by allowing growth from one to 4 or 12-wire link widths.

2.4.2 Gigabit Ethernet (GbE) and 10 Gb/s Ethernet (10GbE)

Perhaps most important among the non-traditional payloads for transport is Gigabit Ethernet (GbE). With the increase of desktop computing power and Internet-based applications using TCP/IP, IP addressing and packet formats have become the dominant standard worldwide. LAN interconnect and Virtual Private Networks (VPNs) have also turned out to be "killer applications" for the bandwidths made available by optical networking. GbE is a native format for short-reach high capacity LAN interconnections, but is not a format suitable for WAN transport. 10 GbE on the other hand is intended to be directly usable as a WAN transport format for IP LAN interconnect.

The original Ethernet provided a shared medium LAN operating at 10 Mb/s using carrier sense multiple access and collision detection (CSMA/CD). What made Ethernet unique was CSMA/CD. In other respects it uses IP packets in HDLC frames, which can actually be transported in many different ways and on different LANs and WANs. Being the first widely used LAN PC technology, Ethernet transceiver manufacturing costs went so low that it became the ubiquitous market standard in the LAN. Ethernet became a real-world demonstration of its own inventor's "theorem" (called Metcalfe's Law) that the value of a network grows exponentially with the number of users it serves. In this case the value of going with Ethernet, rather than a competing LAN technology, grew the more that other users also relied on Ethernet. This dominance was only further established with 100 Mb/s Ethernet on Category 5 twisted pair cabling (100BaseT). Although CSMA-CD is still supported in theory between the two end-nodes of a 100BaseT link in half-duplex mode,

practically all Ethernet LANs adopted a hub or switched star architecture for 100 Mb/s operation. A natural outgrowth of the hub or switched architectures was the desirability for an even higher-rate version of Ethernet to directly exchange aggregations of statistically multiplexed traffic between such devices, as well as IP routers. This led to 1 GbE and 10 GbE.

GbE in the form of 1000BaseT allows a 1 Gb/s point-to-point link over four pairs of Category 5 unshielded twisted pairs to a maximum reach of 100 m. GbE in practice abandons the textbook idea of many sources accessing a media by CSMA/CD. 10 GbE is being developed with direct WAN transport applications in mind and is exclusively a full-duplex point-to-point technology (no CSMA/CD involved). 10 GbE employs simplified SONET/SDH framing and scrambling in its WAN interface. Because the line rate of SONET OC-192 (or SDH STM-64) is within a few percent of 10 Gbps, GbE can use the same higher layer access protocols, although the line rate of the 10 GbE WAN physical layer is actually 9.953 Gb/s, whereas the 10GbE LAN version has a physical layer rate of exactly 10 Gb/s. The binary IP bit stream is 64b-66b coded for compatibility with the 10 GbE electrical chip-level interface. SONET frame layer scrambling then prepares the signal for serial optical transmission. This reuse of SONET is an aspect that helps make 10GbE even more cost-effective because existing OC-192 optics, laser drivers, clock recoveries, scrambling and decision circuits from several suppliers can all be reused for the 10 GbE application. Direct IP interconnection of high capacity routers is a prime application for 10 GbE. Physical media interface specifications for 10 GbE include short-reach 1310 nm optics reaching 10 km and 1550 nm (single carrier) optics reaching 40 km, allowing 10 GbE to be used as an access format from customer premises to a transport node. Further discussion of GbE and 10 GbE can be found in [\[Gora01\]](#).

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

2.5 Enhancing SONET for Data Transport

The motivation for long-haul transport of these formats is to keep their payloads as much as possible in the native format and structure in which they exist in the LAN environment. Doing so can eliminate the latency, power, and space consumption, and complexity of management, of going through a stack of adaptation protocols. For some applications it is also important to permit statistically multiplexed access to the full bandwidth of the channel for a short burst by any source application. This is not something SONET ordinarily supports well. The standard SONET concatenation method can create certain clear-channel ("big pipe") bandwidths for Packet-over-SONET (PoS) or other unchannelized uses, but for some payloads, the nearest concatenated bandwidth is still fairly inefficient for the intended payload. One of the most important such mis-matches is that for Gigabit Ethernet (GbE). With 8b/10b coding for transmission, direct transport of the GbE physical layer signal requires an OC-48c. But this results in use of 2.5 Gb/s of transport capacity to carry the 1 Gb/s payload. An enhancement to SONET, called *virtual concatenation*, addresses this kind of situation, allowing SONET formats to provide a much wider family of contiguous (i.e., unchannelized) but previously nonstandard SPE rates to better match the actual payload rate requirements. The technique works by moving the concatenation capability, previously only implemented at the STS-1 level, down to the SONET Virtual Tributary (VT) level so a finer granularity of transport bandwidths can be provided. Because the resulting rates are non-standard, however, in the sense of previous SONET transmitter/receiver rates, such SONET virtual concatenation formats are conveniently also handled by the new Generalized Framing Protocol (GFP) technique. Three coordinated efforts in ANSI and ITU standards-making organizations have essentially solved the problems of using SONET for any number of different data transport applications. Additional details and references on these developments are available in [BoRo0] and [GhDe01].

2.5.1 Generalized Framing Procedure (GFP)

GFP is a standard way to adapt any frame-oriented packet data sequence or block-code data signal for transport over a suitably sized SONET SPE. Despite the suggestion by the simplified phrase "IP on optics" that IP packets might be directly applied to a laser transmitter, there are actually quite stringent requirements for an arbitrary bit sequence to be reliably transmitted and received over a physical layer path. There is always a need for some type of synchronous bit or byte timing for regeneration, certain transmission coding properties for receiver decision threshold control, an assured bit transition density for low-jitter clock recovery, and so on. A fundamental requirement is also frame alignment to correctly identify the frames or packet data units within the serial transmission stream. In their own ways these technical details are largely what each of the proprietary formats mentioned above are concerned with. In this regard GFP is an important compliment to SONET standards that provides a universal means to adapt any frame-oriented or byte-oriented data signal into a corresponding SONET SPE. Once in SONET, the non-traditional payload is suitable for almost unlimited distance transport on either single-wavelength or DWDM systems.

2.5.2 Virtual Concatenation (VC)

Virtual Concatenation is a technique to overcome the coarse basic channelization of SONET and allow SONET OC- n rate bandwidth to be allocated in multiples a single VT1.5 or STS-1, improving the efficiency of data transport with SONET. VC is based on inverse multiplexing techniques applied at VT1.5 or STS-1 levels between the path end-nodes. The respective transport capacities are referred to as "VT1.5- n " or "STS-1 n " where n is the multiplicity of the basic unit in the concatenated service. As an example consider the problem of directly transporting the 10 Mb/s Ethernet physical layer signal for a LAN interconnect application. Ordinary SONET methods would require mapping the 10 Mb/s into an STS-1 SPE dedicated for the purpose at about 21% utilization. Under VC, however, a concatenation of 7 VT1.5s, denoted VT1.5-7v, can be created. At about 10.5 Mb/s the VT1.5-7v container then carries the 10 Mb/s Ethernet payload at nearly 90% efficiency and the remaining 21 VT1.5s in the same SPE can be used for other applications, such as conventional DS-1 transport or similarly restructured under VC for other uses. The same principles apply at the STS-1 level under VC, where the 1 GbE payload provides an especially significant example. As seen before, 1 GbE would normally require an intact OC-48c, at about 42% utilization. Under VC, however, a concatenation of 21 STS-1s could be formed (the VCG is STS-1-21v) providing an almost perfect

capacity match (98% efficiency). Alternately, 7 STS-3cs can be virtually concatenated into a STS-3c-7v VCG operating at 95% efficiency. In both cases this allows a second entire GbE physical layer signal to be transported in the same OC-48, or any other payloads in the remaining capacity.

Virtual concatenation works differently than normal SONET concatenation. The STS-1 and the standard concatenations are still created and transported intact even though their bandwidth may be part of a VCG. For instance, an STS-3c is entirely routed and switched as a single 155 Mb/s rate signal throughout the network. VCGs are formed synthetically, however, by the transmitter breaking up the non-standard payload into portions mapped in parallel onto separate standard VT1.5, STS-1 or STS-3c SONET tributaries. At the VCG receiver the separate tributaries being "virtually concatenated" are compensated for any differential delay and the payload portions received in parallel over the network are reassembled for the single serial rate payload. Importantly, the transport network itself never sees any non-standard SONET rates or formats as a result of VC and the SONET line rates are not altered. The installed capacity is still comprised of fully standard fiber systems or lightpaths operating at, say, OC-48 or OC-192 rates and only standard rates continue to be cross-connected and rerouted for restoration. But now non-standard payloads are being efficiently managed over the available path capacity by the VCG processing at path end points.

[Table 2-1](#) lists examples of payloads that can be adapted by GFP for SONET transport, either in an existing SONET frame or concatenation or in the closest matching VC group (VCG) using virtual concatenation. In [Table 2-1](#) "frame mapped" means that the packet data units (PDUs) of the native application are identified and they alone are stripped out of the physical layer LAN signal and mapped into GFP frames. The rates provided support the same peak rates of data through GFP as in the corresponding LAN, but with buffering in the interface it would be possible to assign a GFP bandwidth closer to the actual mean frame or packet data rate. In other cases the physical layer bit sequences are mapped "transparently" into GFP frames for transmission. Frame mapping thus moves only payload while transparent mapping literally conveys the native physical layer signal without recognizing the application PDUs. Transparent mappings require more bandwidth but give the least delay.

Table 2-1. New SONET Payloads Accommodated by GFP and Virtual Concatenation

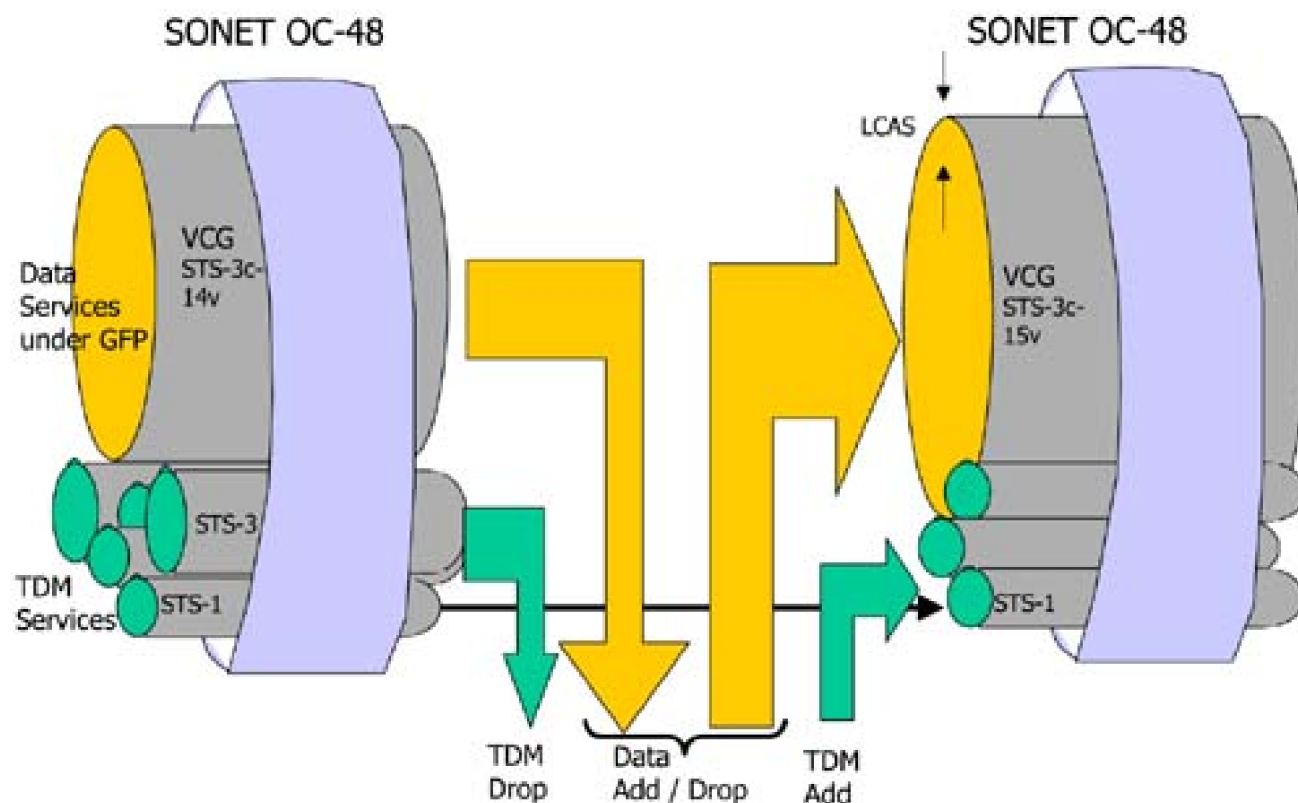
Payload or Application	Rate	SONET VC Group
Ethernet (frame mapped)	10 Mb/s	VT1.5-7v
Fast Ethernet (frame mapped)	100 Mb/s	VT1.5-64v
Gigabit Ethernet (frame mapped)	1000 Mb/s	STS-3c-7v or STS-1-21v
Fiber Channel or FICON	850 Mb/s	STS-3c-6v
ESCON	260 Mb/s	STS-1-4v
GbE Ethernet (phy 8b/10b coded)	1250 Mb/s	STS-1-25v
Infiniband (frame mapped)	2.5 Gb/s	STS-3c-14v
DVB-ASI (digital video)	216 Mb/s	STS-3c-2v
general purpose "bit pipe" at	160 Mb/s	SONET STS-1-4v
general purpose "bit pipe" at	425 Mb/s	SONET STS-3c-3v
general purpose "bit pipe" at	850 Mb/s	SONET STS-3c-6v

2.5.3 Link Capacity Adjustment Scheme (LCAS)

LCAS is the signaling protocol through which the two end-nodes of a SONET-formatted lightpath can dynamically manage and reconfigure the VCG configuration of the SONET path between them, allowing them to dynamically resize the bit pipes allocated to various data flows between them as well as accommodate variable numbers of TDM circuit connections. [Figure 2-9](#) portrays how LCAS (in conjunction with GFP and VC) opens up the SONET line-rate such as OC-48 or OC-192 to highly flexible and mixed use for both data and TDM services. In the example, an OC-48 coming in from the left has two STS-3 and three STS-1s configured for TDM traffic. The remaining OC-48

bandwidth is available as an STS-3c-14v providing a clear bit pipe of 2.1 Gb/s under GFP for statistically multiplexed access by any desired data applications. The GFP portion of the OC-48 SONET bandwidth can be used by any combination of PDU (frame-mapped) or transparent data signal flows, subject only to suitable traffic engineering. Outgoing from the site, on the right, the balance of TDM and data may be different, so the VCG is configured accordingly. In this case the TDM component has gone down by three STS-1s, so the VCG capacity is raised to STS-3c-15v. The LCAS protocol is used to make this adjustment. GFP lets any or all of the SONET SPE function as a transparent bit-pipe for pure statistical multiplexing of any type of packet data traffic and/or transport of the intact physical layer signal of various LAN interfaces.

Figure 2-9. How SONET flexibly integrates data and TDM using GFP, VC and LCAS.



There are at least two senses in which the GFP, VC and LCAS are not only relevant as background to work in transport networking, but are directly relevant to subsequent problems in this book. One is that the simple model of aggregation of total demands in determining the number of wavelength or OC-n paths required between node pairs is made even more directly applicable to network planning. With GFP, VC, and LCAS it is not necessary to forecast precisely by application type how many paths will be needed on each node pair, only what the aggregate of all requirements is expected to be. The second is that GFP, VC and LCAS make it especially easy to adapt the configuration of each OC-n link or path to meet time-evolving requirements. In other words, the feasibility of rearranging the logical configuration of already deployed transport capacity is enhanced, enabling the application of incremental growth and rearrangement applications of the planning models that follow. For more detail on GFP, virtual concatenation and LCAS see [BoRo02](#) [CoMa02](#), and the references therein.

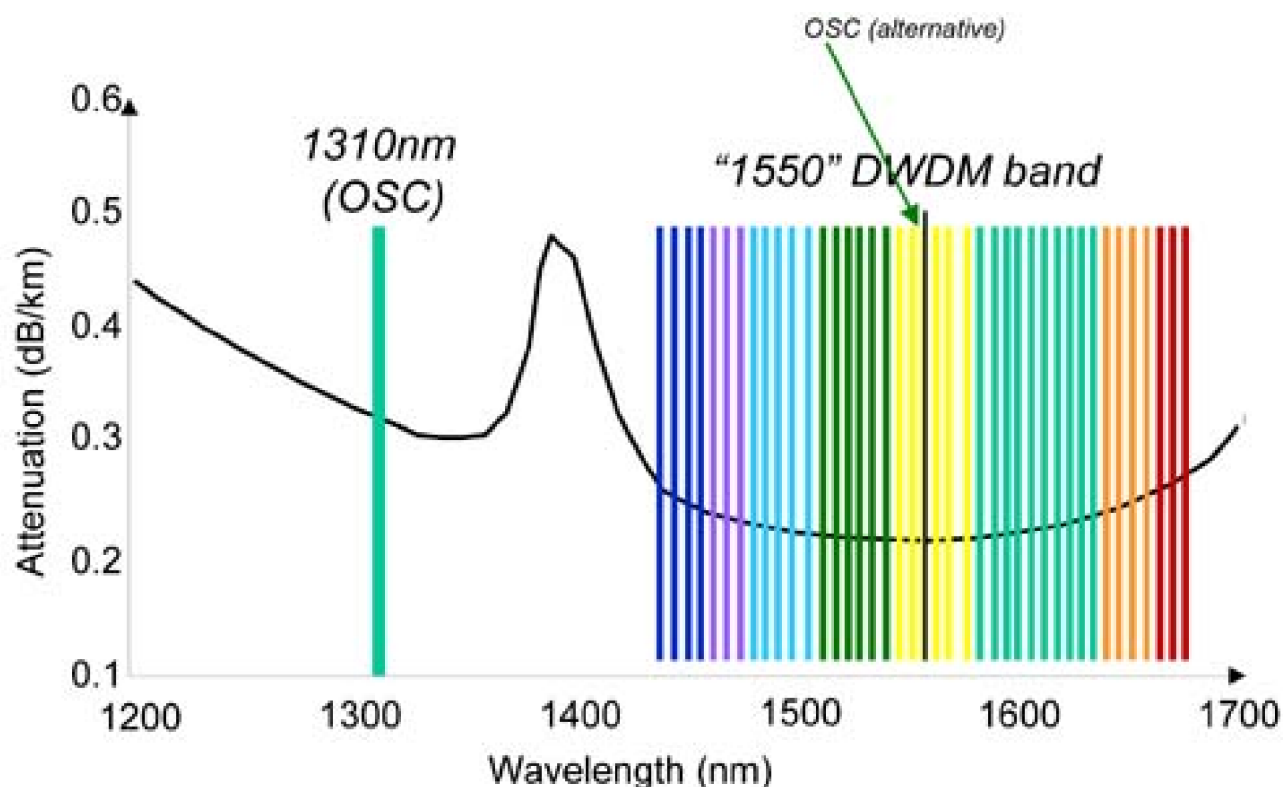
2.6 Optical Service Channels and Digital Wrapper

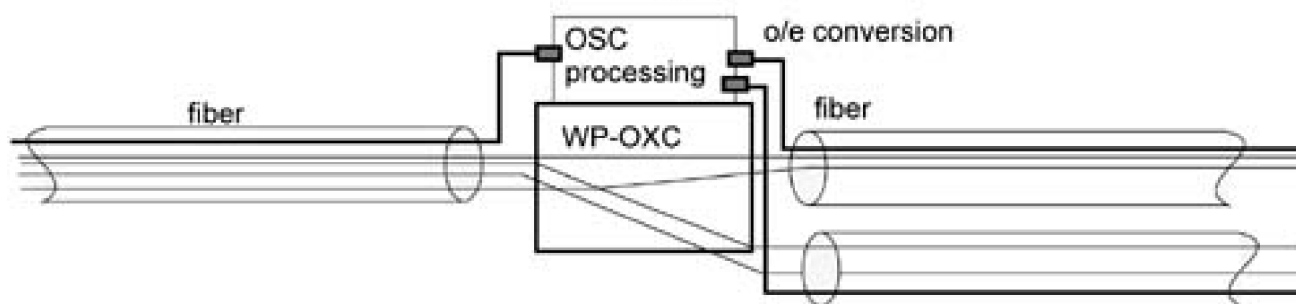
An early view of all-optical networking was that of a fully transparent network where lightpaths would follow the same wavelength assignment end-to-end without conversion into the electrical domain anywhere en route. Paths through the network itself would be completely independent of the rates and formats of any payload. Only the end nodes of each path would have to agree on the payload structure. But this also implies that only the end nodes can monitor the integrity of the lightpath and its payload. Such complete transparency does not turn out to be workable in practice. An important attribute of SONET that turns out to be equally essential in optical networking is to have some form of channel-associated overhead signaling. SONET section, line and path overheads facilitate fault isolation, protection switching, signal identification tracing, general purpose in-band data communication channels for remote operations, maintenance and control, and so on. These are enormously useful capabilities. Yet when we contemplate an OTN, how do we achieve the same kind of signaling functionality? Optical Service Channel and Digital Wrapper are two basic options to provide the required overhead capabilities in optical networks.

2.6.1 Optical Service Channel (OSC)

The Optical Service Channel (OSC) is a designated wavelength that bears all link-associated status and signaling information. Only this designated channel need be electrically detected and processed on every fiber link, allowing other wavelength channels to pass transparently through an associated WP OXC without the expense of leaving the optical domain. The OSC approach thus permits a network to stay in the pure WP mode of operation on all other bearer optical channels. The OSC concept is summarized in [Figure 2-10](#).

Figure 2-10. Optical Service Channel Concept: transparent optical cross-connection for most channels, opaque processing of the OSC for link related OAM&P functions.





Most vendors provide an OSC carrier and supervisory data interface function as a specialized laser transmit/receive card that feeds into the WDM mux/demux for the corresponding fiber. Thus, an OSC can be established as desired over any DWDM fiber span. The supervisory wavelength may be out of the DWDM band (at 1310 nm, say) or can be a designated wavelength within the 1500 nm band. Since this channel is fully terminated and processed at each node, it is not crucial which wavelength is used as the processed outgoing contents will always be placed on a new outgoing optical source. Adjacent nodes need only know which wavelength is bearing the OSC function. Some optical networking product lines have levered off the necessity of the OSC being present to use 1310 nm optics that are lower in cost and completely out of the DWDM band, but can still support an OC-48. The 1310 nm OSC channel is by default routed through all nodes of the network in a ring-like way and bears a standard OC-48 frame that is fully terminated and processed at every node. In this way the necessity of o/e conversion for the OSC channel is used to also establish an internal LAN using the SONET data communications channel (DCC) for communication from a control center to all nodes, as well as monitoring optical integrity of each link and so on. But the OC-48 SPE, itself a non-trivial amount of bandwidth, also serves to provide for a number of "milk run" IP flows or DS-1 requirements functioning as a logical ring of OC-48 add/drop multiplexers.

2.6.2 Digital Wrapper

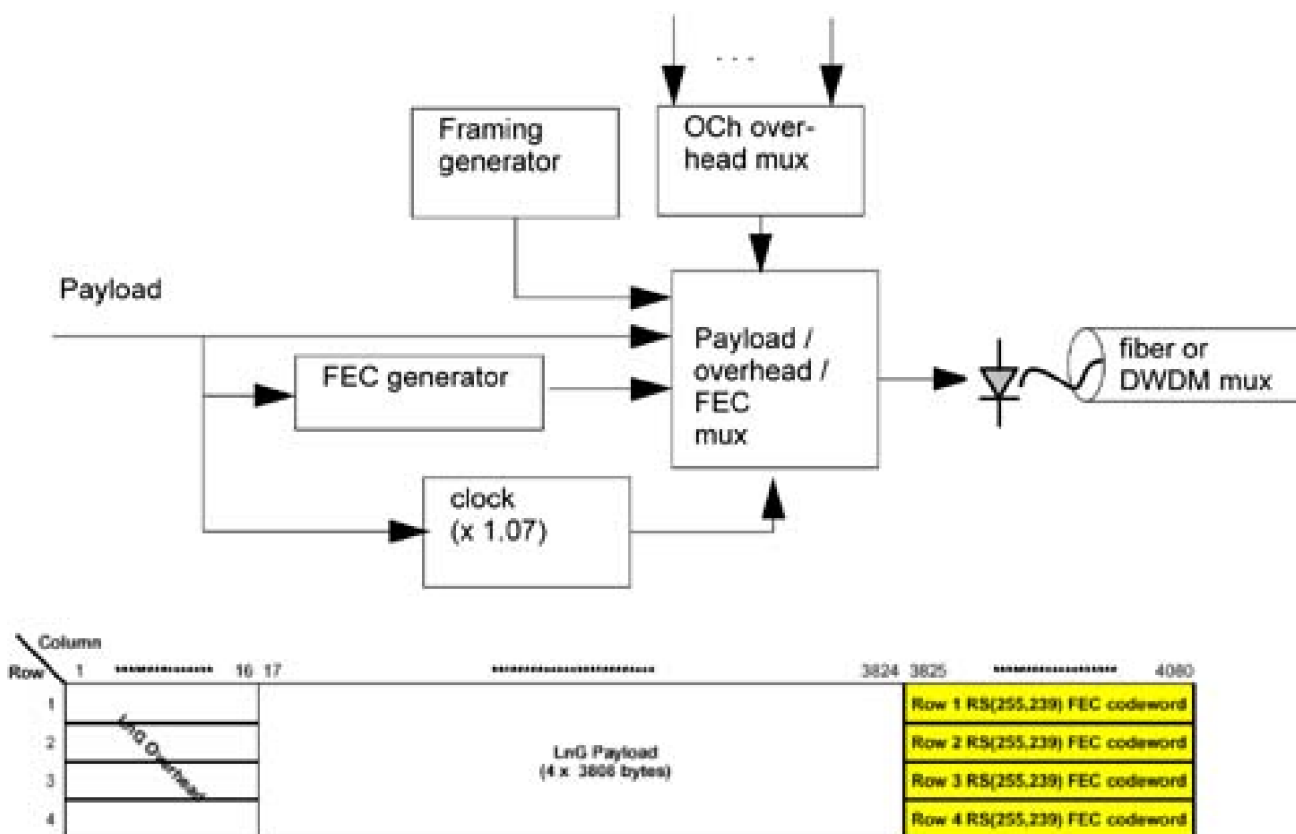
The Digital Wrapper (DW) approach [Brun00] is more SONET-like in the sense that every wavelength channel (not just one designated OSC per link) has overheads added to it. These overheads also adopt a section-, line- and path-like hierarchy and are meant to be detected and processed electrically at every node. Digital Wrapper is therefore much more oriented to opaque optical networks using o-e-o cross-connects. DW even implements some capabilities that were proposed but not adopted for SONET, most notably Forward Error Correction (FEC). DW also provides a framework for carrying high rate unchannelized packet or cell-based IP traffic, GbE or ATM cell flows directly over the optical layer of the network. Digital wrapper is sometimes qualitatively explained as "adding a wrapper around a wavelength." This of course has no direct physical meaning but expresses the concept that the wrapper helps safeguard the payload (especially since it includes FEC), and contains various other data about the payload itself. The notable thing is that DW is implemented in the electrical domain at the full payload bit rate as a specialized sequence of additional headers and trailers added to blocks of the payload bit sequence. The digital wrapper adds about 7 percent increase in the signal rate (relative to the payload rate alone) to include new overheads for restoration signaling, framing and FEC. The augmented electrical payload then drives a laser to put the composite "wrapped" payload signal onto a wavelength. DW recreates virtually all of the overhead signaling that SONET provided, but does so for wavelength channels and without asserting its structure on the payload. One of the most important features of DW that SONET never did incorporate is a standardized way for protecting any lightpath payload with FEC. FEC is a feature most manufacturers had implemented in some proprietary way in their OC-192 systems.

However, the DW technique for adding overhead is different from SONET in important ways. Whereas SONET defined the frame structure, frame timing, and overheads, requiring payloads to be adapted or "mapped into" the SONET SPE, DW reverses the relationship by conforming ("wrapping") itself to the payload in a more transparent way. Any payload data sequence will appear to see a clear serial channel at a rate that is ~93.3 percent of the clock rate selected to drive the laser. This allows arrangements where the payload rate and timing is not adapted into a set OC-n rate, but rather the payload rate is independent and the clock rate for the digitally wrapped composite signal is raised slightly as needed to add the DW overhead. This eases the otherwise complex process of synchronizing/desynchronizing a constant bit-rate payload signal with a custom SONET payload mapping structure. It also allows SONET-framed signals themselves to be payloads of the DW, in which case they obtain the benefits of FEC which SONET itself never provided. The overhead consists of Optical Channel (OCh) overheads preceding a block of payload data, followed by the FEC check bit data. The FEC scheme used is an adaptation of the Reed-Solomon (255, 239) FEC coding already developed for undersea fiber optic applications (ITU G.975) in which 16 bits of FEC check data protect against single byte errors in a 255 bit block (including the check bits). The composite DW signal that is applied to a wavelength is referred to as an LnGm "lambda signal" where n.m describes the aggregate bit rate. These conceptual aspects of digital wrapper are summarized in [Figure 2-11](#). Among the channel associated overhead capabilities that Digital Wrapper provides are:

- Path trace identification

- Forward and backward failure indications
- General purpose DCC channel
- Operator specific order wire channels
- DW framing
- Bit interleaved parity checking (BIP)
- Overheads reserved for future applications
- A push-pop data structure recording performance information at each section along a path
- Protection switching protocol bytes analogous to those in SONET K1-K2 bytes

Figure 2-11. Concept of digital wrapper for wavelength-path transport of arbitrary clear-channel payloads, but with rich overhead functionality.

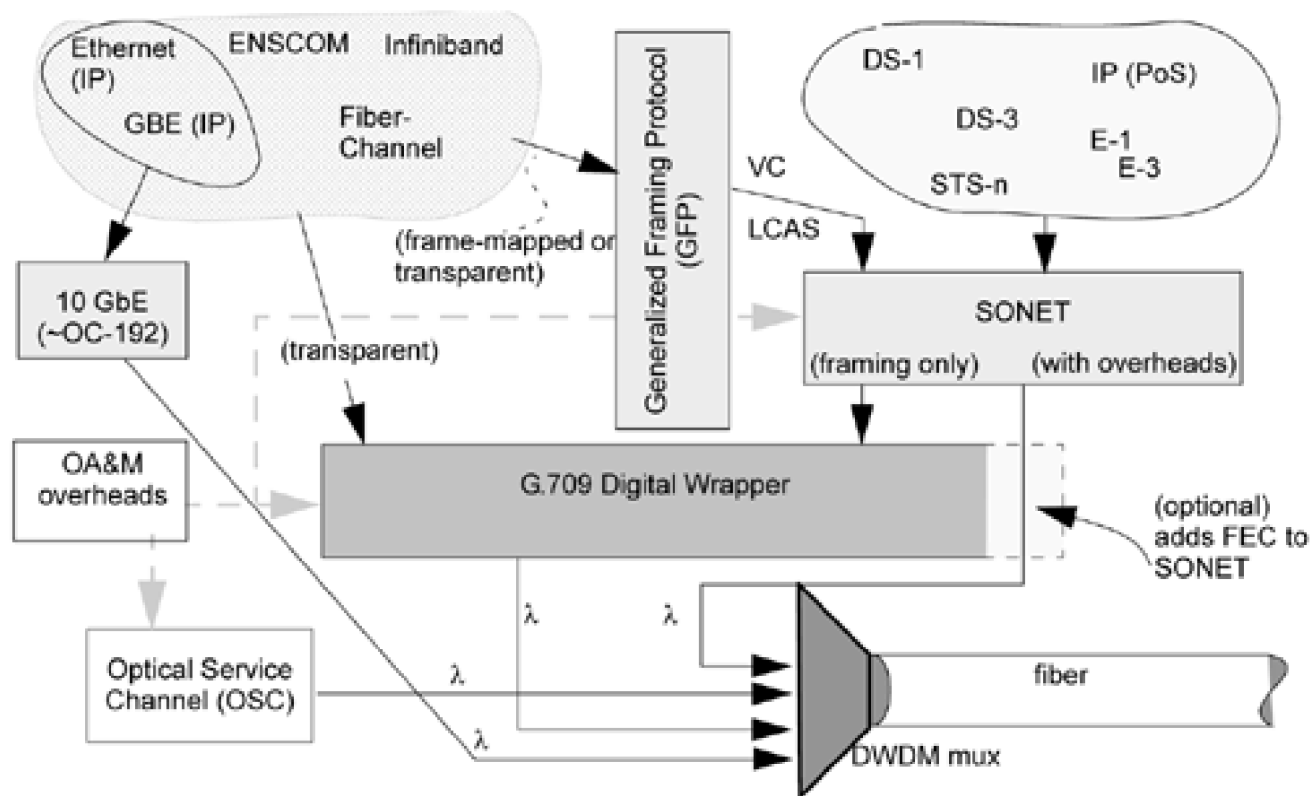


What is most significant about the advent of DW is that from a network planning and network operations standpoint an optical network with o-e-o cross-connects terminating and processing DW on every lightwave channel is essentially a perfect logical analogy to a SONET mesh network based on B-DCS nodes. First, wavelength assignment is not an issue; we need only address the pure capacity planning problem. Secondly, the channel-associated signaling overheads that DW provides allow the same or better capabilities in fault sectionalization and support for distributed restoration and a variety of distributed protection preplanning schemes such as follow in the book.

[Figure 2-12](#) gives an overview of how these new developments for data-centric payloads and optical network channel overhead schemes interrelate. The figure attempts to show all the ways a payload can wind up riding on a lightpath and the options for overhead signaling that may also apply to the lightpath. Still present for traditional PDH or SONET TDM payloads is the option of mapping into a normal SONET STS-n structure applied directly to a DWDM wavelength path. DW can optionally be applied outside the SONET frame if the benefit of FEC is desired. The clustering on LAN/SAN and IP-centric data payloads in the upper left can either be transported over existing SONET transport using GFP, in which case the added benefits of virtual concatenation and LCAS are available to efficiently size the bandwidth allocations of a SONET OC-N to a mixture of TDM and data flows. GFP allows the allocated VCG bandwidth to be used in a stat-muxed packet traffic mode or it can also carry these sources "transparently," moving the physical layer signal of the input intact. Alternately, if the bit rate or application warrant a dedicated wavelength path, the latter payloads can also be adapted for direct wavelength transport by

digital wrapper. In addition, a specific adaptation of OC-192 technology leads to a pure Ethernet transport option (10 GbE). In addition, Digital Wrapper, basic SONET framing, and OSC all provide options for network status monitoring, fault location and implementation of mesh-based survivability schemes or signaling for topology and resource discovery. All this goes hand in hand with the capacity design methods that follow, as these are the basic ways through which capacity, for both working and protection, can be managed, monitored, provisioned, configured, and cross-connected for mesh-based survivability.

Figure 2-12. Options for TDM and data transport over an optical network.



2.7 IP-Centric Control of Optical Networks

We have already seen the importance of IP as the dominant type of traffic, and how this leads to new transport techniques such as 10 GbE and GFP for SONET. These are ways in which IP has influenced the *data plane*. But another significant influence on transport networking has been the transfer and extension of ideas and techniques used for topology discovery, routing, and virtual circuit establishment by the Internet *control plane*. This section is devoted to an overview of the key concepts and techniques through which an optical transport network may be managed and controlled through adaptations of IP-network routing protocols. The overall approach is usually referred to as "IP-centric control" of the transport network [BeYa00]. In [Gers00], Gerstel compliments that article with an interesting discussion of just what combination of pure Internet-style signaling and centralized network management to use. Also recommended for background in this area is [McBo98].

Transport networks have traditionally been centrally controlled in terms of provisioning (and taking down) service paths as needed. Transport path requirements grew or changed relatively slowly and centralized, even semi-manual, operations control was adequate. In contrast the circuit-switched voice network handles major changes in terms of its internal circuit configuration every minute and has therefore been highly automated in its call by call circuit establishment and tear-down functions. Although under overall supervisory monitoring by network management, individual call routing evolved to become a highly distributed and adaptive decision making process handled among the switching nodes themselves.

One view of the future in transport networking is that high-end applications and major source/sinks of packet flows such as router nodes will "dial up" lightwave paths on demand and release them again, perhaps only minutes later. It is at least debatable, given that a lightpath supports 10 to 40 Gb/s, when the transport network would see lightpath demand that is truly as random in its arrival and departure times as connections in the voice switched network. It seems more practical and likely that lightpath requirements would often be amenable to some form of scheduling or advance notification of the upcoming lightpath requirements. More frequent random changes can be expected from applications and client networks that generate requests for STS-1, STS-3, GbE levels of connection or capacity augmentation. As with the airlines, there is always a discount for customers that pre-announce their plans. The truly random arrival pays the most. In addition, scheduling or batch forecasting is plausible because the optical layer path requirements have primarily only to respond to the changes in the aggregations of such lower rate service connections. Nonetheless, whatever characteristic time scale is envisaged for changes in the OTN connection state the question becomes: How can the OTN autonomously establish and take down requested lightpaths within itself, efficiently, and on an ongoing basis, driven only by the "user" demands, without centralized control on a call by call basis?

The vision of a transport network operating in this "self-organizing" way, using simple interaction between cross-connect nodes to establish and remove service paths as needed, was researched in the context of SONET-based mesh networks in [GroV89, MacG91, GroV97]. The approach then used for distributed nodal signaling interaction was, however, an extension of K1-K2 state byte signaling methods used for restoration path finding in a mesh network. Today a more general and expandable paradigm for distributed cooperation between nodes is provided by explicit topology discovery and routing protocols used on the Internet. Note that using extensions of these widely known protocols for *control* of an optical network should not be confused with meaning that all transport is also by IP routing. In other words, it is the *control plane* of a transport network that we consider here, not the *data plane*.

2.8 Basic Internet Protocols

The needed background is an overview of the key protocols for Internet routing and how these are extended for dynamic provisioning in an OTN. In later chapters this material will help readers appreciate some of the options for implementation of mesh-based survivability, in particular how IP-centric protocols can be the basis for setting up certain types of path protection arrangements, how MPLS layer restoration with oversubscription works, and the feasibility of various types of networks that are self-planning against failures based on global topology knowledge. To avoid confusion in what follows, note that in the suite of basic Internet protocols, the view of the network is purely topological. A "link" either exists or does not between two nodes. Its capacity is not described or resolved although it may have an associated routing cost. This is of course quite different from the transport network view where every span has a specific capacity that must be taken into account in routing, etc. It is in the extension of the basic IP protocols that a capacitated view of network resources becomes available for use in transport network applications. For more through treatments of the basic Internet Protocols see [\[Moy98\]](#), [\[Huit95\]](#), [\[Stev94\]](#), [\[Perl92\]](#).

2.8.1 TCP

TCP stands for transmission control protocol. It is the basic protocol to create and manage logical connection-oriented services over a connectionless IP routing layer. TCP provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with retransmission of packets when necessary. TCP uses a windowed transmission protocol and slow-start fall back when it encounters errors or congestion, to try to manage connection throughput. TCP also multiplexes many applications between hosts or clients by managing port numbers that label each TCP/IP packet to maintain separate associations between different communicating process pairs between the same two computers.

2.8.2 OSPF

Open Shortest Path First (OSPF) is a routing protocol used to determine "least cost" routes for packets within IP networks. The cost criterion is an arbitrary link weight that administrators associate with each link adjacent to OSPF nodes. OSPF is a fundamental protocol to the Internet and to IP-centric lightpath routing because as part of its operation it discovers the entire network topology. The basic sequence of operations performed by OSPF routers is:

- Discovering OSPF neighbors
- Forming logical routing adjacencies
- Synchronizing databases
- Calculating the routing table
- Advertising Link States

Routers will go through these steps when they first come up, and will repeat these steps in response to events that occur in the network. A router must perform these steps for each subnetwork it is connected to, except that it generates and maintains a single integrated routing table even if connected to several networks. A router employs the following subsidiary protocols and processes for its overall implementation of OSPF.

"Hello" Protocol

Every ten seconds (nominally, but this can be changed) a router sends a hello packet to each immediately connected (logically not physically) adjacent router on each link that is in operation on one of its ports. As long as the adjacent router answers, the link is considered in an "up" state. If four "hello" packets are sent without an acknowledgment, the link is determined to be in a "down" state.

Designated Router Status

When a new router is connected to the network, or a new link added, each pair of adjacent routers goes through a process based on router ID number ranking or assigned priorities to determine which will be the "designated router" with respect to the link connecting them. The priority arbitration to determine this is supported in the "hello" protocol. The Designated Router then has the responsibility to generate any link state advertisements (LSAs) for the link.

Link State Advertisement

If a new link is established in the network, or if a previously operating link goes down, the designated router for the link sends out LSA packets that are relayed by all other nodes creating a simple flood to disseminate the link state change information. The LSA packet flood is killed off by a time to live measure, but this is assumed high enough so that all nodes in the domain receive the notification.

Link State Database Synchronization

After a neighbor has been discovered, bidirectional communication ensured, and a Designated Router elected, Database Description packets are exchanged between adjacent nodes. These contain a summary of the sender's link state database. Multiple packets may be used to describe the database using a poll-response procedure with the router with the higher ID being the master. Database Description packets sent by the master (polls) are acknowledged by Database Description packets sent by the slave (responses). Once a router has a local copy of the complete network topology, each router tracks changes in the overall network topology through LSA packets received (announcing up/down changes in link states). But when a router first comes on-line it can request the entire topology database from a neighbor router to get itself started. In all cases, however, before a router can update its routing tables it must make sure it has the identical global topology database as all other routers in the same OSPF domain are using. The process of achieving this state is called synchronization of the Link State database.

Routing Table Construction

When all Link State Request packets have been answered, the databases are synchronized and all routers have the same global network topology view. Each router then locally runs a shortest path tree algorithm (typically Dijkstra's algorithm in [Chapter 4](#)) to build its routing table from the link state database. When the algorithm produces multiple equal cost routes, OSPF can distribute the routing load across them evenly. Each router's derived tree encodes the least-cost route to take from its position to each other node in the network. Note that a router has to complete a complete resynchronization of its link state database before it can update its routing tree.

OSPF is called "open" because it is an open standard, not related to any particular vendor. The above description is only an overview of

the principles. John Moy's book on OSPF [Moy98] is recommended to delve further. The complete set of variations, special cases and options in OSPF is extensive. The beauty of OSPF is that it provides a complete global topology view of the network in which the node is present. Most of the time this view will also be in the synchronized state, although there may be significant time-lags. As can be appreciated, however, many distributed control and routing strategies become feasible if one assumes a reliable OSPF service running on network elements to provide a global network database. In principle any network element running OSPF has as complete a network view as a traditional control center. The price for this is that OSPF tends to be processor and memory intensive and its mode of operation based on LSA flooding can be associated with "routing flaps" which are periods of time in which the topology view and routing states are unstable or not current. If one link is intermittently up and down every few seconds, or in complicated states of a router reboot combined with a link failure, or if a physical layer cable fails and many IP layer links suddenly fail, OSPF updates can dominate the network traffic trying to inform every other router of the state changes and resynchronizing all databases to update routing trees as needed. Nonetheless within the confines of the control plane of an optical network (as opposed to the public network itself) where relatively generous overhead channel capacities are reserved for its use, OSPF may be more stable and keep every node relatively quickly aware of any changes in topology, or with extensions to follow, of any changes in capacity and resource usage on each span.

2.8.3 Multi-Protocol Label Switching (MPLS)

The basic technique of label swapping or label switching and its advantages were described above in [Chapter 1](#). MPLS is a way to accelerate and simplify the packet-forwarding process by establishment of virtual circuit-like label switched paths (LSPs). The relevance to optical network control is not in packet forwarding itself, but rather that the distributed protocols that MPLS defines to determine a route and lay down label associations to establish an LSP can be fairly naturally extended for distributed creation of desired paths through an optical network as well, including aspects of wavelength or time-slot assignment and routing diversity. The most current extensions and updates to MPLS (and its generalized form GMPLS) can always be found in the form of "Internet Draft" working documents at www.ietf.org. The number and detail of such documents can be somewhat overwhelming, however, so the book [JaRe00] is also recommended.

Label Distribution Protocol (LDP)

So how are the desired label sequences laid down in the LSRs to effect the LSPs that we want? One possibility is to precompute and centrally download a set of label swapping tables that actually constitute a full logical mesh on the set of nodes in the MPLS domain. Even for a 100 node network this requires only 9,900 (unidirectional) LSPs and label-swapping tables at each LSR that have no more than 100 entries per port. But more generally we want a more scalable solution and some means of requesting and releasing LSPs on demand. This is the role of LDP. For MPLS to function correctly each connected pair of LSRs must have the same interpretation of the labels used to forward traffic. This is achieved by LDP by having LSRs inform one another of the "label bindings" they have made. Although the LDP establishes LSPs, it is itself a protocol that operates via normally routed TCP/IP packets and has more than one implementation, including having its packets piggyback on those of some other protocols. There are also two basic approaches to LDP.

In basic or generalized LDP, LDP sessions are established between connected LSRs to exchange their label spaces. In this approach there is no explicit mechanism to direct the creation of end-to-end LSPs. Rather this happens indirectly as a result of the local actions of each LSR to find a label mapping with which to replace normal routing decisions for packets that arrive on a given label for which no forwarding label is yet established. To illustrate, assume that an LSR X knows the existing label space of its neighbors, A, B, C, D. The label space indicates the outgoing port to which that node will forward any packet arriving on a certain input port and label at its site. (Such label spaces are built by the node first observing the outcome of a normal routing decision for such packets using the OSPF-built routing tables.) Now consider two possible packet arrivals at node X, from node A, as an example. The first case conveys the idea of how a local label space at the node is developed. The second case will convey how LSPs are implicitly created as needed to move packet flows from normally routed handling to LSP-expedited handling.

In the first case, assume that a normal IP packet arrives at the LSR from node A. That is, it arrives on the port from node A not having an MPLS encapsulation and label. One option the LSR has is to make a normal IP routing decision and forward the packet in the conventional way. Since this packet has arrived without MPLS encapsulation the normal routing decision has to be made in any case. The result of the normal routing decision is which output port to send the packet to. Let us say this is to send it to node B. Having made this routing decision, however, the LSR is in a position to add this destination IP address to the list of all IP destinations associated with output port B (i.e., the forwarding equivalence class (FEC) for port B).^[2] Once this is done, subsequent packets on that IP address are automatically sent to port B by virtue of appearing in the FEC for port B. But more importantly, on the next LDP session with adjacent

node A, node A will learn of this forwarding policy established at B and can therefore, henceforth MPLS-encapsulate such traffic with a label that is in effect a shortcut to the routing decision for node X. In effect the label says: "node X, when you get this packet, switch it right to your port to node B." This shows how label space is generated at a node, and then when disseminated to neighbors, effectively shifts flows from normal routing to faster label-switching treatments. Another way to interpret this is that each new FEC-to-label binding that a node A learns that its neighbor X is maintaining is like an additional virtual output port at node A. In fact functionally, the label assigned at node A is the entry to a virtual circuit directly to the destination.

^[2] FEC as an acronym for *forwarding equivalence class* in the context of IP routing should not be confused with FEC as an acronym for *forward error correction* in digital transmission or data storage applications. Both are formally established acronyms in use within their respective fields.

In the second case, we imagine an MPLS-encapsulated packet arriving in the port from node A at node X and node X finds that it does not have a currently established label-switching entry to forward this packet via label switching. It does, however, have the label spaces of its neighbors via LDP sessions. All node X has to do, therefore, is forward the packet to the port indicated in the MPLS label (say this is label T), then look up the label associated with that IP address in the FEC of neighbor node T (let us say, this is K) and create a new label switching table entry recording "if from X on label T, switch to port T, with label K." The same process applied at all nodes inherently builds label switching tables that indirectly define a whole fabric of LSPs. The key to making this work is LDP for reliable and timely exchange of the FEC-to-label mappings. FEC-to-label bindings can also be made to dissolve with time so that the current set of implicit LSPs tracks actual traffic flows over some time scale. This is referred to in general as a "soft state" mode of operation.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

2.9 Extensions for IP-Centric Control of Optical Networks

We can now look at a number of enhancements to adapt existing Internet protocols for direct application to the task of dynamic path setup and tear-down on a "call by call" basis.

2.9.1 OSPF-TE

OSPF-TE [[KoRe00](#)] refers to the extension of OSPF to have awareness of link capacities and other technical attributes of the links in the topology seen by the routers. Generally in the Internet engineering community the term "traffic engineering" is used to refer to a context where the actual capacity of a link or a path is taken into account. Basic OSPF views the network purely as a simple-graph; either a link is present or it is not. So adding the concept of a link having a specific capacity to be respected in routing, etc. is the reason for OSPF-TE. The key change is a new type of LSA that carries much more information about links and the available ways of accessing bandwidth on each link through its end-nodes. The resulting network view obtained in each node becomes a complete global capacitated view of the network.

With an OSPF-TE database, paths can thus be computed by any node as an originator of a new path using various criteria, often different from the shortest path, to route around a failed link or congested link or to establish a disjoint backup route, for example. OSPF-TE can also support the setup of LSPs that are capacity aware (and hence performance managed) to support service level agreements (SLAs). With TE information it will also be possible to set up backup LSPs as a protection arrangement, with knowledge of the number of other primary LSPs also making backup arrangements over such links, and even with consideration of shared-risk link group (SRLG) information that indicates the mapping of common physical structures into logical links that have a correlated failure risk. The TE-LSA describes the link's transmission resources and current usage in terms of:

- A list of Shared Risk Link Groups to which the link belongs.
- The maximum bandwidth and available bandwidth on the link.
- An assigned link usage cost.
- The basic unit of bandwidth management on the link.
- An assigned LSP setup cost.
- A current measure of the total oversubscription of bandwidth reservations on the link.

A TE-LSA also includes information about the switching level capabilities of the end node of the link issuing the LSA. This includes indications of whether the node can provide:

- LSR capability
- LER (label-switching edge router) capability
- Basic Packet Switch capability
- STS cross-connection
- Wavelength channel cross-connection
- Fiber level cross-connection

One issue with TE extensions to OSPF is the frequency of LSA updating to disseminate changes in capacity usage on links. Advertising the entire link state in response to a unit change on any single link could be excessive in terms of flooding the network with TE-router LSAs. The TE *incremental*/link update LSA is therefore a smaller packet that advertises only incremental link updates and may be issued only when link capacity changes by a preset threshold amount.

2.9.2 Link Management Protocol (LMP)

LMP [LaMi01] is an IP-based protocol to handle issues that arise when the logical link between nodes is actually a channelized optical line transmission system, as opposed to a single bit-pipe link model in the original IP networking paradigm. LMP runs as an ongoing session established between each pair of directly connected OXC nodes over a mutually agreed choice of one of the available overhead channels on the optical span. This may be either a predefined OSC or one of the general purpose overhead channels provided by digital wrappers on the lightwave channels. LMP is primarily concerned with making sure that OXCs at each end of the span have coherent channel numbering schemes and summarizing the current status in terms of available capacity and other properties of the span for efficient dissemination in TE-LSAs to support capacity-aware provisioning processes. LMO also detects and isolates failures and generates alarm information from the span to the host OXC. More specifically, LMP functions are:

- Link Verification: Procedures for the two ends to determine the logical mapping of link ends between themselves. (This is analogous to using a tone sender to "buzz out" the mapping of wire pairs at each end of a twisted pair cable.)
- Link Summarization: Procedures to synchronize the span-information view for each associated OXC and produce summary LSAs that describe the span as a single overall TE-link from a network-wide view.
- Link Group ID: To reduce control traffic during failures, the alarm status of all channels is organized as a single link group ID.
- Shared Risk Link Group Identifier (SRLG): Identifies which SRLGs this span is affected by. This information is manually configured by the user to support diverse route path computation under backup path type protection schemes to be discussed later.
- Bit Error Rate (BER) Estimate.
- Optical Protection Span-Switching: This is an indication of whether the optical line includes a same-span protection switching arrangement to protect against single channel failures or dedicated diverse routed 1+1 protection switching arrangements. This information can be used in service provisioning to influence the restoration or protection arrangements that might otherwise be put in place.
- Total Span Length: This is manually entered configuration data that may be used as a routing metric or to estimate delay.
- Fault Detection, Localization and Notification functions: LMP includes a very frequent lightweight type of "Hello" protocol to serve as a constant monitor on the span's continuity and gets an assured failure notification to the attached OXCs in the event of a failure. If the OXC is an *o-e-o* type it may immediately detect the failure on its own as well, but in a purely transparent OXC, the LMP notification may be the only source of alarm activation to initiate OXC-based restoration or protection. LMP also provides support for interaction with optical line terminating systems for section-level fault location.
- Alarm Management: To suppress cascading and/or spurious alarms during normal connection procedures.
- Trace Monitoring: To allow an OXC to request that unique marking code be applied to the overhead bytes of a certain channel, to support applications such as physical path trace audits and other tests of the optical line system continuity and integrity.

2.9.3 MPIS and Generalized MPLS (GMPLS)

What has been called "MPIS" is based on the analogy between an LSP and a lightpath. Both are generically "switched" path constructs

in that once the switching relationships are set up, the result is an inescapable sequence of relaying actions that direct any input to the predetermined output in a completely circuit-like way. Thus MPIS is just MPLS where the label space of each span is simply the set of wavelength channels on the span. The analogy is that the completely predetermined sequence of relaying actions that will route a packet over an LSP established on a set of LSRs is just as circuit-like and hard-wired as is the route of a bit stream applied to a wavelength path cross-connected through a set of OXCs. The only difference is a technical one having to do with the switching fabric involved. Instead of label-swapping relating each input to output of an LSR, the physics of optical cross-connect switching relate each input wavelength channel to the corresponding output once the path is set up. In MPIS the available wavelengths of each fiber thus are called the *implicit label set* that is available allocation at the adjacent OXCs when those OXCs are viewed (from a control-plane standpoint) as if they were LSRs.

But it is just one more conceptual step to realize that *any* transmission channel resource can be viewed as an implicit label. Then the same processes that set up an LSP in an MPLS network can be used to provision entirely general paths through a transport network. If fast enough, certain forms of fast reprovisioning may even be considered for restoration purposes. [DoYa01]. This is referred to as *generalized MPLS* (GMPLS) [GMP01]. In GMPLS any unit of allocation, of any type of physical transport resource, is thought of as a *generalized label*. For instance, in GMPLS a router port ID, an ATM VPI/VCI, an STS-1 VT1.5 number, a fiber, a waveband, or a wavelength on a fiber can all be thought of as labels through which label-switched paths can be established. For cases that we would otherwise just recognize as circuit switching or cross-connection, the actual "label swapping" mechanism is the physical switching process for the respective medium, and the labels are the identity numbers of the respective channels, timeslots or fibers being switched together. The physics of light reflection in a MEMs array is thus re-conceptualized as a "generalized label" swapping mechanism.

This is much more than just an interesting analogy, however. The practical advantage is that by identifying and treating the ID numbers of all different transmission resources as labels, we allow all forms of path setup to follow fairly simple extensions of the same logical process and protocol implementations, and use the same form of databases, as MPLS. The extensions include mechanisms to support hierarchical and type-consistent label association at each node. Type consistency is obvious: a label of type {lightwave channel} cannot be connected to a label of type {STS-1}, but could be connected to a label of the same type or of type {fiber} or {waveband}. The sense of hierarchical label distribution is illustrated, for example, by establishment of a new DS-1 layer path, that may need to trigger creation of a new STS-1 layer path and, if yet further needed, also trigger a new lightpath. But in all cases the process will first try to "pack" the new path into any already commissioned paths at each level.

Label Stacking

GMPLS will also inherit the label-stacking feature of MPLS. This allows a kind of en route aggregation of many small granularity LSPs into one high capacity LSP over a common segment of their routings. In GMPLS this means, for example, that several lightpaths sharing different end-nodes, but which are co-routed over several spans in common, may efficiently be merged into a single waveband path (i.e., an LSP whose granularity is a waveband) over the common segment. This is just an example, but it is easy to see how this could be advantageous because it can permit lower-cost transport via OADMs with a waveband pass-through feature. In GMPLS, where labels can represent physical channels, fibers, wavelengths etc., label stacks allow a finer granularity of traffic classification between service path ingress and egress nodes (that may be at the STS-1 level, say) than is visible to the "LSRs" in the core of the network (which are really OXCs optimized to handle only whole wavelengths). Label stacking allows this by "pushing" down the incoming label of any incoming packet and adding a new label that switches the flow into an established high capacity transport LSP. Since, at any node, the combination of input port and label uniquely defines the path and implicitly the egress node of the LSP, it is possible for a backbone router to recognize the routing equivalence of perhaps a thousand different end-to-end LSPs, and push a single new label down on all their stacks so that their aggregate flow follows one "backbone LSP." At the egress of the transport LSP, the label stack is popped and forwarding continues based on the lower level label.

Label stacking also makes it possible to route through various regions or network domains where the entry and egress points are known but the internal routing details of the domains are not. In other words, entire subnetworks can be abstracted to single hops in the end-to-end route. Virtual Private Networks (VPNs) can also be supported by the same mechanisms. Consider an end-to-end LSP involving different network operators. It may have the logical structure: {node X to Y to K (in my own network)} {across other network Z to node P} {node P to B to M (in my own network)}. In this case the first level label would relay the traffic through nodes X,Y,K and then obtain a *next-node = P* label reflecting the logical jump right across the intervening "other network" for which routing details are not known. On entry to the intermediate network the first level label sequence is "pushed" by the ingress LER of the intervening "network Z," and a new label applied that is internally known in network Z to be an LSP to node P in network Z. Once traversal of the intervening network is finished, the stack is "popped" at the egress LER. The popped stack then has a label "to B" sitting there ready to continue, just as if the entire middle network had been only one hop at the first level of label-switched routing.

2.9.4 Constraint-Based Routing Label Distribution Protocol (CR-LDP)

The basic label distribution protocol (LDP) used for MPLS in the Internet was described above and we noted that it implicitly defined a set of LSPs from any node to each other node through sharing of label spaces with neighbors. But these label spaces were generated in each case from a first observation of a normal routing decision for packets with various absolute destination addresses and therefore inherently reflect only shortest path routes. And in all cases there is only one implicit LSP from each node to each other. CR-LDP is a signaling protocol for use with GMPLS that shares generalized label spaces but also allows nodes to originate LSPs over explicitly requested routes and of explicitly requested capacity. In contrast to the basic LDP, where LSP routes are only implicitly defined, a CR-LDP signaling sequence follows an explicit route requested by the initiator node, through to the terminator node of a new path to be established. CR-LDP is thus *constrained* in two senses:

- The requested path will be set up over an explicit route chosen by the initiator of the path request. A key use of this is to explicitly load-balance flows over a network or to establish a working and backup path pair with explicit control of the diversity between them.
- The assignment of (generalized) labels on successive hops of the path may also be constrained through the label-set attributes of CR-LDP packets. This allows paths of specific capacity and other attributes to be set up; for instance an STS-3c path using only links that are span-protected.

CR-LDP is designed to be used in conjunction with a network that is running OSPF-TE. In this context an initiator node for a new path has visibility of the available capacity resources, granularity, and available modes of access for each TE-summarized link in the network. (In an optical network the TE-link is essentially the same as what we call a span. That is, a logical link between OXC nodes, but which has specific attributes of capacity, channelization, length, multiplexing granularity, and so on.) In other words, the initiator has an essentially complete global view of the network and can therefore locally solve a centralized routing problem to arrive at a desired routing solution for the new path. Using information in the TE-link database, such paths may be solved for on the basis of minimum cost, minimum length, maximum availability, or other criteria, with the important difference from prior routing protocols in that the selected route is also known to be feasible capacity-wise for path establishment. CR-LDP is described further in [\[CRLDP\]](#).

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

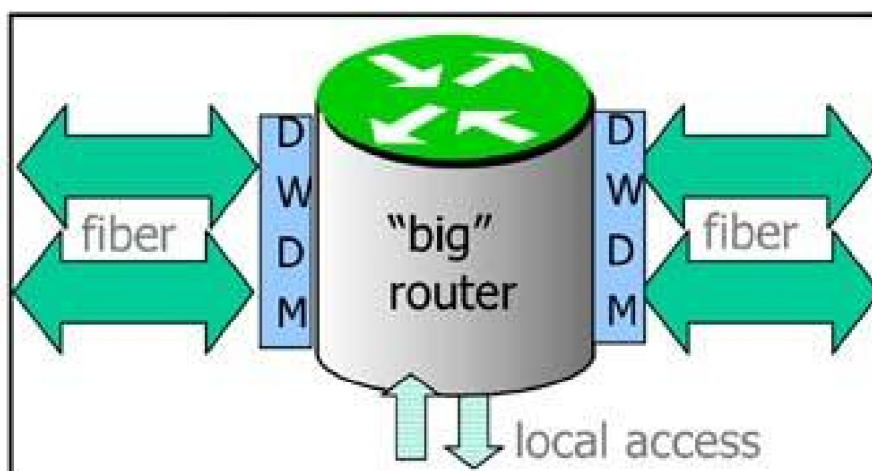
2.10 Network Planning Issues

2.10.1 Does IP-Centric Control Also Imply an "All-router" Network?

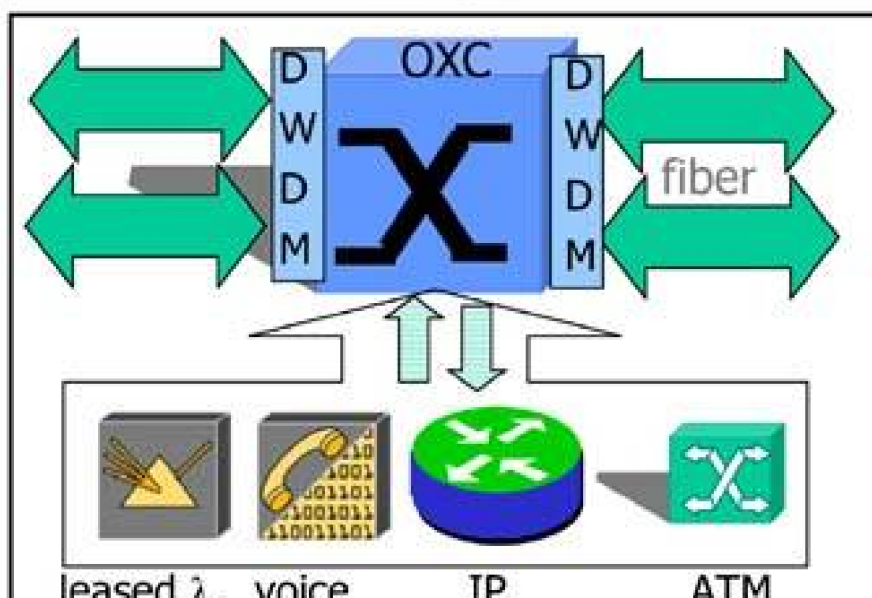
By collapsing the number of layers, and adopting an IP-centric *control* paradigm, should we assume this means that routers will also be the basic nodal element in the future transport network? This is referred to as router-centric transport and is different than IP-centric *control* of an OXC-based transport network. In the latter view, routers, private networking nodes, leased lambda services, and any other services needing transport path services are *clients* of the transport network. Advocates of the router-centric approach refer to the concept of *smart routers, simple optics* (SRSO). The opposite viewpoint can be thus also be aptly described as *smart optics, simple routers* (SOSR). An extension of the SOSR philosophy is the concept of a "transport-stabilized" Internet. The two paradigms are summarized conceptually in [Figure 2-13](#).

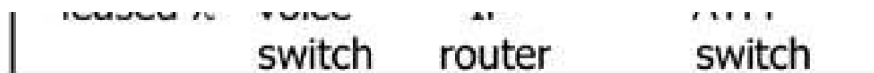
Figure 2-13. Two different views of future transport networking.

(a) A terabit router sits in a sea of point-to-point capacity



(b) An optical cross-connect serves capacity to client devices and service networks





[Figure 2-13\(a\)](#) shows the view of pure router-handled IP transport. There are no OXCs or OADMs. All client services of the IP backbone transport network interface at an IP packet level to the backbone router node. Point-to-point unprotected optical links interconnect these super-routers directly and the router looks after all connection management, QoS, and protection or restoration needs. Advantages of this architecture are the uniformity of how all services and traffic are handled and the capacity efficiency that arises from re-consolidation of traffic onto each link leaving the node. This allows for the highest utilization of the point-to-point links between routers. On the other hand, because the physical network graphs are typically rather sparse, router-centric transport implies that there will be more router-hops in the average path than with an OXC-managed optical layer that can create numerous direct inter-router links. In addition, most of the packet flows forwarded through such backbone routers will be transiting traffic.

As network traffic grows, router-centric transport also requires continually higher capacity routers and there are definite limits to the size and speed of an electronic router in terms of space, power and reliability. Power consumption limits of 7kW for a 7-foot equipment rack are already a dominant limiting factor in the design of high capacity routers. Routers are also extremely software-intensive and highly dependent on accumulated network state information. In this regard it seems ironic that the number of lines of software in a conventional digital circuit switch was often pointed to as a key reason explaining how unmanageable, and ultimately expensive these systems had become. But it seems reasonable that the amount of software and database dependencies in the "big router" model would be at least as great if it was to control all service layer functions and all of the logical transport constructs they require simultaneously. The router-centric architecture is thus likely to be most effective for private networks that are not mission-critical and not of extremely high capacity.

For carrier-grade networks where extremely high availability and scalability for growth are paramount concerns, the "smart optics" architecture in [Figure 2-13\(b\)](#) has more advantages. Operating below the service layer, OXCs can provide flexible optical pipes and much faster restoration against physical failures. The optical network layer also decouples the router connectivity from limitations of the physical graph, providing a wavelength networking layer that can richly connect the routers, reducing the load on each and reducing average packet delay throughout the network, as well as permitting smaller, less costly routers. Through coordination among OXC nodes, using communication overheads within itself, the optical layer can establish, adapt and evolve a set of lightpaths that supports the required time-varying logical connectivity and capacity between routers as well as nodes of any other service layer networks.

An important simplifying practice that the OXC-based architecture also enables is the aggregation and grooming of traffic onto lightpaths at or near the edge of the network. By grooming and aggregating packet flows onto optical paths at the OC-48 level or higher, these transit flows bypass the routers en route optically, without adding to the forwarding load on the routers. Such optical bypass is especially important as demand becomes independent of distance, increasing the fraction of transiting flow at nodes. The only advantage of handling transiting flow at the packet level at every hop is a theoretical gain in statistical multiplexing efficiency in each hop. But doing so also increases average delay and packet loss and generates more cost, power and space consumption in the router. Well groomed optical paths can bypass the routers at almost as high utilization levels and without the corresponding risk of sudden extreme congestion that comes with the operation of single "fat" bit pipes at high utilizations. (The transient congestion of large pipes is why we say the higher utilization is only a theoretical benefit; it is risky in practice to reach high IP pipe utilization levels.) In the OXC-based architecture routers tend to handle only traffic that is within one of two remaining hops of their destination. The optical bypass the OXCs support thus also shifts the problem of scalability into the optical transport domain where doublings in capacity are far more easily obtained than in router speed. The OXC-based architecture is also more scalable because cross-connects are more compact, economic and lower in power consumption than large routers.

The OXC architecture also divides functionality and control complexity into smaller service-specific domains and its network state is largely physical, not based on software state and databases, and hence intrinsically more robust. Through user-network interfaces to the OXC, a variety of service layer networks can also manage their own internal state and service-related functionality and simply act as clients of the OXC asking for and/or releasing paths of requested capacity and protection status from the OXC-managed transport network. This is further distributes control responsibility, inherently putting fewer eggs in each basket compared to the entirely router-based architecture.

2.10.2 Concept of a "Transport Stabilized" Internet

The architecture of [Figure 2-13\(b\)](#) creates the option of using intelligence in the optical transport layer to simplify and enhance both the performance and stability of an overlying Internet layer. Internet "brownouts," routing storms, and other unpredictable and complex performance degradations are often caused by somewhat self-induced congestion effects and debilitating interactions with large numbers of IP traffic applications that compound these effects. An unexpected overload on one link may cause millions of TCP/IP sessions to back off, producing a massive self-synchronized load on the network again a few seconds later. In the meantime a flurry of LSAs may be generated from some router, or link failure or sheer transient link congestion may send the routers into a signaling-intensive phase of trying to re-synchronize their global database views and redistribute label information. The propensity for such complex dynamics in the

Internet is well known. In addition, there are more and more software complexities and configuration and database dependencies in evolving protocols.

But what if the Internet layer effectively never saw either link congestion or link failures? In fact one of the most common empirical practices today to stabilize IP layer performance is to simply keep all link capacities well over-provisioned. It is called "pouring bandwidth on the problem." If inter-router links in the IP layer are kept at 15% utilization or below, then experience is that delays and packet losses stay low and risks of complex interactions is minimized. In fact despite much research and development, IP layer links that approach 30% utilization tend to mark onset of loss and stability problems for network operators (signaling the need to pour on more bandwidth, etc). But this is hardly an efficient or safe way to run a network.

A more efficient alternative to pouring bandwidth onto the inter-router links of an inherently all-router transport network is intelligent transport networking below the IP layer. The idea is that many such complex dynamic degradations in the Internet could be avoided by adaptive capacity management in the optical network layer. The IP router layer would work under the simplest forwarding protocols possible, within a richly connected mesh of direct inter-router logical pipes, and with little or no apparent changes in the logical link topology and hence little or no activity for topology and routing updates. In the IP layer the capacity of each pipe in the logical fabric would appear to spontaneously grow (and decrease) just as needed to match the IP flows. Such an apparently stable and perfectly capacitated environment of network connectivity for the IP layer would be an "artificial world" created for the IP layer by the transport layer. Under such idealized conditions, the simplest suite of basic IP forwarding protocols actually works quite well. This is the vision of a "transport stabilized" Internet. An adaptive and survivable transport layer would *create the illusion* of an artificially perfect world for the Internet layer. This is done through adaptive creation and deletion of transport layer paths, and restoration or protection in the tens of milliseconds range. The interaction among OXC nodes would create this apparent world for the IP layer in which:

- Every time an IP pipe begins to approach congestion, or crosses a preset utilization threshold, the IP layer link capacity "magically" increases (then later decreases as well when this will be unnoticeable to reduced traffic).
- The IP links between routers appear not to fail, other than due to single link interface card failures on routers themselves, because physical failures are hidden in the transport layer.

Creation of this artificially ideal world for the IP layer stabilizes the IP layer: routing tables are almost never updated and LSAs are rare. End applications see a stable, predictable, low-delay environment and rarely need to go through TCP/IP window size and other transmission load-managing dynamics. The underlying transport layer produces this view by self-organizing adaptation of its logical capacity configuration to protect against actual failures and to adaptively configure physical capacity to best support the current loads on point-to-point logical links of the overlying IP layer.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Chapter 3. Failure Impacts, Survivability Principles, and Measures of Survivability

In this chapter we will look at causes of fiber cable failures, identify the impacts of outage, and relate these to the goals for restoration speed. We then provide an overview of the different basic principles and techniques for network survivability. This provides a first overview appreciation of the basic approaches of span, path and p -cycle based survivability which we treat in depth in later chapters. The survey of basic mesh-oriented schemes in this chapter also lets the reader see these schemes in contrast to ring-based schemes that are 100% or more redundant, and which we do not consider further in the book. The chapter concludes with a look at the quantitative measures of network survivability, and the relationships between availability, reliability and survivability.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

3.1 Transport Network Failures and Their Impacts

3.1.1 Causes of Failure

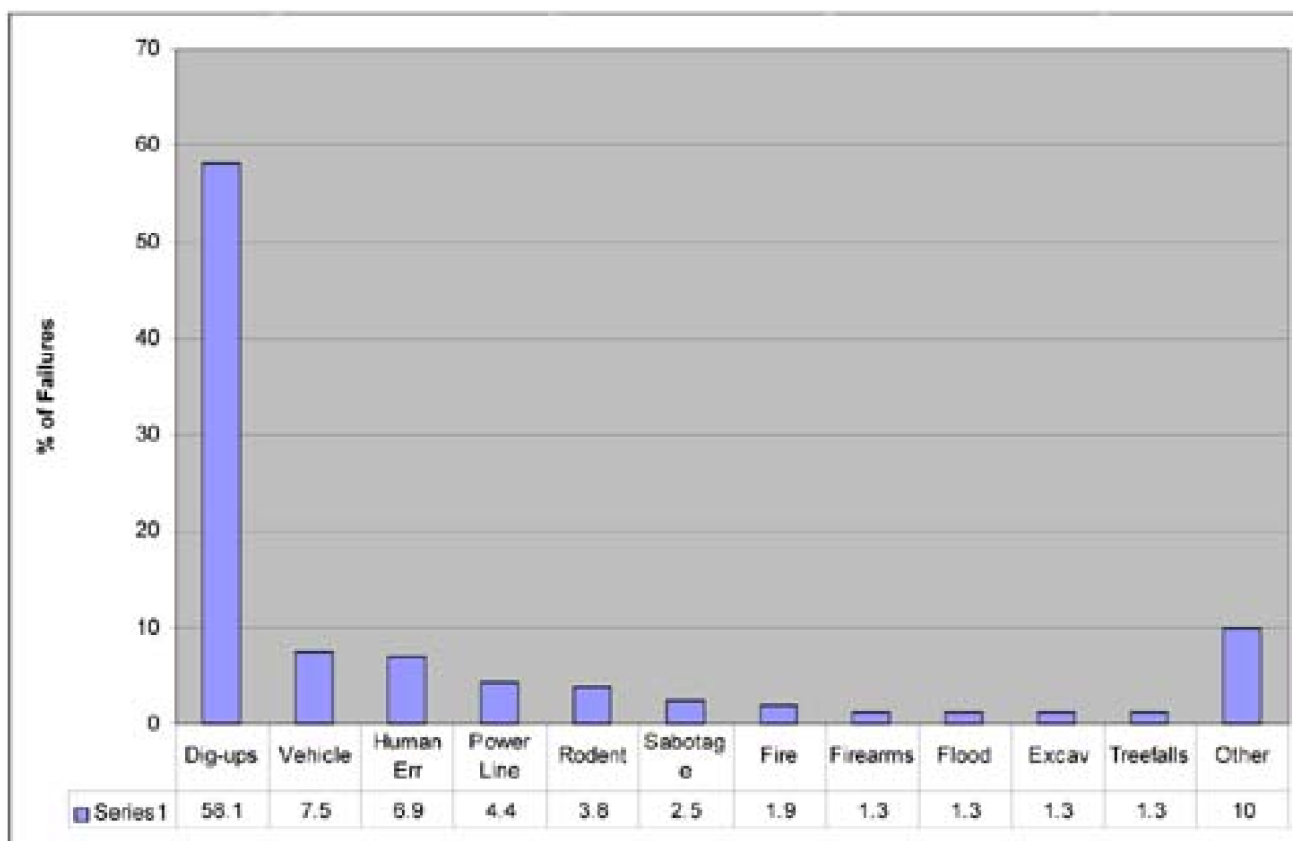
It is reasonable to ask why fiber optic cables get cut at all, given the widespread appreciation of how important it is to physically protect such cables. Isn't it enough to just bury the cables suitably deep or put them in conduits and stress that everyone should be careful when digging? In practice what seems so simple is actually not. Despite best-efforts at physical protection, it seems to be one of those large-scale statistical certainties that a fairly high rate of cable cuts is inevitable. This is not unique to our industry. Philosophically, the problem of fiber cable cuts is similar to other problems of operating many large-scale systems. To a lay person it may seem baffling when planes crash, or nuclear reactors fail, or water sources are contaminated, and so on, while experts in the respective technical communities are sometimes amazed it doesn't happen more often! The insider knows of so many things that *can* go wrong [[Vau96](#)]. Indeed some have gone as far as to say that the most fundamental engineering activity is *the study of why things fail* [[Ada91](#)] [[Petr85](#)].

And so it is with today's widespread fiber networks: it doesn't matter how advanced the optical technology is, it is in a cable. When you deploy 100,000 miles of any kind of cable, even with the best physical protection measures, it *will* be damaged. And with surprising frequency. One estimate is that any given mile of cable will operate about 228 years before it is damaged (4.39 cuts/year/1000 sheath-miles) [[ToNe94](#)]. At first that sounds reassuring, but on 100,000 installed route miles it implies more than one *cut per day* on average. To the extent that construction activities correlate with the working week, such failures may also tend to cluster, producing some single days over the course of a year in which perhaps two or three cuts occur. In 2002 the FCC also published findings that metro networks annually experience 13 cuts for every 1000 miles of fiber, and long haul networks experience 3 cuts for 1000 miles of fiber [[VePo02](#)]. Even the lower rate for long haul implies a cable cut every four days on average in a not atypical network with 30,000 route-miles of fiber. These frequencies of cable cut events are hundreds to thousands of times higher than corresponding reports of transport layer node failures, which helps explain why network survivability design is primarily focused on recovery from span or link failures arising from cable cuts.

3.1.2 Crawford's Study

After several serious cable-related network outages in the 1990s, a comprehensive survey on the frequency and causes of fiber optic cable failures was commissioned by regulatory bodies in the United States [[Craw93](#)]. [Figure 3-1](#) presents data from that report on the causes of fiber failure. As the euphemism of a "backhoe fade" suggests, almost 60% of all cuts were caused by cable dig-ups. Two-thirds of those occurred even though the contractor had notified the facility owner before digging. Vehicle damage was most often suffered by aerial cables from collision with poles, but also from tall vehicles snagging the cables directly or colliding with highway overpasses where cable ducts are present. Human error is typified by a craftsperson cutting the wrong cables during maintenance or during copper cable salvage activities ("copper mining") in a manhole. Power line damage refers to metallic contact of the strain-bearing "messenger cable" in aerial installations with power lines. The resulting i^2R (heat dissipation) melts the fiber cable. Rodents (mice, rats, gophers, beavers) seem to be fond of the taste and texture of the cable jackets and gnaw on them in both aerial and underground installations. The resulting cable failures are usually partial (not all fibers are severed). It seems reasonable that by partial gnawing at cable sheaths, rodents must also compromise a number of cables which then ultimately fail at a later time. Sabotage failures were typically the result of deliberate actions by disgruntled employees, or vandalism when facility huts or enclosures are broken into. Today, terrorist attacks on fiber optic cables must also be considered.

Figure 3-1. Immediate cause breakdown for 160 fiber optic cable cuts ([Craw93](#)).

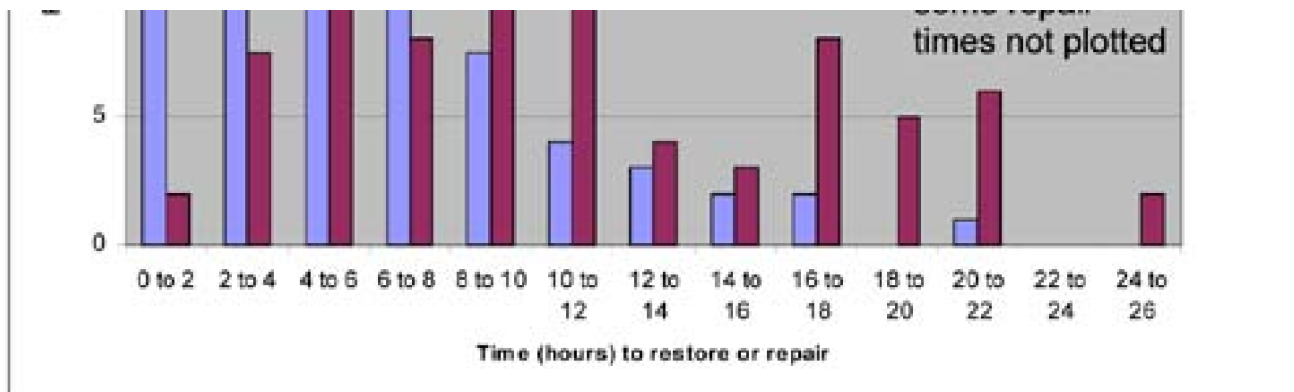


Floods caused failures by taking out bridge crossings or by water permeation of cables resulting in optical loss increases in the fiber from hydrogen infiltration. Excavation damage reports are distinct from dig-ups in that these were cases of failure due to rockfalls and heavy vehicle bearing loads associated with excavation activities. Treefalls were not a large contributor in this U.S. survey but in some areas where ice storms are more seasonal, tree falls and ice loads can be a major hazard to aerial cables. Conduits are expensive to install, and in much of the country cable burial is also a major capital expense. In parts of Canada (notably the Canadian shield), trenching can be almost infeasible as bedrock lies right at the surface. Consequently, much fiber cable mileage remains on aerial pole-lines and is subject to weather-related hazards such as ice, tree falls, and lightning strikes.

Figure 3-2 shows the statistics of the related service outage and physical cable repair times. Physical repair took a mean time of 14 hours but had a high variance, with some individual repair times reaching to 100 hours. The average service outage time over the 160 reported cable cuts was 5.2 hours. As far as can be determined from the report, all 160 of the cable failures reported were single-failure events. This is quite relevant to the applicability and economic feasibility of later methods in the book for optimal spare capacity design.

Figure 3-2. Histogram of service restoration and cable repair times (data from [Craw93]).





In 1997 another interesting report came out on the causes of failure in the overall public switched network (PSTN) [Kuhn97]. Its data on cable-related outages due to component flaws, acts of nature, cable cutting, cable maintenance errors and power supply failures affecting transmission again add up to form the single largest source of outages. Interestingly Kuhn concludes that human intervention and automatic rerouting in the call-handling switches were the key factors in the systems's overall reliability. This is quite relevant as we aim in this book to reduce the dependence on human intervention wherever possible in real-time and effectively to achieve the adaptive routing benefits of the PSTN down in the transport layer itself. Also of interest to readers is [Zorp89] which includes details of the famous Hinsdale central-office fire from which many lessons were learned and subsequently applied to physical node protection.

3.1.3 Effects of Outage Duration

There are a variety of user impacts from fiber optic cable failures. Revenue loss and business disruption is often first in mind. As mentioned in the introduction, the Gartner research group attributes up to \$500 million in business losses to network failures by the year 2004. Direct voice-calling revenue loss from failure of major trunk groups is frequently quoted at \$100,000/minute or more. But other revenue losses may arise from default on service level agreements (SLAs) for private line or virtual network services, or even bankruptcies of business that are critically dependent on 1-800 or web-pages services. Many businesses are completely dependent on web-based transaction systems or 1-800 service for their order intakes and there are reports of bankruptcies from an hour or more of outage. (Such businesses run with a very finely balanced cash-flow.) Growing web-based e-commerce transactions only increase this exposure. Protection of 1-800 services was one of the first economically warranted applications for centralized automated mesh restoration with AT&T's FASTAR system [ChDo91]. It was the first time 1-800 services could be assured of five minute restoration times. More recently one can easily imagine the direct revenue loss and impact on the reputation of "dot-com" businesses if there is any outage of more than a few minutes.

When the outage times are in the region of a few seconds or below, it is not revenue and business disruptions that are of primary concern, but harmful complications from a number of network dynamic effects that have to be considered. A study by Sosnosky provides the most often cited summary of effects, based on a detailed technical analysis of various services and signal types [Sosn94]. Table 3-1 is a summary of these effects, based on Sosnosky, with some updating to include effects on Internet protocols.

The first and most desirable goal is to keep any interruption of carrier signal flows to 50 ms or less. 50 ms is the characteristic specification for dedicated 1+1 automatic protection switching (APS) systems. An interruption of 50 ms or less in a transmission signal causes only a "hit" that is perceived by higher layers as a transmission error. At most one or two error-seconds are logged on performance monitoring equipment and data packet units for most over-riding TCP/IP sessions will not be affected at all. No alarms are activated in higher layers. The effect is a "click" on voice, a streak on a fax machine, possibly several lost frames in video, and on data services it may cause a packet retransmission but is well within the capabilities of data protocols including TCP/IP to handle. An important debate exists in the industry surrounding 50 ms as a requirement for automated restoration schemes. One view holds that the target for any restoration scheme must be 50 ms. Section 3.1.4 is devoted to a further discussion of this particular issue.

As one moves up from 50 ms outage time the chance that a given TCP/IP session loses a packet increases but remains well within the capability for ACK/NACK retransmission to recover without a backoff in the transmission rate and window size. Between 150-200 ms when a DS-1 level reframe time is added, there is a possibility (<5% at 200 ms) of exceeding the "carrier group alarm" (CGA) times of some older channel bank^[1] equipment, at which time the associated switching machine will busy out the affected trunks, disconnecting any calls in progress.

^[1] A channel bank is the equipment that digitizes and interleaves 24 analog voice circuits into a DS-1.

Table 3-1. Classification of Outage Time Impacts

Target Range	Duration	Main Effects / Characteristics
Protection Switching	< 50 ms	No outage logged: system reframes, service "hit", 1 or 2 error-seconds (traditional performance spec for APS systems), TCP recovers after one errored frame, no TCP fallback. Most TCP sessions see no impact at all.
1	50 ms - 200 ms	< 5% voiceband disconnects, signaling system (SS7) switch-overs, SMDS (frame-relay) and ATM cell-rerouting may start.
2	200 ms - 2 s	Switched connections on older channel banks dropped (CGA alarms) (traditional max time for distributed mesh restoration), TCP/IP protocol backoff.
3	2s - 10 s	All switched circuit services disconnected. Private line disconnects, potential data session / X.25 disconnects, TCP session time-outs start, web page not available errors. Hello protocol between routers begins to be affected.
4	10s - 5 min	All calls and data sessions terminated. TCP/IP application layer programs time out. Users begin attempting mass redials / reconnects. Routers issuing LSAs on all failed links, topology update and resynchronization beginning network-wide.
"Undesirable"	5 min - 30 min	Digital switches under heavy reattempts load, "minor" societal / business effects, noticeable Internet "brownout."
"Unacceptable"	> 30 min	Regulatory reporting may be required. Major societal impacts. Headline news. Service Level Agreement clauses triggered, lawsuits, societal risks: 911, travel booking, educational services, financial services, stock market all impacted.

With DS1 interfaces on modern digital switches, however, this does not occur until 2.5 +/- 0.5 seconds.^[2] Some other minor network dynamics begin in the range from 150-200 ms. In Switched Multi-megabit Digital Service (SMDS) cell rerouting processes would usually be beginning by 200 milliseconds. The recovery of any lost data is, however, still handled through higher layer data protocols. The SS7 common channel signaling (CCS) network (which control circuit-switched connection establishment) may also react to an outage of 100 ms at the SONET level (~150 ms after reframing at the DS-1 level). The CCS network uses DS-0 circuits for its signaling links and will initiate a switchover to its designated backup links if no DS-0 level synch flags are seen for 146 ms. Calls in the process of being set up at the time may be abandoned. Some video codecs using high compression techniques can also require a reframing process in response to a 100 ms outage that can be quite noticeable to users.

^[2] Whether at 230 ms or 2.5 s, it is reasonable to ask why a switch deliberately drops calls at all. One reason is that the switch must "busy out" the affected trunks to avoid setting up new calls into the failed trunk group. Another is to avoid processing the possibly random supervisory signaling state bits on the failed trunks. Doing so can threaten the switch call-processing resources (CPU, memory and real-time) causing a crash.

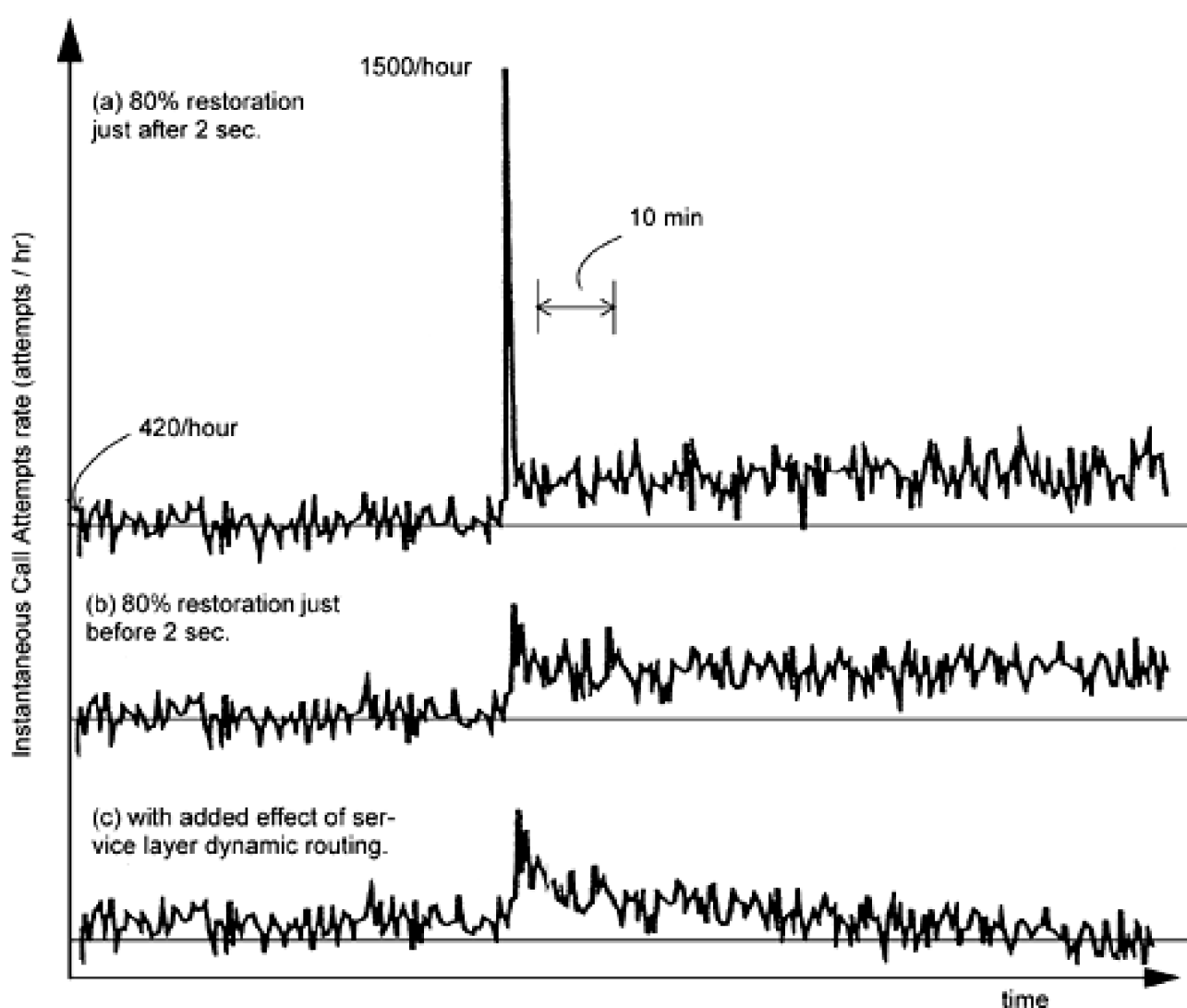
In the time frame from 200 ms to two seconds no new effects on switched voiceband services emerge other than those due to the extension of the actual signal lapse period itself. By two seconds the roughly 12% of DS0 circuits that are carried on older analog channel banks (at the time of Sosnosky's study) will definitely be disconnected. In the range from two to 10 seconds the effects become far more serious and visible to users. A quantum change arises in terms of the service-level impact in that virtually all voice connections and data sessions are disconnected. This is the first abrupt perception by users and service level applications of *outage* as opposed to a momentary hit or retransmission-related throughput drop. At 2.5 +/- 0.5 seconds, digital switches react to the failure states on their transmission interfaces and begin "trunk conditioning"; DS-0, (n)xDS-0 (i.e., "fractional T1"), DS-1 and private line disconnects ("call-dropping") occur. Voiceband data modems typically also time out two to three seconds after detecting a loss of carrier. Session dependent applications such as file transfer using IBM SNA or TCP/IP may begin timing out in this region, although time-outs are user programmable up to higher values (up to 255 seconds for SNA). X.25 packet network time-outs are typically from one to 30 seconds with a suggested time of 5 seconds. When these timers expire, disconnection of all virtual calls on those links occurs. B-ISDN ATM connections typically have alarm thresholds of about five seconds.

In contrast to the 50 ms view for restoration requirements, this region of 1 to 2 second restoration is the main objective that is accepted by many as the most reasonable target, based largely on the cost associated with 1+1 capacity duplication to meet 50 ms, and in recognition that up until about 1 or 2 seconds, there really is very little effect on services. However, two seconds is really the "last chance" to stop

serious network and service implications from arising. It is interesting that some simple experiments can dramatically illustrate the network dynamics involved in comparing restoration above and below a 2 second target (whereas there really are no such abrupt or quantum changes in effects at anywhere from zero up to the 2 second call-dropping threshold).

Figure 3-3 shows results from a simple teletraffic simulation of a group of 50 servers. The servers can be considered circuits in a trunk group or processors serving web pages. The result shown is based on telephony traffic with a 3 minute holding time. The 50 servers are initially in statistical equilibrium with their offered load at 1% connection blocking. If a call request is blocked, the offering source reattempts according to a uniform random distribution of delay over the 30 seconds following the blocked attempt. Figure 3-3(a) shows the instantaneous connection attempts rate, if the 50 trunk group is severed and all calls are dropped, then followed by an 80% restoration level. Figure 3-3(b) shows the corresponding dynamics of the same total failure, also followed by only 80% restoral, but before the onset of call dropping. Figure 3-3(c) shows how the overall transient effect is yet further mitigated by adaptive routing in the circuit-switched service layer to further reduce ongoing congestion. This dramatically illustrates how beneficial it is in general to achieve a restoration response before connection or session dropping, even if the final restoral level is not 100%.

Figure 3-3. Traffic dynamic effects (semi-synchronized mass re-attempts) of restoration beyond the call-dropping limit of ~2 seconds (collaboration with M. MacGregor).



The seriousness of an outage that extends beyond several seconds, into the tens of seconds, grows progressively worse: IP networks begin discovering "Hello" protocol failures and attempt to reconverge their routing tables via LSA flooding. In circuit-switched service layers, massive connection and session dropping starts occurring and goes on for the next several minutes. Even if restoration occurred at, say, 10 seconds, there would by then be millions of users and applications that begin a semi-synchronized process of attempting to re-establish their connections. There are numerous reports of large digital switching systems suffering software crashes and cold reboots in the time frame of 10 seconds to a few minutes following a cable cut, due to such effects. The cut itself might not have affected the basic switch stability, but the mass re-attempt overwhelms and crashes the switch. Similar dynamics apply for IP large routers forwarding packets for millions of TCP/IP sessions that similarly undergo an unwittingly synchronized TCP/IP backoff and restart. (TCP/IP involves a

rate backoff algorithm called "slow start" for response to congestion. Once it senses rising throughput the transmit rate and window size is multiplied in a run up to the maximum throughput. Self-synchronized dynamics among disparate groups of TCP/IP sessions can therefore occur following the failure or during the time routing tables are being updated). The same kind of dynamic hazards can be expected in MPLS-based networks as label edge routers (LERs) get busy (following OSPF-TE resynchronization) with CR-LDP signaling for re-establishment of possibly thousands of LSPs simultaneously through the core network of LSRs. Protocols such as CR-LDP for MPLS (or GMPLS) path establishment were not intended for, nor have they ever been tested in an environment of mass simultaneous signaling attempts for new path establishment. The overall result is highly unpredictable transient signaling congestion and capacity seizure and contention dynamics. If failure effects are allowed to even enter this domain we are ripe for "no dial tone" and Internet "brown outs" as switch or router O/S software succumbs to overwhelming real-time processing loads. Such congestion effects are also known to propagate widely in both the telephone network and Internet. Neighboring switches cannot complete calls to the affected destination, blocking calls coming into themselves, and so on. If anything, however, the Internet is even more vulnerable than the circuit switched layer to virtual collapse in these circumstances. ^[3]

^[3] The following unattributed quote in the minutes of a task force on research priorities makes the point about Internet reliability rather imaginatively: (paraphrasing) "What would you do if I grabbed my chest and fell down during a meeting—Dial 911? Imagine opening a browser and typing in <http://www.911.org> instead?"

Beyond 30 minutes the outage effects are generally considered so severe that it is reportable to regulatory agencies and the general societal and business impacts are considered to be of major significance. If communications to or between police, ambulance, medical, flight traffic control, industrial process control or many other such crucial services break down for this long it becomes a matter of health and safety, not just business impact. In the United States any outage affecting 30,000 or more users for over 30 minutes is reportable to the FCC.

3.1.4 Is 50 ms Restoration Necessary?

Any newcomer to the field of network survivability will inevitably encounter the "50 ms debate." It is well to be aware that this is a topic that has been already argued without resolution for over a decade and will probably continue. The debate persists because it is not entirely based on technical considerations which could resolve it, but has roots in historical practices and past capabilities and has been a tool of certain marketing strategies.

History of the 50 ms Figure

The 50 ms figure historically originated from the specifications of APS subsystems in early digital transmission systems and was not actually based on any particular service requirement. Early digital transmission systems embodied 1:N APS that required typically about 20 ms for fault detection, 10 ms for signaling, and 10 ms for operation of the tail-end transfer relay, so the specification for APS switching times was reasonably set at 50 ms, allowing a 10 ms margin. Early generations of DS1 channel banks (1970s era) also had a Carrier Group Alarm (CGA) threshold of about 230 ms. The CGA is a time threshold for persistence of any alarm state on the transmission line side (such as loss of signal or frame synch loss) after which all trunk channels would be busied out. The 230 ms CGA threshold reinforced the need for 50 ms APS switches at the DS3 transmission level to allow for worst-case reframe times all the way down the DS3, DS2, DS1 hierarchy with suitable margin against the 230 ms CGA deadline. It was long since realized that a 230 ms CGA time was far too short, however. Many minor line interruptions would trigger an associated switching machine into mass call-dropping because of spurious CGA activations. The persistence time before call dropping was raised to 2.5 +/- 0.5 s by ITU recommendations in the 1980s as a result. But the requirement for 50 ms APS switching stayed in place, mainly because this was still technically quite feasible at no extra cost in the design of APS subsystems. The apparent sanctity of 50 ms was further entrenched in the 1990s by vendors who promoted only ring-based transport solutions and found it advantageous to insist on 50 ms as the requirement, effectively precluding distributed mesh restoration alternatives which were under equal consideration at the start of the SONET era. As a marketing strategy the 50 ms issue thus served as the "mesh killer" for the 1990s as more and more traditional telcos bought into this as dogma.

On the other hand, there was also real urgency in the early 1990s to deploy some kind of fast automated restoration method relatively immediately. This led to the quick adoption of ring-based solutions which had only incremental development requirements over 1+1 APS transmission systems. However, once rings were deployed, the effect was to only further reinforce the cultural assumption of 50 ms as the standard. Thus, as sometimes happens in engineering, what was initially a performance *capability* in one specific context (APS switching

time) evolved into a perceived *requirement* in all other contexts.

But the "50 ms requirement" is undergoing serious challenges to its validity as a ubiquitous requirement, even being referred to as the "50 ms myth" by data-centric entrants to the field who see little actual need for such fast restoration from an IP services standpoint. Faster restoration is by itself always desirable as a goal, but restoration goals must be carefully set in light of corresponding costs that may be paid in terms of limiting the available choices of network architecture. In practice, insistence on "50 ms" means 1+1 dedicated APS or UPSR rings (to follow) are almost the only choices left for the operator to consider. But if something more like 200 ms is allowed, the entire scope of efficient shared-mesh architectures become available. So it is an issue of real importance as to whether there are any services that truly require 50 ms.

Sosnosky's original study found no applications that require 50 ms restoration. However, the 50 ms requirement was still being debated in 2001 when Schallenburg [[Schal01](#)], understanding the potential costs involved to his company, undertook a series of experimental trials with varying interruption times and measured various service degradations on voice circuits, SNA, ATM, X.25, SS7, DS1, 56 kb/s data, NTC digital video, SONET OC-12 access services, and OC-48. He tested with controlled-duration outages and found that 200 ms outages would not jeopardize any of these services and that, except for SS7 signaling links, all other services would in fact withstand outages of two to five seconds.

Thus, the supposed requirement for 50 ms restoration seems to be more of a techno-cultural myth than a real requirement—there are quite practical reasons to consider 2 seconds as an alternate goal for network restoration. This avoids the regime of connection and session time-outs and IP/MPLS layer reactions, but gives a green light to the full consideration of far more efficient mesh-based survivable architectures.

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

3.2 Survivability Principles from the Ground Up

As in many robust systems, "defence in depth" is also part of communication network survivability. We will now look at various basic techniques to combat failures and their effects, starting right at the physical layer. [Table 3-2](#) follows the approach of [\[1A193\]](#) and identifies four levels at which various survivability measures can be employed. Each layer has a generic type of demand unit that it provides to the next higher level. As in any layering abstraction, the basic idea is that each layer exists to provide a certain service to its next higher layer, which need know nothing about how the lower layer implements the service it provides. Here it is capacity units of various types that each layer provides to the next to bear aggregations of signals or traffic formed in the next higher layer. It is important to note that although a layered view is taken it is not implied that one or more methods from each layer must necessarily be chosen and all applied on top of each other. For instance, if rings are implemented at the system layer, then there may be no survivability measures (other than against intra-system circuit-pack level of failures) implemented at the logical layer, and vice-versa. Additionally, certain service layer networks may elect to operate directly over the physical layer, providing their own survivability through adaptive routing. In contrast, however, certain physical layer measures must always be in place for any of the higher layers to effect survivability. In this framework it is usually the system and logical layers, taken together, that we refer to when we speak of "transport networking" in general.

Table 3-2. Layered view of networks for survivability purposes

Layer	Elements	Service and Functions	Demand Units Generated	Capacity Units Provided	Generic Survivability Techniques
Service	IP routers, LSRs telephone switches, ATM switches, smart channel banks	Circuit-switched telephony and data, Internet, B-ISDN private networks, multi-media	OC-3, OC-12, STS-1s, DS-1s, DS-3s GbE, etc.	n/a	Adaptive routing, demand splitting, application re-attempt
Logical	OXC DCS ATM VP X-connects	Services grooming, logical transport configuration, bandwidth allocation and management	OC-48, OC-192, wavelength channels, wavebands	OC-3, OC-12, STS-1s, DS-1s, DS-3s GbE, etc.	Mesh protection or restoration DCS-based rings <i>p</i> -cycles
System	SONET OC-n TM, LTE, ADMs, OADMs WDM transmission systems	Point-to-point bit-transmission at 10 to 40 Gbs/s Point-to-point fiber or wavelengths	fibers, cables	OC-48 OC-192 wavelength channels, wavebands	1:N APS 1+1 DP APS, rings
Physical	Rights-of-way, conduits, pole-lines, huts, cables, ducts	Physical medium of transmission connectivity	n/a	Fibers, cables	Physical encasement, physical diversity

3.3 Physical Layer Survivability Measures

The physical layer, sometimes called Layer 0, is the infrastructure of physical resources on which the network is based: buildings, rights-of-way, cable ducts, cables, underground vaults, and so on. In this layer, survivability considerations are primarily aimed at physical protection of signal-bearing assets and ensuring that the physical layer topology has a basic spatial diversity so as to enable higher layer survivability techniques to function.

3.3.1 Physical Protection Methods

A number of standard practices enhance the physical protection of cables. In metropolitan areas PVC tubing is generally used as a duct structure to give cables a fairly high degree of protection, albeit at high cost. Outside of built up areas, fiber cables are usually direct-buried (without the PVC ducts), at 1.5 to 2 meters depth, and a brightly colored marker ribbon is buried a foot above the cable as a warning marker. There is usually a message such as "Warning: Optical Cable—STOP" on the tape. It is standard practice to also mark all subsurface cable routes with above-ground signs, but these can be difficult to maintain over the years. In some cases where the water table is high, buried cables have actually been found to move sideways up to several meters from their marked positions on the surface. "Call before you dig" programs are often made mandatory by legislation to reduce dig-ups. And hand digging is required to locate the cable after nearing its expected depth within two feet. Locating cables from the surface has to be done promptly and this is an area where geographical information systems can improve the operator's on-line knowledge about where they have buried structures (and other network assets). Cable locating is also facilitated by application of a cable-finding tone to the cable, assuming (as is usual) that a metallic strength member or copper pairs for supervisory and power-feeding are present. Measures against rodents include climbing shields on poles and cable sheath materials designed to repel rodents from chewing. On undersea cables the greatest hazard is from ship anchors and fishnets dragging on the continental shelf portions. Extremely heavily armored steel outer layers have been developed for use on these sections as well as methods for undersea trenching into the sea floor until it leaves the continental shelf. Beyond the continental shelf cables are far less heavily armored and lay on the sea floor directly. Interestingly, the main physical hazard to such deep sea cables appears to be from shark bites. Several transoceanic cables have been damaged by sharks which seem to be attracted to the magnetic fields surrounding the cable from power-feeding currents. Thus, even in this one case where it seems we might not have to plan for cable cut, it is not so.

Underground cables are either gel-filled to prevent ingress of water or in the past have been air pressurized from the cable vault. An advantage of cable pressurization is that with an intact cable sheath there will normally be no flow. Any loss of sheath integrity is then detected automatically when the pressurization system starts showing a significant flow rate. In addition to the main hazards to "aerial" cables of vehicles, tree falls and ice storms mentioned by Crawford, vandalism and gunshots are another physical hazard to cables. A problem in some developing countries is that aerial fiber optic cables are mistaken for copper cables and pulled down by those who steal copper cable for salvage or resale value. Overall, however, aerial cables sustain only about one third as many cable cuts as do buried cables from dig-ups. And (ironically), while buried cable routes are well marked on the surface, experience with aerial cables shows it better *not* to mark fiber optic cables in any visibly distinct way to avoid deliberate vandalism.

3.3.2 Reducing Physical Protection Costs with Restoration Schemes

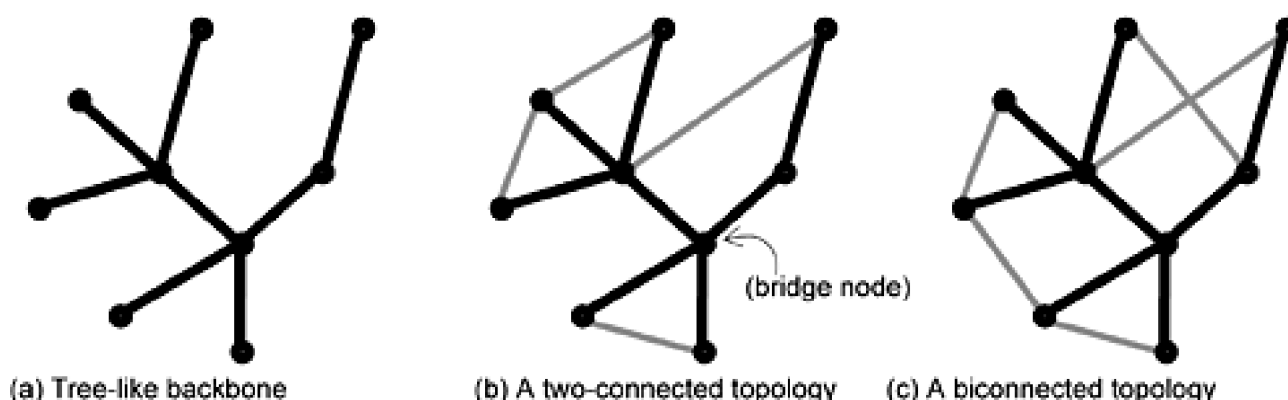
The cost of trenching to bury fiber cable can be quite significant and is highly dependent on the depth of burial required. An interesting prospect of using active protection or restoration schemes is that an operator may consider relaxing the burial depth (from 2 m to 1.5 m, say), relative to previous standards for point-to-point transmission systems. An (unpublished) study by the author found this to be quite a viable strategy for one regional carrier. The issue was that existing standards required a 2 meter burial depth for any new cable. It would have been very expensive to trench to 2 meters depth all the way through a certain pass in the Rocky Mountains. But since these cables were destined to be part of either a restorable ring or mesh network, the question arose: "Do we really still have to bury them to two

meters?" Indeed, availability calculations (of the general type in [Section 3.12](#)) showed that about a thousand-fold increase in the physical failure rate could be sustained (for the same overall system availability) if the fibers in such cables were parts of active self healing rings. Given that the actual increase in failure rate from relaxing the depth by half a meter was much less than a thousand-fold, the economic saving was possible without any net loss of service availability. Essentially the same trade-off becomes an option with mesh-based restorable networking as well and we suggest it as an area of further consideration whenever new cable is to be trenched in.

3.3.3 Physical Layer Topology Considerations

When a cable is severed, higher layers can only restore by rerouting the affected carrier signals over physically diverse surviving systems. Physically disjoint routes must therefore exist in Layer 0. This is a basic requirement for survivability that no other layer can provide or emulate. Before the widespread deployment of fiber, backbone transport was largely based on point-to-point analog and digital microwave radio and the physical topology did not really need such diversity. Self-contained 1:N APS systems would combat fading from multipath propagation effects or single-channel equipment failures but there was no real need for restoration considerations in the sense of recovery from a complete failure of the system. The radio towers themselves were highly robust and one cannot easily "cut" the free space path between the towers. National scale networks consequently tended to have many singly connected nodes and roughly approximated a minimum length tree spanning all nodes. Fiber optics, being cable-based, however forces us to "close" the physical topologies into more mesh-like structures where no single cut can isolate any node from the rest. The evolution this implies is illustrated in [Figure 3-4](#).

Figure 3-4. For survivability against failures that overcome physical protection measures, the physical layer graph must be either two-connected or biconnected.



Technically, the physical route structure must provide either two-connectedness or biconnectedness over the nodes. In a biconnected network, there are at least two fully disjoint paths between each node pair. Two-connectedness implies that two span-disjoint paths exist between all node pairs, but in some cases there may be a node in common between the two paths. Algorithmic tests for this property are discussed in [Chapter 4](#), although these properties are readily apparent to a human viewing a plan diagram of the network. Note that this topological evolution to a closed graph of some form has by itself nothing to do with the speed or type of restoration scheme to be employed. It is just topologically essential to have at least two physically disjoint routes between every node pair for automatic restoration by diverse routing to even be an option.

In practice, however, the acquisition of rights-of-way to enhance physical layer diversity can be very costly. Whereas a spanning tree requires as few as $N-1$ spans to cover N nodes, and can do so efficiently in terms of total distance, a biconnected graph requires at least N spans (which makes a single but long physical ring) and more typically up to $1.5N$ for a reasonably well-connected and distance-minimized topology to support either ring- or mesh-based survivable transport networking. Thus, depending on the legacy network or starting point for evolution to mesh-based operation, a major expense may be incurred in the physical layer to ensure that higher layer survivability schemes can operate. Thus, optimization and evolution of the physical layer topology is one of the fundamental problems faced by modern network operators. This problem is treated further in [Chapter 9](#).

3.3.4 The Problem of Diversity Integrity

Related to the creation of physical layer diversity is the need also to be able to validate the details of physical structures that underlie logical protection or restoration routes to ensure integrity of the mapping from physical to logical diversity. For instance, how does one know that the opposite spans of a SONET ring correspond to cables that are on different sides of the street? Maybe they are on one street but two blocks later they share the same duct or bridge-crossing. This is the issue of shared risk link groups mentioned in [Chapter 1](#). It is one thing to recognize that we will have to take SRLGs into account, but the further point here is that even knowing with certainty what the mapping of each logical path into physical structures is (hence defining the SRLGs) is itself a difficult and important aspect of the physical network. This general issue is one of being able to correlate logical level service or system implementations to the ultimate underlying physical structures. This is a significant administrative challenge because cables and ducts may have been pulled in at different times over several decades. The end points may be known, but correlating these to different outside plant conduits and pole structures, etc., is the problem. Many telcos are investing heavily in geographic information systems and conducting ground-truth audits to keep tabs on all these physical plant details. Without assured physical diversity (or at least knowledge of the SRLGs on a given path pair), attempts to provide redundancy to enable active protection or restoration are easily defeated. More about the problem of ensuring physical diversity follows after our review of protection options at all layers.

The "Red and White" Network

One interesting proposal to address this physical diversity assurance problem, and provide a very simple and universal strategy for survivability, is the concept of a "red and white" network.^[4] The suggestion, not made frivolously, is to purchase and install every physical item of a network in duplicate and classify one of each pair as either "red" or "white," and literally paint every item accordingly. Only one rule then ever need be applied network-wide: always keep red and white apart, whether cables, power supplies, equipment bays, etc. When followed through the result would be an entirely dual-plane network with complete physical disjointedness between planes. Every application warranting protected service would then be realized once in the red plane and again in the white plane, network-wide. The result is assured physical diversity and the operations and planning simplicity of 1+1 tail-end selection as the network-wide survivability principle, for a 100% investment in redundancy in both node and span equipment. Lest it seem that the idea of completely duplicating the network is unrealistic, it should be pointed out that ring-based networks often embody 200 to 300% capacity redundancy, and although the nodal equipment elements are not completely duplicated in ring-based transport, it is normal to have 1+1 local standby redundancy on high speed circuit packs, processors and power supplies built into these network elements. In contrast each plane of the "red and white" network would use fully non-redundant individual equipment items. Importantly, however, we will see that mesh-based networking can achieve survivability with much less than 100% capacity redundancy and can also provide differentiated levels of service protection.

^[4] While the concept is easy to remember, the author's best recollection is only that this proposal was made in a verbal presentation by T. Sawyer of Southern Bell Communications at an IEEE DCS Workshop in the mid-1990s.

3.4 Survivability at the Transmission System Layer

Next up from the bottom in [Table 3-2](#) is the "system layer," named in reference to the transmission systems found at this layer. This is the level at which, historically and prior to consideration of mesh-based survivability, almost all active measures to react against single-channel failures or cable cuts have been implemented. This includes mainly linear APS schemes, ring schemes, and the recent p -cycle technique, of which we give an overview in this section.

It is important to note that survivability techniques at the system layer also include basic equipment-level design redundancy. Built-in equipment redundancy includes dual power feed connections and power converters, usually dual maintenance and control processors and may often include 1+1 redundant high-speed optical transmit/receive interface cards as well. Standard design methods may also include error-correction coding, in-service bit-error-rate monitoring, loss of signal detectors, laser bias current monitors, and so on. These are all measures that increase the system availability by ensuring its ability to carry on performing its function in the face of faults that may arise *within the system itself*. This kind of designed-in redundancy is extensive in telecommunications equipment: redundant power supplies, processors, tape drives, frequency allocations, antennas, lasers, etc. All this goes in to achieve basic operational availability levels that are often matched only in space, military, and nuclear applications.

Generally system layer survivability schemes for combating cable cuts are *protection* schemes (as opposed to later *restoration* schemes). The main characteristics of a protection scheme is that the protection route and standby capacity are predefined and the mechanism is self-contained within the transmission system layer. It usually involves redirecting the composite optical line signal as a whole without processing or identifying any of its constituent tributaries. The distinction between protection and restoration is not wholly black and white, however, and we later refine the basic categorizations.

3.4.1 "Linear" Transmission Systems

Historic examples of linear transmission systems include point-to-point digital carrier systems operating at the DS-1 rate on twisted pair, up to DS-4 on coaxial cable, and PDH-based transmission systems operating at 12 and 24 x DS-3 rates (565 Mb/s and 1.2 Gb/s) per fiber using proprietary higher-than-DS3 framing structures. Such systems included a 1:N automatic protection switching (APS) subsystem to protect single-fiber or regenerator failures, but had no designed-in measures to combat a complete cable cut. SONET not only defined standards for the higher-than-DS3 line rates but also extended the capabilities of the transmission systems to include a "nested" 1:N APS configuration which allowed for add/drop multiplexing within the span of the protection channel. Effectively the one protection system would be shared by all the subtending add/drop segments by breaking into and out of the protection channel where needed, as opposed to switching end-to-end over to protection. [\[Wu92\]](#) provides more details. Using any of these so-called "linear" transmission systems (in contrast to later ring systems), the only way to withstand cable cuts was to use 1+1 APS and route the protection system over a physically diverse route. This would establish a "1+1 DP" APS arrangement, DP standing for diverse protection. 1+1 DP is effectively two parallel instances of the same linear transmission system.

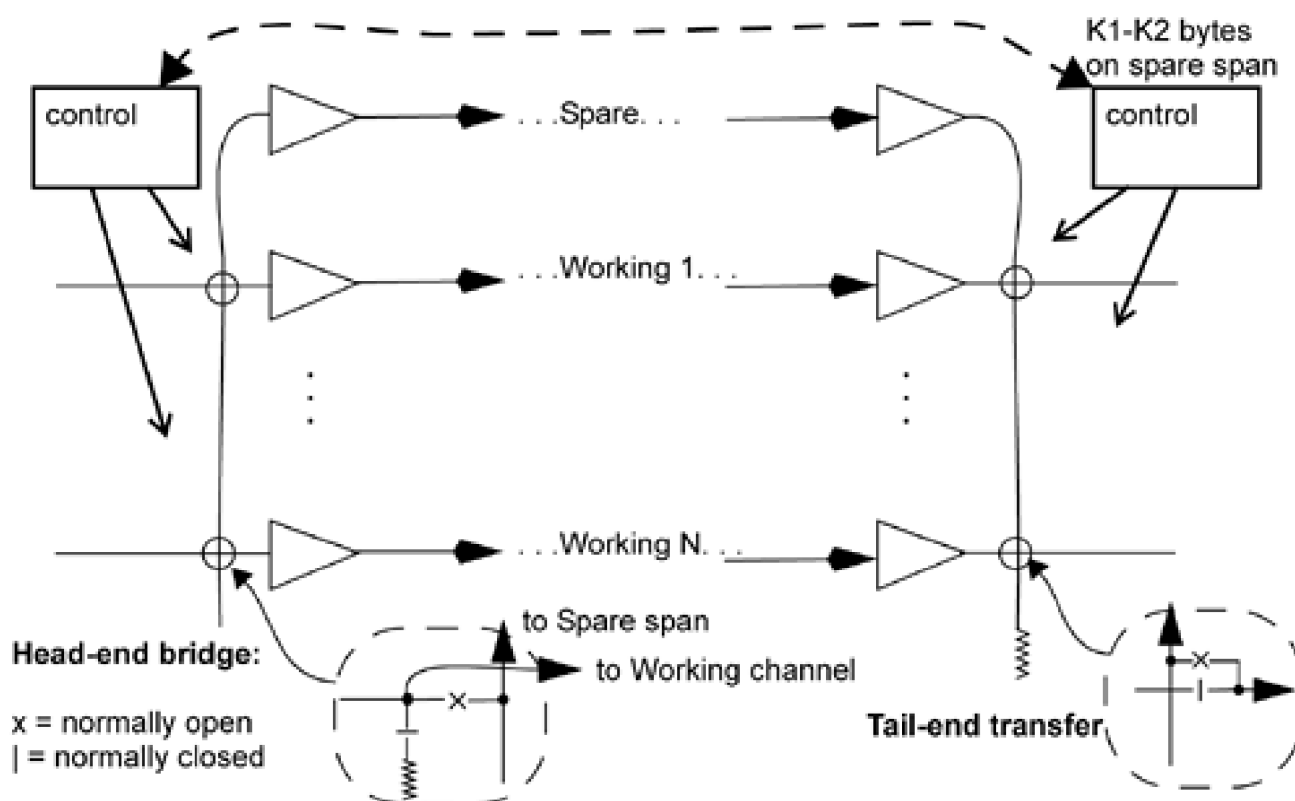
3.4.2 Automatic Protection Switching (APS)

Let us now give meaning to the notations 1:1, 1+1, etc. which are widely used in the context of APS systems. 1+1 denotes a dedicated standby arrangement: one working system and a completely reserved backup system in which the transmit line signal is copied (called head-end bridging) and drives both signal paths. 1+1 DP implies that the protection channel is routed over physically diverse rights-of-way from the working system. The fastest possible switching speed is obtained with 1+1 because the receivers need only monitor both receive signal copies and switch from one to the other if either fails. The receiver switch-over is called "tail-end-transfer." 1:1 APS is like 1+1 but the transmit signal is not kept in a bridged state. The resources needed are the same as in 1+1, but 1:1 can allow other uses for the protection channel when not needed by the working channel. 1:1 operation can be of advantage in providing "extra traffic" services (below)

or in maintenance and trouble testing because one of the two signal paths can be taken off line for intrusive testing, then the signal "force switched" and the other path separately tested. The switching speed of 1:1 APS is, however, slightly slower than 1+1 because the transmit signal is not bridged at all times and receiver detection of a working channel failure must be signalled back to the head-end to request the head-end bridge establishment to effect protection.

1:N APS denotes that N working systems share one standby "protection" system in an arrangement such as in [Figure 3-5](#). The intent and ability of a 1:N APS system is only to protect against single channel or fiber failures by using a standby channel within the system itself. The standby need not be diverse routed because there is no ability or intent with 1:N APS to protect against a complete cable cut. The protection or spare channel system in 1:N APS is inherently shared since $N > 1$. In 1:N APS the receiving end of a failed channel detects the failure and checks if the spare span is available. If so, it signals to the other end of the system to request a head-end bridge of the failed channel onto the spare span. Return signaling confirms the number of the selected channel and once the spare-span receiver is in-lock at suitably low BER on the new spare-span signal, a tail-end transfer relay substitutes the spare span signal for the original working channel signal at the system output port. A converse head-end bridge and tail-end transfer is also set up for the other direction of transmission on the failed channel. In SONET this is implemented in a simple state-driven protocol using the K1-K2 overhead bytes on the protection channel. The same protocol also drives the SONET BLSR ring switching mechanism. Although the K1-K2 byte signaling protocol is simple, it is significant conceptually in that it stands as an alternate paradigm to explicit inter-processor data messaging to perform certain time-critical rerouting functions in the most robust and reliable real-time way possible. The distributed restoration algorithms (DRAs) mentioned later for span restoration ([Chapter 5](#)) use an extension of this form of signaling to realize generalized mesh rerouting as opposed to APS. The K1-K2 byte protocol involves two finite state machines at each end of the APS system, one associated with its transmit direction, for which it has head-end bridge responsibility, the other for its receive direction, for which it has tail-end transfer responsibility. If the spare span is not already in use, the protocol is:

Figure 3-5. Head-end bridge and tail-end transfer functions illustrated in a 1:N APS system.



1:N APS Protocol

Tail-end role:

{state = idle; event= receive failure on Ch x ;

```
action = transmit "x" on K1 byte on spare span; (bridge request)
next state: wait}
{state = wait; event= receive "x" on spare K2; (bridge confirm)
action = tail-end transfer (substitute spare span output for Ch x system output);
next state = protected}
```

Head-end role:

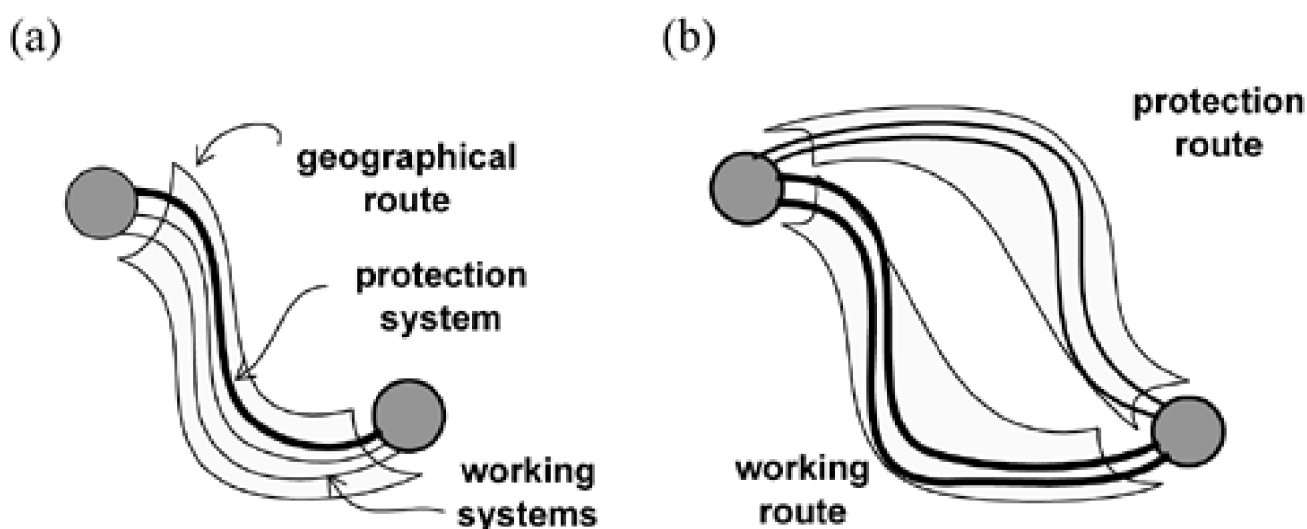
```
{state = idle; event= receive "x" on spare span K1 byte;
action = {set up head-end bridge for Ch x;
          transmit "x" on K2 byte on spare span (bridge confirm);
          transmit "x" on K1 byte on spare span (bridge request);}
next state: wait}
{state = wait; event= receive "x" on spare K2 byte(bridge confirm);
action = tail-end transfer (substitute spare span output for Ch x system output);
next state = protected}
```

The extension of 1:N APS protection to $k:N$ APS where $k > 1$ follows directly except that the protection control logic must then manage allocation of single channel failures on N working channels to k available protection channels.

A possibly confusing industry trend is to refer to shared mesh restoration schemes as achieving "*m for n protection*," and even denoting this $m:n$. In this context, however, people do not mean to suggest that $m:n$ APS systems are literally being employed. Rather they are referring in general to the attribute that mesh networks achieve a certain characteristic sharing of protection capacity an *overall* network basis; that is to say that for every n units of working capacity there are m units of spare capacity ($n > m$) in the network *on average*. This will come to have more meaning as we look at mesh restoration schemes and capacity design; it will become apparent how such sharing of all protection channels occur over all working channels of the network without implying specifically established $m:n$ APS subgroups or systems.

Note that if APS methods are intended to protect against cable cuts they have to be either 1+1 or 1:1 DP APS, not 1:N APS. 1:N APS is usually employed as a high-availability equipment level system design method to protect against *internal* failures of the system and all channels—the N working and one standby—are routed together. But protection against *externally imposed* failures requires 1+1 or 1:1 DP (or other schemes which follow). Typically therefore one finds equipment designs employing combinations of 1:1 APS on the high-speed "line" side (where the risk is of a complete cable cut) and something like 1:7 APS on lower-speed circuit packs or cross-office interconnection interfaces etc. (where the risk is primarily of a single electronics or connector failure). [Figure 3-6](#) illustrates 1:N APS versus 1+1 DP APS.

Figure 3-6. (a) 1:N (N=2) co-routed APS and (b) 1+1 diverse-protection (DP) APS.



3.4.3 Reversion

In a 1:N protection switching system, and more generally in restoration or protection of any kind using spare capacity sharing, the protected signal path must be returned to its normal working path (or another working path) following physical repair so that the protection capacity is accessible again for the next failure that might arise. This is usually done with care to minimize subsequent "hits" on the customer payload. To minimize such a reversion hit the usual procedure is to set up a head-end bridge to supply a copy of the signal to the repaired signal path. This does not interrupt the restored signal path. Tests are done while in the bridged state to validate the receive signal quality and then a tail-end transfer switch substitutes the repaired working path signal for the restored signal. A hit, usually only of 10 ms or so in duration, arises only due to the tail-end transfer. This is called a "bridge and roll" process.

In systems such as 1+1 APS or a UPSR, reversion is not actually necessary. If done at all it is only for administrative reasons. Generally in any mesh protection or restoration scheme reversion will be required after physical repair, because we always want to return the system to a state of readiness for a next failure. Since we are always using shared capacity for efficiency, this means returning signals to their pre failure routing. This practice also avoids the accumulation of longer-than-required working routes that would result from a non-reversion policy in a mesh restorable network. Unlike the failure that triggered a restoration event, however, reversion itself is a process whose timing and pace the network operator controls following physical repair and can be scheduled late at night and/or coordinated directly with the service users to minimize customer impact.

3.4.4 "Extra Traffic" Feature

1:N or 1:1 APS systems, including their extensions into BLSR rings (to follow shortly) support a practice called "extra traffic." This is a feature that allows the network operator to transport any other lower-priority traffic (in compatible format for the APS or ring's line-rate signal) over the protection channel. Extra traffic is bumped off if the APS or ring system switches to protect its own working channels.^[5] The ability to access the protection channels of a ring via the extra traffic inputs of an APS or ring terminal is later—in [Section 11.7](#) (and [Section 11.8](#))—employed a part of a strategy called "ring-mining" for ring to mesh (or ring to cycle) evolution.

^[5] An amusing, possibly apocryphal, account about the use of the extra traffic feature is the story that "Hockey Night in Canada" used to be distributed nation-wide during Stanley Cup playoffs as extra traffic on protection channels of backbone digital radio and fiber systems. Hockey enthusiasts would not have been pleased.

3.4.5 AIS Concept

One of the features of transmission systems that helps isolate the location of failures is the generation of Alarm Inhibit Signal (AIS), also called Alarm Indication Signal. The idea is to suppress the propagation of the loss of valid signal in one section of a transmission system from setting off the alarms all the way down the rest of the path. OXC nodes can also perform AIS insertion. To illustrate the concept, consider two OXC nodes in a mesh network connected by a WDM transmission system with several OAs. If one of the OAs has a catastrophic failure, or if a cable cut occurs, the adjacent OXC nodes will register the physical loss of signal and insert an "AIS" on the surviving directions on the failed paths. AIS is a standardized dummy payload structure such as a fully framed SONET signal, but with "all-ones" payload that is easily recognized. As a dummy payload of proper framing, timing, power level, etc., it suppresses downstream alarms and indicates to each downstream node that the signal path has failed upstream but that another node has already realized this. AIS is relevant to survivability strategies in at least three regards:

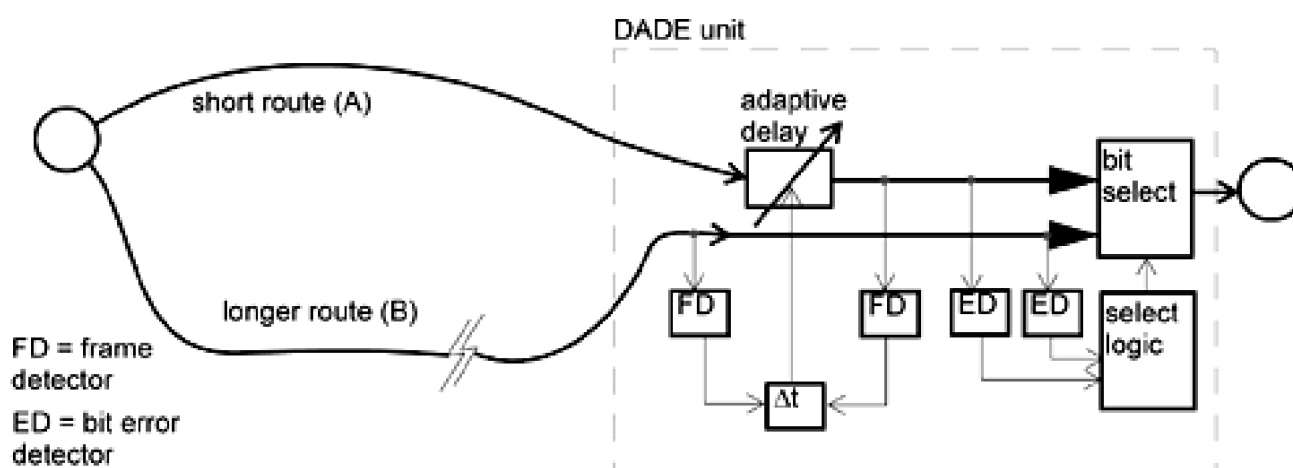
1. For span restoration AIS techniques ensure that only one pair of custodial nodes are activated.
2. In end-to-end path-oriented survivability schemes, the appearance of AIS at path end nodes is what initiates restoration.
3. In some strategies for mesh restoration, the appearance of AIS on a working channel anywhere in the network can indicate that the channel can be taken over as equivalent to spare capacity for restoration. This is part of the later concept of "stub release" in [Chapter 6](#).

3.4.6 Hitless Switching

"Hitless switching" is a special technique that can be used in conjunction with 1+1 DP APS so that a cable cut would not even cause a single *bit* error. This is not often implemented in practice as the costs are high and seldom is such an extreme performance guarantee really required. However, hitless switching has often been specified in a "wish list" sense in at least the first drafts of the APS subsystem requirements in transmission system designs. More often it has been implemented on digital radio to hide the effects rather frequent 1+1 APS switching actions in combatting multipath fading. It is of interest to recognize this scheme, as it defines the ultimate quality of system-level hiding of physical layer disruptions.

The technique uses an adaptive delay buffer switched into the shorter path of the two (1+1) signal paths at each receiver. In conjunction with a suitably long masterframe alignment sequence, the receiver in [Figure 3-7](#) frames on both incoming signals and adaptively delays the signal copy that is arriving with less propagation delay (A) to bit-align it in time with the later arriving signal copy (B). If signal A is considered the normal working signal and if the buffer delay is greater than the alarm detection time for a loss of signal on either signal feed, then it is possible to switch from signal A to signal B at the buffer output *before damaged bits from A reach the output*. Alternately an error checking code on each signal path can give a byte, column, or frame-by-frame level selection between delay aligned A and B outputs. Both of these schemes would be realizations of hitless switching in which not one bit error occurs during protection switching. The delay alignment process is called differential absolute delay equalization (DADE). DADE has been employed in digital radio systems to combat fading but seems not to have been used on fiber systems to date.

Figure 3-7. The ultimate: 1+1 DP / DADE: "Hitless switching" (only one direction shown).



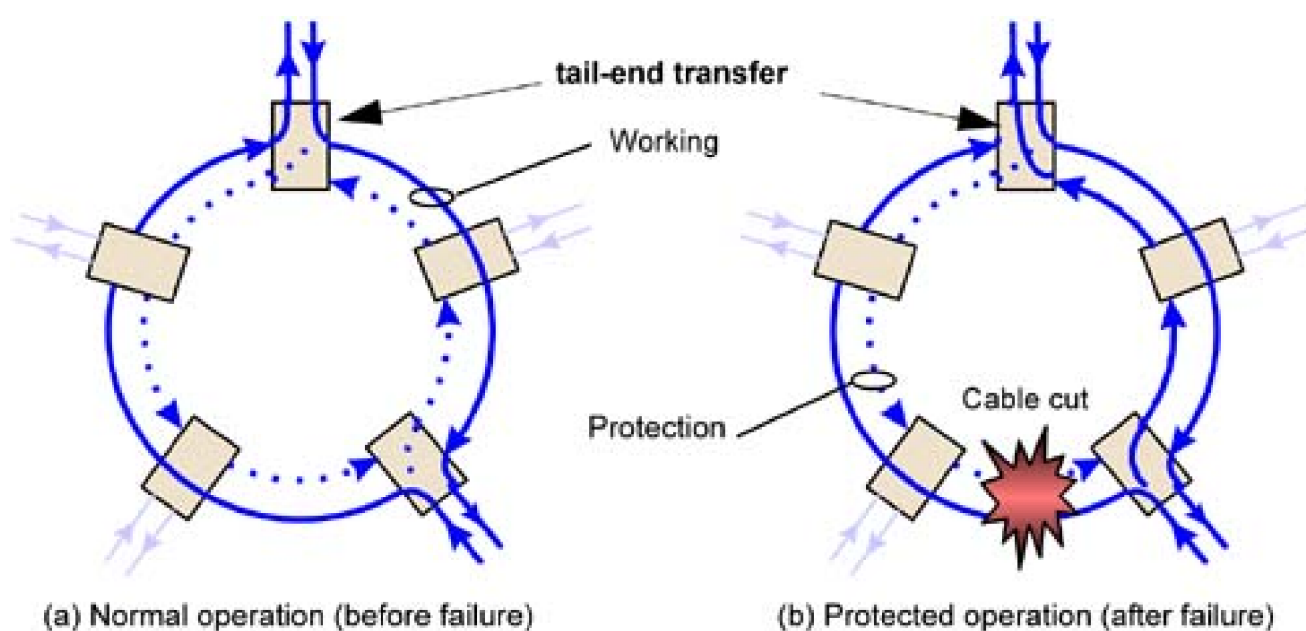
3.4.7 Unidirectional Path-Switched Rings (UPSR)

Ring-based transmission systems are an evolution from APS systems. It is perhaps easiest to see this with the UPSR which can be seen as packaging up of several logical 1+1 DP APS systems to share a common higher-speed transmission system. A drawback of "standalone" 1+1 DP APS is that the optical line signal (single-fiber OC-n or a DWDM waveband for instance) on each fiber is delivered to the other end as a complete unit, without the possibility to add or drop individual STS or wavelength channels at intermediate locations. Another issue is that unless a large point-to-point demand exists, a dedicated 1+1 APS transmission system may not be justifiable economically. We need some way to fill these large transmission capacities to exploit the economy of scale in cost vs. capacity for transmission. This is in essence what the UPSR [\[Bell95\]](#) does. To understand the role of the UPSR, imagine a number of nodes exchanging, say, single STS-3c demands. Conceptually one could serve each demand with its own 1+1 DP arrangement with OC-3 transmission systems. But if an OC-48 transmission system costs, say, only four times that of an OC-3 system, then it would be better to combine all the individual 1+1 DP requirements to take advantage of a single proportionately cheaper OC-48 transmission technology. This is precisely the idea of the *unidirectional path-switched ring* (UPSR).

Thus UPSRs comprise a number of logical 1+1 APS systems on a set of nodes aggregated onto a common closed-loop path that provides each with the disjoint A and B signal feeds for 1+1 DP APS operation. Nodes in a ring are connected by equal-capacity working and

protection fibers (or fiber pairs) in a closed loop or cycle. The diverse route for every working system is the remaining part of the ring of which it is a member. Each unit-demand is transmitted in opposite directions around the ring on both the working and protection fibers. As in 1+1 APS, the receiving node independently selects the better of the two received signals and has no need for signaling to any other nodes. A UPSR also requires just two fibers. Nodes X,Y exchange a bidirectional working demand pair by virtue of X sending clockwise to Y and Y doing likewise, sending to X around the remainder of the same-direction ring. Thus, each bidirectional signal exchange (X to Y plus Y to X) completes a path around a whole unidirectional ring. ADM nodes are connected by a single working and protection fiber, each of which transmits the line signal in the opposite direction. (Really under UPSR the distinction between one fiber as working the other as protection is arbitrary and it is also a per-channel attribute, not an overall system attribute.) Figure 3-8 illustrates. Under normal conditions, the demand between pairs of nodes in the ring is transmitted on the working fiber in one direction around the ring. A copy of each demand is also transmitted on the protection fiber in the opposite direction. At the receiving node, a path selector continuously monitors the working and protection signals and switches from the working to the protection fiber when the working signal is lost or degraded. Protection switching decisions are made individually for each path rather than for the entire line. SONET standards call for a protection switching time less than 50 ms after detection of signal loss or degradation in the UPSR. This UPSR specification seems to be the general source of the belief that *all* rings give 50 ms switching notwithstanding that BLSR systems in general *do not* necessarily provide 50 ms switch times.

Figure 3-8. UPSR protection switching operation.



An important capacity-planning principle about the UPSR is that because (when considering both A and B signal feeds) the working signal for each demand pair is transmitted all the way around a UPSR, the total demand on any span is the sum of all the demands between all nodes on the ring. This implies that the UPSR line transmission rate must be greater than the sum of all demands served by the ring (regardless of the specific demand exchanged by each node-pair). That is:

Equation 3.1

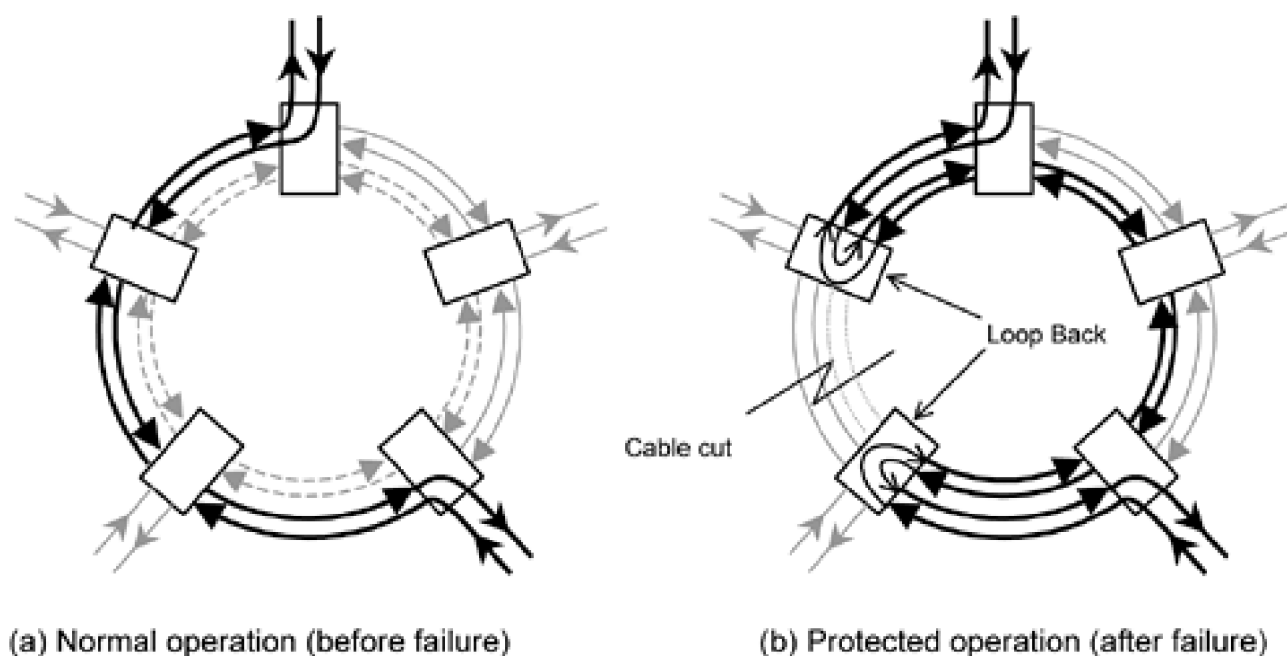
$$UPSR_Line_rate \geq \sum_{\forall (i,j) \in \mathbf{D}, i \neq j} d_{ij}$$

where \mathbf{D} is the demand matrix and d_{ij} is the demand from node i to node j . Conversely this means the total demand served cannot exceed the number of channels (i.e., time-slots or wavelengths) provided by the ring. SONET OC-48 and OC-192 UPSR rings have been fairly widely deployed for metro area customer access applications where the hubbed nature of the demand pattern on such a ring makes it basically as efficient as otherwise generally more efficient the BLSR. Recently the UPSR logical structure has been implemented in DWDM technology where each channel comprised a lightpath or a waveband. The optical version is called an Optical Path Protected Ring (OPPR). In Europe the UPSR logical structure is often called an SNCP-ring standing for subnetwork connection protection ring.

3.4.8 Bidirectional Line-Switched Rings (BLSR)

A more efficient arrangement under general demand patterns is obtained when a linear multi-point SONET 1:1 nested APS system is closed on itself. One then obtains what was initially called a shared protection ring (SPRing) and now referred to in North America as the bidirectional line-switched ring (BLSR) and in Europe as multiplex-section protected ring (MSPRing). A basic reference documenting the BLSR is [Be1195b]. Unlike the UPSR which uses receive path selection, *anyline-switched* ring protects demands by looping the entire working line signal back onto the protection fiber system at both nodes adjacent to a failure. This is a bit more like a 1:N APS in that access to the protection facility must be coordinated at both ends of the failure and signaling is involved. Unlike 1:N APS, however, the protection fiber system has to have equal capacity to the working system so as to be 100% restorable. In a 4-fiber bidirectional line switched ring (BLSR/4), a separate pair of bidirectional fibers is used for working and for protection. Working demands are not permanently bridged to the protection fiber. Instead, service is restored by looping back the working demand from the working fiber to the protection fiber at the nodes adjacent to the failed segment, as shown in [Figure 3-9](#).

Figure 3-9. BLSR protection switching operation.



A failed segment may include a span, a node, or several spans and nodes. The SONET K1, K2 line-level overhead bytes perform signaling in the SONET BLSR. Because the protection fiber passes through one or more intermediate nodes before reaching its destination, addressing is required to ensure that the APS message is recognized by the proper node and protection switching is initiated at the right pair of nodes. For this purpose, the SONET reserves four bits in the K1 byte for the destination node's ID and four bits in the K2 byte for the originating node's ID. Thus, the maximum number of nodes in a SONET BLSR is 16. A two-fiber bidirectional line switched ring (BLSR/2) operates in the same logical fashion as a BLSR/4 except over predefined working and protection channels groups on each bidirectional fiber pair. As an example which also shows the direct extension to DWDM, a logical BLSR/2 system using 12 wavelengths per fiber might define wavelengths 1 through 6 to protect wavelengths 7 through 12 on the reverse direction fiber. Standards for DWDM optical rings can be expected to emerge just as for SONET rings but the DWDM version of the BLSR has so far been called the optical shared protection ring (OSPR).

An advantage of BLSRs over UPSRs is that channels can be reused around the ring and the protection bandwidth is shared among all working span sections. Because a demand occupies a channel only between its entry and exit nodes, and is usually routed on the shortest path between nodes on the ring, the same channel can be reused for other demands on unused spans of the given path. Since the demands travel directly between the entry and exit nodes (and not all the way around the ring, as in a UPSR), the load on any one span is the sum of demands that are routed over that span. Thus, because the line rate of the BLSR is the same on all spans, the required capacity (i.e., the line transmission rate) of the working and protection rings (or channel groups in the case of a BLSR/2) must be:

Equation 3.2

$$BLSR_Line_rate \geq \max_{k \in S}(w_k)$$

$$\text{where } w_k = \sum_{\forall (i,j) \in D, i \neq j} d_{ij} \cdot \delta_{ij}^k .$$

w_k is the total demand routed over span k based on the choice of bidirectional routing made for each demand, i.e., either clockwise or counterclockwise around the ring. For [Equation 3.2](#), the indicator parameters δ_{ij}^k allow calculation of these span-wise working capacity totals, which unlike the UPSR, may differ on each span. $\delta_{ij}^k = 1$ if the route chosen for demands on node pair (i,j) crosses span k , otherwise it is zero. Simply put, the capacity of the protection ring has to meet or exceed the largest total of the demand flow crossing any span of the ring. A consequence of this is that (unlike the UPSR) the total demand serving capacity of a BLSR is dependent on the demand pattern and the routing choices for each demand on the ring. In a pure-hubbed demand pattern, BLSR efficiency is no better than a UPSR. At the other extreme, the ideal (but essentially fictitious) pattern of demand for a BLSR is where all non-zero demands are exchanged only between adjacent nodes, and all exchange equal demand totals. Under these ideal conditions the BLSR reaches its best possible redundancy of 100%. In random mesh-like demand patterns BLSRs can be between 200 to 300% redundant (i.e., the ratio of total protection plus unused working span capacity to the total working capacity required using the shortest paths on the graph for the demands served).

Two key issues in planning and operating networks based on BLSR rings are the capacity-exhaustion of a span and the related concept of "stranded capacity." When a single span exhausts (i.e., reaches full working capacity utilization) the whole ring is in effect exhausted with respect to its ongoing usefulness to take up more growth. The condition of equality in [Equation 3.2](#) is then reached on at least one span of the ring. At that point no further demands can be routed through the ring in a way that would cross that span. The side effect of this can be that other spans that have remaining working capacity which is "stranded."

For more details of both UPSR and BLSR as well as APS systems, in both SONET and ATM contexts, see [\[Wu92\]](#) [\[WuNo97\]](#).

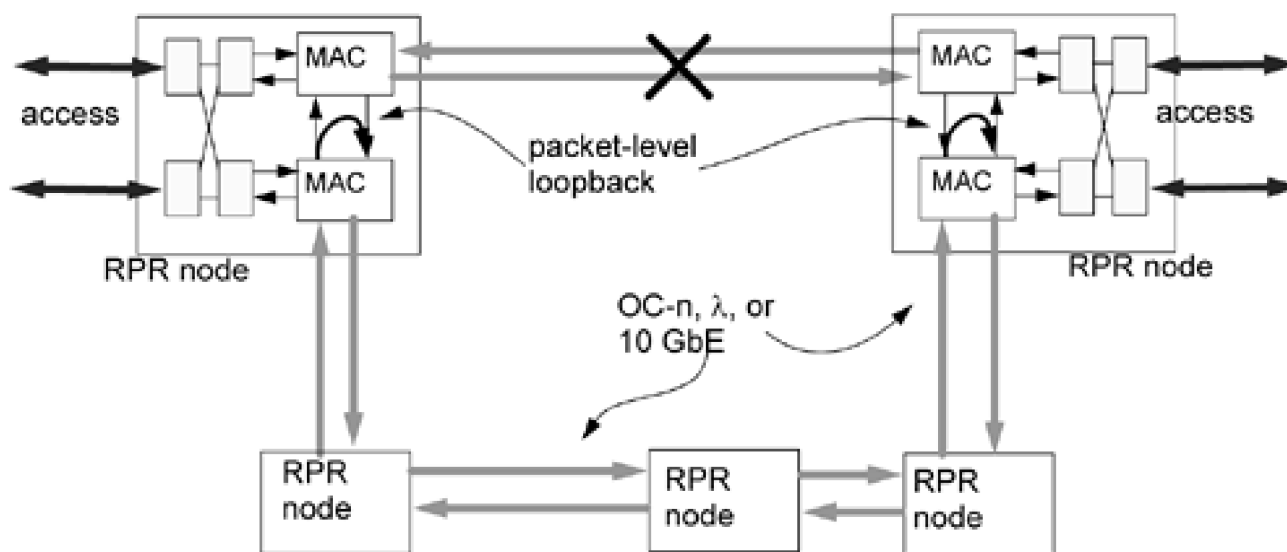
3.4.9 Resilient Packet Rings (RPR)

With the dominance of IP packet data as the primary traffic type, a packet-oriented evolution of the BLSR has been developed [\[Cisc99\]](#). When considering packet data flows, the channelized nature of the line capacity of SONET rings, where capacity units are parceled out to each node pair, restricts the maximum burst rate available to any one data connection on a rigidly channelized ring. It is also fairly management intensive to establish a full set of logical point-to-point OC-12 connections within, say an OC-192 BLSR ring. Data-centric applications may also want to have many more than 16 nodes on the ring and leave the total line transmission capacity available for sharing among all data sources. The 100% reservation for protection capacity is also seen as inefficient when data applications could be using that capacity as well, simply for added performance, during non-failure times. These are some of the motivations for the IEEE 802.17 initiative to develop a Resilient Packet Ring (RPR) standard.

An RPR is a kind of "hollowed out" 2-fiber BLSR. It uses OC- n , wavelength, dark fiber or other physical layer options for bidirectional transmission on each span and a line-level loopback mechanism for protection. It is "hollowed out" in the sense that circuit-like channelization of the line capacity is abandoned and the entire line-rate transmission capacity is available at each node for packet access. The BLSR attribute of capacity reuse on spans is also achieved by the RPR under a spatial reuse protocol (SRP). Under SRP, destination nodes strip packets off the ring rather than the source node following a full ring transit, as in prior token-ring and FDDI ring standards.

The nodes of an RPR are essentially routers that connect LANs, enterprise data centers, web server farms, etc. to the ring via an SRP Media Access Control (MAC) interface that terminates each bidirectional span. [Figure 3-10](#) illustrates. The access routers can decide to send new packets onto the ring via either of its line transmit interfaces. This inserts new packets onto the ring in either the long or short directions to their destination so load leveling is facilitated. In the receive direction each MAC unit can either receive a packet destined for it or forward packets en route. Normally a received packet is stripped off the ring but for multicast it can be received and forwarded as well. A pass-through mode also allows express flows to be defined that are essentially invisible to the RPR node, passing directly through at the link layer.

Figure 3-10. Resilient Packet Ring: a combined Layer 2–Layer 3 packet ring survivability technique (amenable to the over-subscription based capacity planning method of [Chapter 7](#)).



Failures are handled by "wrapping" the ring at both nodes adjacent to the failure. This is conceptually the same as the BLSR's loopback mechanism but the wrapping happens by re-direction at the packet level, inboard of the MAC interfaces (rather than at the line signal level as in the BLSR). This allows protection status to be allocated selectively by priority or by other traffic attributes. Obviously, if both counter-rotating fibers are in use for traffic in normal times, the wrapping for protection imposes a sudden additional packet load on the surviving ring spans. Unlike a BLSR, the performance of an RPR under a fiber cut is therefore a matter of *oversubscription* based planning of protection capacity, a topic introduced in [Chapter 7](#). At first instance, it appears that there could be up to 100% oversubscription of capacity in the wrapped state, but the packet-level congestion effects depend on the actual utilization of capacity at the failure time, the mix of protected and unprotected flows, and the adjustments that the SRP fairness protocol will make, as well as the backoff effects that user application protocols may undertake. RPR can thus exploit the soft degradation of services to avoid needing a 100% reservation of strictly unused protection capacity.

For the same reasons, RPR is not easily classified as purely a system layer survivability scheme. It uses a link-level packet loopback mechanism as well as service-level means to accomplish the overall recovery from failure. In reference to the classical data protocol layering it is therefore often referred to as a combined layer 2 and 3 (L2/3) scheme.

3.4.10 Ring Covers

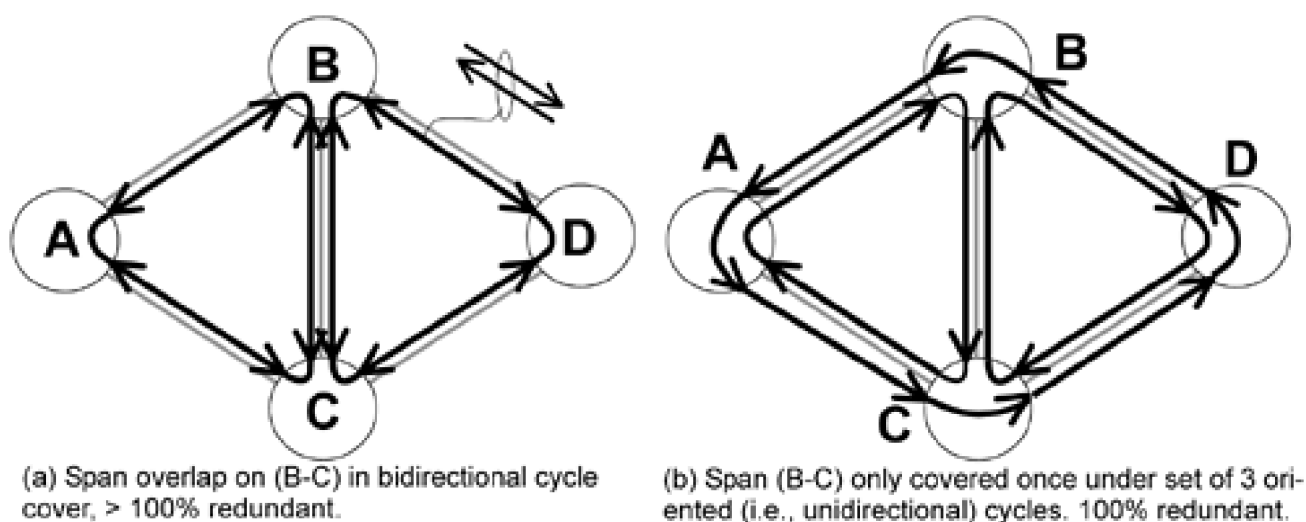
A single BLSR ring is often said to be 100% redundant in the sense that each pair of bidirectional working fibers (or channels) is exactly matched by a protection fiber or channel pair. This simple definition is not entirely adequate because it disregards the fact that because of the ring construct asserted on the routing of demands, it may not be possible to usefully fill each span of such working fiber. In practice, when whole transport networks are designed with multiple interconnected rings, the total installed capacity is usually much more than two times the capacity needed only to route all demands via shortest paths over the graph (i.e., the standard working capacity in the sense of [Section 1.5.3](#)). Such networks are thus much more than 100% redundant as a whole. Three of the main reasons for this are:

1. Demand routing has to follow ring-constrained paths, not shortest paths over the graph, so demands take longer routes in the first place than they otherwise would, even before their subsequent matching with 100% protection is considered.
2. A set of rings that overlies each span with at most one ring only on each span is usually not possible. In other words, ring covers usually involve some span overlaps.
3. When the working capacity on one span of a ring is filled, it blocks still available working capacity on other spans of the ring from being used for routing.

Unidirectional ring covers are a way of addressing the second of these contributing factors: the problem of span overlaps in trying to lay out a set of rings on the graph. A span overlap is a span whose working capacity could be handled by one ring alone but for purely topological layout reasons the solution requires two rings (each with their own protection) to both overlie that span. The inefficiency of such overlaps can be avoided using unidirectional rings instead of bidirectional ring covers. Formally the technique involves finding an *oriented cycle double-cover* (O-CDC) of the graph [ElHa00]. If we temporarily ignore the other two factors above which bear on the true redundancy of ring networks, the O-CDC principle can achieve ring-type network protection with exactly 100% redundancy in the restricted sense we mentioned of that being the simple ratio of working to protection fibers or channels over the network as a whole.

To explain this let us start by considering an ordinary (i.e., bidirectional) cycle cover, which essentially represents a ring network design based on the span coverage principle. The network demands are routed over working fibers on each span and an overlay of dedicated protection bidirectional fiber pairs, connected in cycles, "covers" each span. If that span fails, the working fiber pair is looped back onto the protection fiber pair, just as in a BLSR. Figure 3-11 shows an example. To cover every span with at least one BLSR ring (or more generally any type of bidirectional cycle which would include a UPSR), the two required cycles unavoidably overlap on span (B-C), in Figure 3-11(a). It is easy to see in general that anywhere an odd-degree node is involved, an ordinary bidirectional cycle cover (where the two directions are locked together on the same cycle) will not be possible without at least one span overlap. The problem with such overlaps is they lay down two working fibers and two protection fibers on a span where (as previously postulated) working demand flow requires at most one of each. With a single overlap, such a span is effectively 300% redundant instead of 100% redundant (in the simple sense of counting working to non-working fiber or channel ratios).

Figure 3-11. Showing how oriented unidirectional cycle covers can avoid the span overlaps that occur in bidirectional cycle covers or BLSR multi-ring network designs.



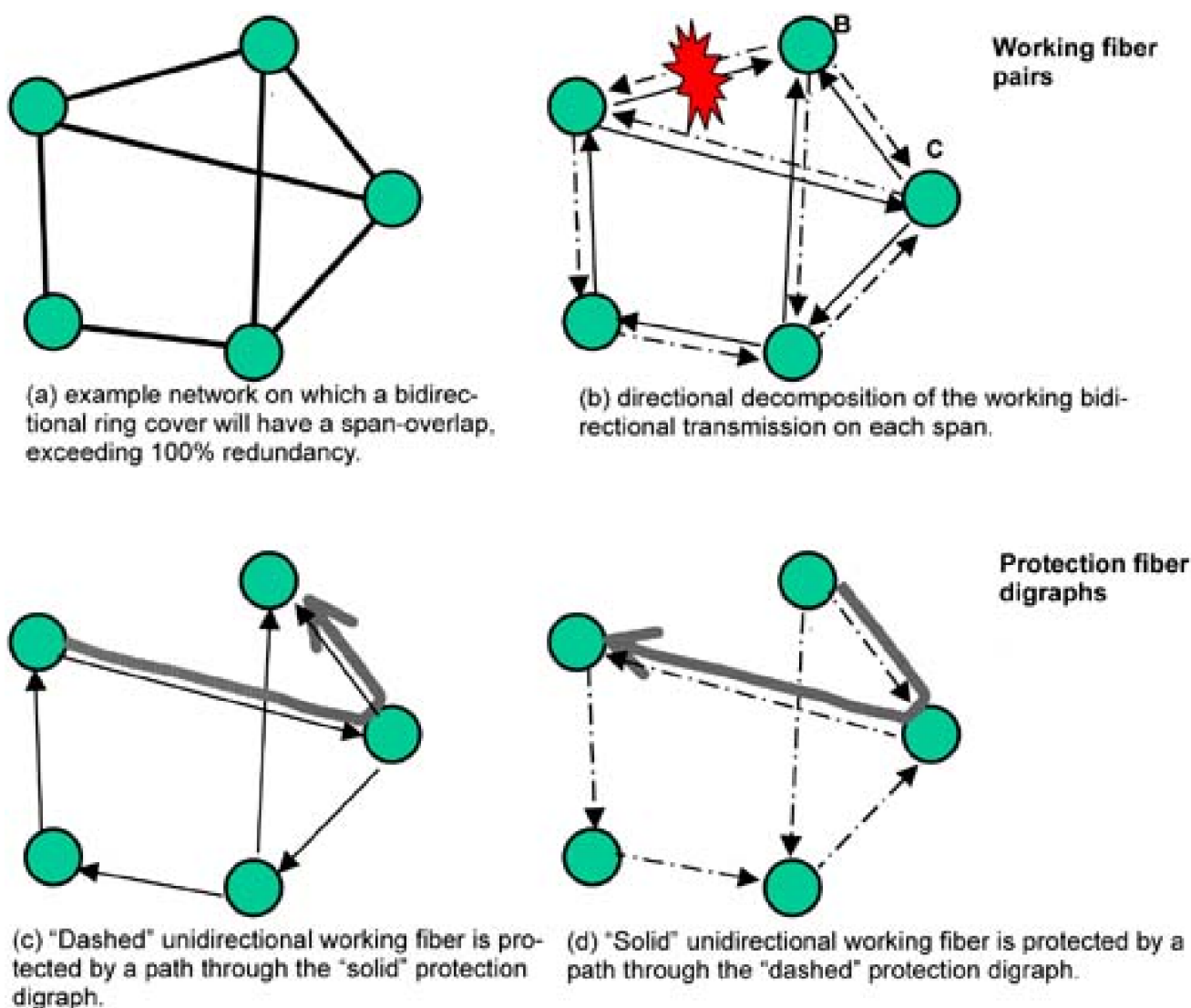
The motivation behind *directed* or "*oriented*" cycle double covers is to at least improve the situation relative to conventional ring covers, by avoiding such overlaps in planning the cycle covers, so as to get to exactly 100% redundancy (in the sense above) over a network as a whole. Figure 3-11(b) shows how the overlap can be avoided if a set of three unidirectional cycles are used instead of two bidirectional cycles. The total capacity provisioned in (a) is 12 fiber-hops whereas in (b) it is only 10 because the double coverage of span (B-C) is avoided by the unidirectional cycles. Thus with no overlaps and exactly one protection link for each working link on each edge of the graph (or wavelength or waveband level as applicable) we arrive at a class of networks that are exactly 100% redundant in terms of protection to needed working fiber counts. If planned at the whole-fiber level, this gives rise to "4-fiber" networks which have exactly two working and two protection fibers on each span. The practical idea behind this is that with, say, up to 128 or 256 wavelengths per fiber it becomes possible to consider networks that are based entirely and uniformly on just four fibers per span. The 100% redundancy is not retained, however, if some spans need multiple fiber pairs, while others do not.

The key result in [ElHa00] is to show that with an *oriented* cycle double cover the resulting designs can be exactly 100% redundant at the fiber level. More detailed explanation is deferred to Chapter 10 where OCDCs are compared to *top*-cycles. For reference, other works on the problem of ring-covers include [GHS94], [KeNa97].

3.4.11 Generalized Loopback Networks

"Generalized loopback" networks (GLBN) were introduced in [MeBa02] and are conceptually related to OCDs. As with OCDs the basic idea is to eliminate the use of bidirectional rings or bidirectional cycle covers, while arriving at an overall design that is exactly 100% redundant at the fiber level (or waveband or wavelength level) on every span. Like OCDs, a GLBN also assumes a uniform "4-fiber" logical span model: one bidirectional working fiber pair is assumed adequate for all capacity on each span, and a matching protection fiber pair is also provided on each span. Thus, each span level cross-section is 100% redundant in the same (limited) sense as we would say a 4-fiber BLSR is 100% redundant. The difference from OCDs is that the protection and working fibers of a GLBN are not preconnected into (unidirectional or bidirectional) rings. Instead, a simple flooding-type protocol finds and cross-connects a single replacement path through the protection fibers upon failure. Figure 3-12 shows a small network on which it is impossible to avoid having at least one span-overlap under a bidirectional ring cover, and illustrates how a GLBN works without any predefined cyclic structures.

Figure 3-12. How generalized loopback works to avoid needing more than 100% redundancy in a 4-fiber span protection environment.



The idea is to divide the bidirectional flow of working demands over the basic graph into two directed graphs (digraphs) where each graph has only one directed working flow on each of its edges. To do this each direction must be assigned (appropriately, not arbitrarily) to one or the other working digraphs, as in Figure 3-12(b), where the digraphs are denoted "dashed" and "solid." Once this is done, a protection copy of each working digraph is identically defined. Each of the directed "primary" (i.e., working) graphs is then protected by the protection copy of the other primary digraph. For the failure in Figure 3-12(b) the node that was normally transmitting on the "dashed" working link, now loops back onto the "solid" protection digraph (Figure 3-12(c)), and vice-versa at the other end of the failed span Figure 3-12(d). The transmission from the nodes next to the failure is actually a flooding copy of the working signal into all outgoing fibers of the anti-directional protection digraph. Other nodes also flood but under a protocol that stems off arrivals of duplicate copies so that the single shortest replacement route in the directed protection graph is all that results.

Not every initial assignment of the two directional working links of each original bidirectional link into the two primary digraphs works to protect all failures, however. In fact the directional decomposition of [Figure 3-12\(b\)](#) works for the failure shown, but not for span (B-C). The key to assignment of the working link directions into the two primary digraphs is that each digraph must remain a connected graph, meaning that at least one *directed* path must exist between all nodes. In [Figure 3-12\(b\)](#) the "solid" digraph is not connected because node B has no path from itself to other nodes on "solid" edges. Medard et al. [[MeBa02](#)] give an algorithm for the assignment of directions that ensure the required properties based on finding a directed cycle that visits all nodes.

The result is a network with four fibers on every span with exactly two working and two protection fibers which are like the spans of a single 4-fiber BLSR in that every span consists of a bidirectional working fiber pair and a matching pair of backup fibers. Unlike a BLSR, however, protection inherently takes more generalized routes over the equal-capacity backup network, rather than being restricted to following a particular ring structure. Generalized loopback networks are thus in effect "BLSN"s, where "N" stands for *network* instead of ring and the efficiency they offer relative to a typical 4-fiber BLSR-based network is the removal of the overlapping ring spans that are usually unavoidable in ring-planning. In other words, exactly 100% matching of working and protection resources is achieved, but not worse.

[Table 3-3](#) summarizes the schemes so far discussed, all of which are "ring-like," as defined by virtue of having 100% or higher redundancy. In [Table 3-3](#) logical redundancy refers to the ratio of total non-working fibers or channels to working.

Table 3-3. Overview of ring-like schemes for network protection at the system layer

Logical Redundancy	Scheme or Principle	Logical Equivalences	Notes
> 100%	1+1 DP APS		Basic parallel hot standby redundancy model (head-end bridged)
> 100%	1:1 DP APS	UPSR, SNCP	Permits extra traffic on standby
> 100%	UPSR	OPPR, SNCP	Modularized assembly of 1+1 DP APS arrangements
> 100%	BLSR	OSPR, SRING, MSCP ring	Nested linear APS arranged in a closed loop
> 100%	FDDI	ULSR	Unidirectional ring LAN with "wrapping"
> 100%	Cycle cover	Protection fiber pair ring overlay	BLSRs of dark-fiber pairs protecting working fiber pairs
exactly 100%	Generalized Loopback	Unidirectional fiber-level span protection	Like directionally planned SR (without flow spreading) in 1+1 ("4-fiber") span model
exactly 100%	Oriented CDC	Unidirectional planning of shared dark-fiber rings	Assumes "4-fiber" network model

3.4.12 System Layer Protection Without 100% Redundancy: p -Cycles

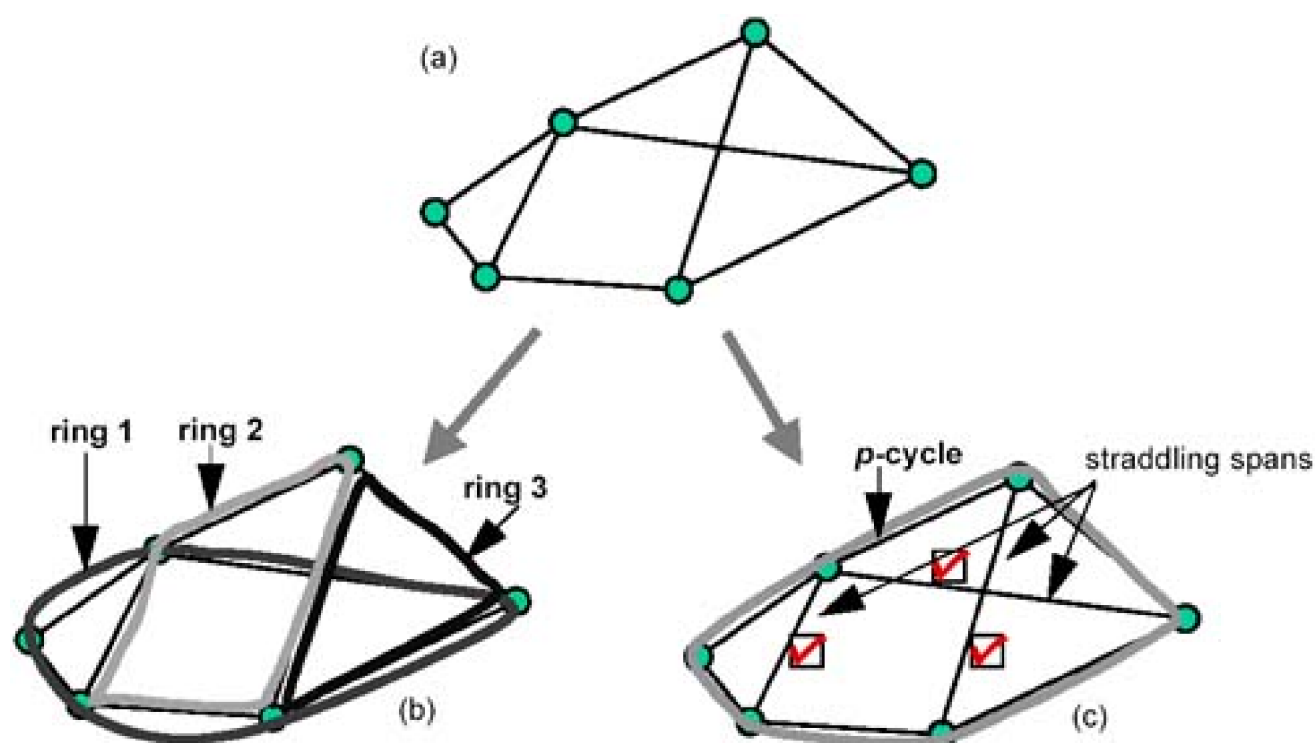
p -Cycles provide another technique that is applicable at the system level using nodal devices that are counter-parts to the ADMs of conventional rings, but p -cycles break below the 100% "redundancy barrier" that characterizes other system level techniques. We introduce it here as a system level protection option, but due to its mesh-like capacity efficiency and its applicability at either logical or system layers, we later summarize it with the family of mesh-type schemes. p -Cycles seem to be a unique approach in the sense that they have applicability as a system layer solution, but are mesh-like, not ring-like, in their spare capacity requirements. It is the only system level protection technique that does not require a direct matching of working and protection fibers on each span and is typically well under 100% redundant in both logical and true redundancy measures (i.e., the "standard redundancy" of [Section 1.5.3](#)).

The easiest way initially to think of p -cycles is as a BLSR to which the protection of *straddling* failure spans is added. A straddling span is one that has its end-nodes on the p -cycle, but is not itself part of the p -cycle. Rather, it is like a chord on a circle. The usual ring-like

loopback protection of spans on the ring itself still applies but are called "on-cycle" failures, to distinguish them from *straddling span* failures. Upon the failure of a straddling span the p -cycle (which is undamaged in such circumstances) provides two protection paths. The simplest and most important distinguishing feature of a p -cycle-based network, compared to rings or cycle covers of any kind and generalized loopback networks, is the protection of straddling spans which themselves may bear *two* units of protected working capacity for each unit of capacity on the p -cycles and require *zero* spare capacity on the same spans.

[Figure 3-13](#) illustrates how a p -cycle provides single-span failure protection to a small network without covering all spans directly. [Figure 3-13\(a\)](#) shows the example network and we assume that all spans shown bear working capacity to be protected. [Figure 3-13\(b\)](#) shows how rings or any (conventional) fiber-level cycle cover can provide such protection, although we note in this example that it is impossible to use fewer than three simple cycles to do this. The particular ring cover shown is actually an efficient one because it matches up odd-degree nodes to share the same span overlaps (four odd-degree nodes are handled with two span overlaps). An O-CDC or GLBN solution can do better than the bidirectional cycle cover shown but will still be 100% redundant because each span failure is protected by a cycle in a ring-like way or by a directional loopback that forms a cycle with the failed span. In each case the protection is by a direct covering of cycles.

Figure 3-13. A network (a) with a (bidirectional) cycle cover or ring cover (b) in comparison to a single p -cycle providing the same protection against single-span failures (c).



But the same set of single-span failures can be protected by a single p -cycle, as in [Figure 3-13\(c\)](#). (Optimal p -cycle designs will not in general use only one cycle, although that is adequate using the Hamiltonian cycle shown in this example.) Should any of the spans on the p -cycle fail, the p -cycle acts just like a BLSR, protecting against on-cycle failures through loopback to protection on the same cycle. The failed signals reverse away from the break and go the other way around the cycle. If any of the three straddling spans shown fail, the same p -cycle is broken into at the end points of the straddling failure span and actually can provide a protection path *both* directions around the p -cycle. For that reason, the efficiency of protecting straddling failures is twice that of an on-cycle failure.

In a sense the addition of straddling failure protection is only a minor technical variation on BLSR rings: [Chapter 10](#) shows the nodal structure in more detail, but the nodal elements remain almost as simple as ring ADMs and the switching function needed is exactly that of the BLSR. Nonetheless, even in a network as simple as [Figure 3-13\(a\)](#) it can be seen that the difference in protection efficiency can be dramatic. The ring cover is assuredly over 100% redundancy, because it has three (unavoidable) span overlaps. The ring cover shown actually protects nine working hops with $5 + 4 + 3 = 12$ ring protection hops. Purely on a fiber-count basis the cycle cover is then $12 / 9 = 133\%$ redundant.

In contrast the p -cycle uses six hops of protection capacity, but protects up to $6 + 2(3) = 12$ hops of working capacity making it only 50% redundant. The true efficiency relative to rings is actually higher than in this simple comparison because with p -cycles the working demands also go via shortest paths over the graph; only the protection structure itself is formed in a cycle. It has been shown in general that p -cycle based networks are essentially as efficient as span-restorable mesh networks to follow [i@chapter 5](#) yet, as will be detailed

further in [Chapter 10](#), they require only ring-like, fully pre-configured switching actions known in advance of any failure. [Chapter 10](#) is dedicated to p -cycle based network planning and implementation at either the system or logical layer where they can be hosted on OXC-nodes and dynamically reconfigured for demand pattern adaptation.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

3.5 Logical Layer Survivability Schemes

System layer protection schemes all rely on essentially fixed transmission and/or protection structures. An advantage of this is that once installed and tested, such systems are discrete identifiable network substructures, their operation is relatively simple and self-contained (i.e., they don't involve a highly general reaction over the network at large), and the restoration path taken for any failure is clearly known in advance. The difference (relative to mesh protection at the logical layer) was once explained to the author that "transmission people are comfortable with self-healing *systems*, but self-healing *networks* are (perceived to be) too general and unpredictable for their liking."

On the other hand, system layer implementations of protection are essentially static, and this can be more to the dislike of the services, planning, and business people within the same companies. If the configuration of as-built systems turns out not to match forecast demand well, it is not at all easy to change the configuration because it is essentially determined by the hardware installation. In addition, if a first-failure has occurred, there is nothing that can be done during the period of physical repair to particularly enhance the readiness of the network to withstand a possible second failure. Fixed system layer protection schemes also do not easily support differentiated quality of protection classes: when whole systems switch at the line rate to protect failures, everyone gets the same class of protection.

This brings us to consider logical layer protection (or restoration) schemes. The flexible ability of the logical layer to create paths on demand between desired end points, out of a general inventory of uncommitted channels on transmission system channels, makes it the natural domain of a number of survivability schemes with features that are not provided by the ring or APS system layer schemes. Foremost among these considerations is the higher capacity efficiency which can be achieved by "mesh" restoration schemes which permit extensive sharing of protection capacity over non-simultaneous failure scenarios. Capacity efficiently arises not only through sharing of spare capacity but also because cross-connects in the logical layer manage capacity at a finer granularity than in the system layer. As an example, a SONET DCS might manipulate STS-1 and STS3c signals whereas a OC-192 BLSR manipulates the entire 10 Gb/s line signal as a unit for protection purposes. Similarly in an optical network the logical layer OXC will manipulate single lightpaths for routing or restoration, whereas protection actions in the system layer (and capacity allocations to go with those actions) will probably be based on whole-fiber or large waveband levels of manipulation.

3.5.1 Concepts of Protection, Restoration and Distributed Preplanning

System layer schemes are inherently all of the class we will define as protection, while logical layer schemes can be either restoration or protection schemes. Let us now make the distinction. The term *protection* derives from its origin in APS systems. In a 1+1 DP APS system, the switching actions are completely predefined and the protection system is fully connected between its end nodes and in a pre-tested, ready-to-use state. The working system is said to be protected, as opposed to restorable, in these circumstances. If a 1:1 APS is involved, then signaling is required to request the head-end bridge and to bump any "extra traffic" off of the spare span. In addition the protection system has to be tested on the fly for correct transmission of the bridged signal. However, the protection route is completely pre-defined and no cross-connections are needed to create the signal path. It follows that UPSR is the same as 1+1 DP APS and BLSR is the same as 1:1 DP APS in these regards. The term protection is generally used for all these as a category. The main difference in *restoration* is really only that the replacement paths that will reroute the payload signals may have to be found and/or cross-connected in real-time when the failure occurs. Thus, one can say that in a *pure protection* scheme, the backup paths are completely dedicated and ready to bear rerouted working demand flow. And in a *pure restoration* scheme all redundant resources are held in a shared pool until configured on demand for restoration against a specific failure that arises.

Having identified these general distinctions it is important to stress that logical layer implementations of mesh-based survivability can be *either* protection schemes or restoration schemes. Possibly for competitive reasons, the classification of schemes as either "protection" and "restoration" has become rather over-emphasized, and over-simplified and coupled with an almost axiomatic assertion that "protection is fast and restoration is slow" and that the most efficient OXC-based mesh-survivability schemes are all "restoration" schemes. All of these points need to be sorted out. We hope to convey in this overview of the issues, and further in [Chapters 5 and 8](#), that these views are overly simplified and dogmatic. In fact the best possible arrangement for survivability may be the combination of a distributed restoration mechanism embedded in the logical layer which self-generates efficient mesh network protection preplans to withstand any first-failure and then executes directly providing best-efforts state-adaptive restoration to a second failure, should it arise.

The basic assumption that needs to be challenged is that a process of finding paths in real-time is always slow and that if the replacement paths are known in advance they will always be fast. Neither are necessarily true as generalizations, especially with distributed preplanning to identify paths with a restoration mechanism *in advance* of any actual failure. Moreover, there are really at least *three* basic categories of scheme to consider of which pure protection and pure restoration are only the extremes. The two extremes and intermediate possibilities are detailed in [Table 3-4](#). A somewhat similar categorization appears in [\[IBo03\]](#) which also surveys a taxonomy of variations between pure protection and pure restoration. The intermediate schemes are in some ways the most promising in terms of combining efficiency and speed and how fast or slow these schemes are is dependent on whether path finding or path cross-connection time dominates, not whether these intermediate schemes are *classified* as protection or restoration. In the *intermediate* category are schemes where the restoration paths are fully known before a failure, but spare channels are not cross-connected until a specific failure arises. In this regard neither span restoration using *distributed preplanning* (SR-DPP) nor shared backup path protection (SBPP) can be classified as simply a protection or a restoration scheme.

Table 3-4. Three Fundamental Classes of Survivability Scheme

Type	Description	Examples	Generic Term
(a)	Pure Protection: Protection routes are known in advance and cross-connection is not required to use them: spare capacity is preconnected and needs only be accessed at end-points.	1+1 APS, SNCP ring, UPSR, OPRP, BLSR, OSPR, <i>p</i> -cycles	Protection
(b)	Pure Restoration: Restoration routes are found adaptively based on the failure and the state of the network at the time of failure; connections to assemble the restoration paths are also made in real-time.	Distributed or centralized adaptive restoration algorithms or distributed restoration algorithms (DRA).	Restoration
(c)	Intermediate: Replacement routes are known in advance and cross-connection maps for fast local action are in place at all nodes but cross-connection is required to assemble the restoration path-set in real-time.	Distributed preplanning with Span restoration (SR-DPP), ATM Backup VP, shared backup path protection (SBPP).	Preplanned Restoration

SR-DPP is an especially powerful technique in that, even if path finding is slow, *distributed preplanning* (DPP) can create (and frequently update) protection preplans that are already in place in the nodes in advance of failure. DPP works by using a series of mock-failure trials responded to by a distributed restoration algorithm (DRA) embedded in the network or other restoration protocol. For each "dress rehearsal" nodes simply record the local set of cross-connections that constituted their participation (if any) in the assembly of restoration paths for each failure trial. The concept is described more fully in [\[Gro94\]](#) or [\[Gro97\]](#) and in [Chapter 5](#). It is a simple technique that retains all of the generality and database freedom of a distributed restoration algorithm, but provides a "protection" scheme of the intermediate type in [Table 3-4](#). This always-present relationship between restoration and a corresponding preplanned "protection" scheme, derivable through DPP, must be kept in mind. Moreover, it is fundamental that if one solves any variety of spare capacity planning problems for different classes of mesh survivability schemes, that spare capacity can either be accessed adaptively by a restoration algorithm in the real network (which gives certain extra tolerances for error) or, the same capacity planning solution for the restorable network can be used to provide a set of preplanned protection arrangements to be used in each node. Finally, regardless of whether any mesh-based survivability scheme operates in real-time with preplanned protection reactions, or with an adaptive restoration algorithm, there is no difference at all in the capacity required or in the definition of the capacity-planning problem (assuming the restoration algorithm is fully capable in the required path finding role).

The relative speed of the intermediate schemes (following a failure) depends on what dominates the real-time performances: *path-finding* time or *cross-connecting* time. Schemes of type (c) can approach the speed of pure protection if OXC cross-connection is fast, occurs in parallel at all nodes, and, through distributed preplanning (DPP), the protection routes and all local switching actions are completely known in advance. Upon failure, real-time is consumed only for failure notification. All nodes put their most recently preplanned actions into effect, in parallel. In span restoration with distributed preplanning on fast OXC nodes, the most dominant time delay could be the simple dissemination of fault notification. As soon as nodes learn the failure identity they assert an already known, locally stored, spare capacity cross-connection map into effect. This happens in parallel at all networks nodes as soon as notification arrives. In another type of intermediate scheme route-finding can take significant time but "assembly" of the path is virtual and takes essentially zero time. e.g., CR-LDP "redial": once label distribution is complete, there is essentially zero subsequent path establishment delay per se.

Thus, we need to appreciate the range of possibilities between pure protection and pure restoration, but avoid the oversimplifications associated with these categorizations, particularly regarding speed and availability. SR-DPP and SBPP are in particular important intermediate schemes for which no categorical statement about relative speed is really justified other than if based on a detailed implementation study. Depending on the relative speed of path finding to cross-connection either scheme may even approach the speed

of pure protection in the same network. A final point related to this discussion is how to refer to spare capacity designed into a network for either protection or restoration purposes. For brevity we will make no further distinction and refer simply to *spare* capacity whether used for protection or restoration.

Let us now return to our overview of restoration or protection schemes that operate in the logical layer. To guide the overview we introduce [Table 3-5](#). Because the book itself is devoted to in-depth treatment of the mesh-based survivability schemes, we will not go into the same depth introducing them here as we did for rings, ODCs and GLBNs, which we will not be covering further.

Table 3-5. Overview of Logical Layer Mesh-type Survivability Schemes

Scheme or Principle	Short Description or Equivalences	Notes
Span restoration (SR)	Dynamic k -shortest paths	Uses a "DRA" Chapter 5 or [Gro94]
Span protection (SP)	Shared-protection routes preplanned	Centrally controlled or self-organized by distributed preplanning with a DRA
Meta-mesh (MM)	SR in meta-mesh graph, loopback in chain subnets	A hybrid between span and path restoration (Chapter 5)
I-based p -cycles	Like a BLSR that also protects straddling spans	ADM-like system level or OCX managed p -cycles (Chapter 10)
Path restoration (PR) (with stub release)	MCMF with limited commodity requirements	Theoretically most efficient possible scheme (Chapter 6)
I-based SBPP	1:N APS sharing arranged over disjoint failures	(Chapter 6)
I-based SLSP	SBPP on redefined sub-path segments	"Short leap" shared backup protection: overlapped SBPP sub-path setups
GMPLS: OSPF-TE / CR-LDP	Independent path reprovisioning attempts by all affected pairs	No assured speed or recovery level

3.5.2 Span Restoration or Span Protection

In span restoration, restoration paths (or preplanned protection paths) reroute locally around the break, between the nodes of the failed span. In pure span restoration the paths are both found and connected in real-time. Span protection refers to a network operating either as outlined above with DPP-based protection preplans or through centrally computed and downloaded preplans. This type of scheme is sometimes also called link protection or line restoration.

3.5.3 Meta-mesh

Meta-mesh is a variation on span restoration that enhances the spare capacity efficiency of span restoration in sparse network graphs. It involves a combination of ring-like ADM loopback within subnetworks that are chains of degree-2 nodes and mesh-like planning of capacity for restoration flows over the logical higher-degree skeleton of a network containing many chain sub networks. It represents a specific partial step toward path restoration. [Chapter 5](#) is devoted to in depth treatment of span restoration including meta-mesh.

3.5.4 p -Cycles

p -Cycles were introduced as a system layer technique where they would use modular-capacity nodal elements similar to an ADM and implemented at the whole-fiber or waveband level. However, because the p -cycle concept separates the routing of working flows from the configuration of protection structures (not locking these two together as in rings), p -cycle based protection is also amenable to logical layer implementations. In this context the OCX nodes can set-up and take-down service paths as demand requires and separately configure and maintain a set of span-protecting p -cycles. Such p -cycles are established and managed at the logical channel, rather than system level and can be easily changed to adapt to shifting demand patterns. Multi-service priority schemes for access to p -cycles for protection can also be fairly easily implemented in a logical layer implementation of p -cycles but not in the system layer. [Chapter 10](#) covers p -cycles.

3.5.5 Path Restoration

In path restoration (PR) the capacity design and corresponding rerouting problems are posed as multi-commodity maximum flow-like rerouting problems to replace affected paths end-to-end following removal of the failed span from the graph. This may or may not involve conversion of surviving working capacity of failed paths into capacity that is available for use in restoration, an aspect called "stub release." This type of path restoration with stub release is of special theoretical significance because it represents the most efficient possible class of survivable network.

3.5.6 Shared-Backup Path Protection (SBPP)

A related method that is particularly amenable to IP-centric control of optical networks is called shared-backup path protection (SBPP). A prior scheme for ATM VP-based transport networking works in the same logical manner and can also be used for MPLS path protection. The approach in SBPP is simplified relative to path restoration by defining a single fully disjoint backup path for each working or "primary" path. In effect, a 1:1 DP APS arrangement is established at the level of each service path. This simplifies real-time operation as the protection response is independent of where the failure occurs on the corresponding working path (whereas the PR response is failure-specific). Relatively high efficiency is still achieved even though a 1-for-1 APS setup exists because spare capacity is shared over failure-disjoint backup paths. [Chapter 6](#) is devoted to path restoration and SBPP schemes. [Chapter 7](#) treats the application of SBPP to the MPLS layer or ATM VP layer transport where oversubscription-based planning of protection capacity is possible to considerably reduce overall capacity requirements.

3.5.7 Segmented or Short-Leap Shared Protection (SLSP)

This is a variation on SBPP in which SBPP-like shared backup protection paths are set up over several segments or sections of the path, rather than end-to-end over the entire length of the working path. This accommodates a working path that may need to travel through several pre-defined protection domains. More generally, division of any working path into segments for protection produces a family of options between the extremes of pure span protection and SBPP. When a primary is protected with segment-wise disjoint paths, not end-to-end with a single backup path, availability is improved and protection arrangements can be managed locally within each domain the primary crosses between entry and egress nodes of that domain. By further defining protection domains to overlap, single point exposures to node failure are avoided where the SBPP segments would be otherwise connected in tandem through a single node. The idea of segmented interlaced backup paths was introduced in [\[KrPr00\]](#) and later applied to lightpath protection in [\[HoMo02b\]](#), [\[SaMu02\]](#), [\[GuPr03\]](#). The concept of *segment-based rerouting* was also studied in [\[JoSa98\]](#) where it was similarly recognized that with ATM backup-VP protection, availability would degrade for long service paths. The methods for SBPP design in [Chapter 6](#) cover SLSP when used on a transformation of the initial demand matrix which converts end-to-end path requirements into apparent demand between the designated segment protection nodes.

3.5.8 GMPLS Automatic Reprovisioning as a Restoration Mechanism

For completeness we should also recognize that GMPLS is being viewed by some as offering a network restoration mechanism in addition to its primary role of provisioning transport paths. The thinking is that since OSPF-TE will eventually produce an updated global view of the topology and available capacity following a failure, then each node will be in a position to begin re-establishing those paths which it lost in the failure using GMPLS to simply "redial" each of their lost connections, over the shortest route following the failure.

It is important to note in this regard that SBPP uses GMPLS to establish a working path, and a corresponding disjoint backup path, but it does so for each path as it is provisioned and ahead of the failure. It thus effects a preplanned protection arrangement that is cognizant of the physical spare capacity present and of failure-coordinated contention or sharing relationships that have been established on each unit of spare capacity. This is significantly different than the direct reliance on OSPF-TE/CR-LDP in real-time following a failure to attempt simultaneous re-establishment of all failed paths. Direct use of OSPF-TE/CR-LDP following a failure involves no preplanned reservation or sharing arrangements for capacity for a backup path. There are therefore considerable drawbacks to direct reliance on GMPLS auto reprovisioning for restoration. All other schemes involve considerations to coordinate or preplan the access to spare capacity for an effective and fast recovery following failure. By effective we mean that guarantees about the restorability level can be made by design. While GMPLS auto reprovisioning would usually be an effective response to isolated failure of a single lightpath, relying on the same method for recovery from a cable cut would seem almost irresponsible. A recovery response of some form would result but there is essentially no control or assurances that can be given about the duration and effectiveness of the overall recovery pattern that results.

When a cable is cut a large number of independent asynchronous instances of the topology update and path redialing protocol will be triggered. Recovery actions must first wait until the OSPF-TE global view is synchronized in each node. As soon as it is, there will be a mass onset of CR-LDP signaling instances as individual end-node pairs attempt to re-establish their failed paths. Each acts without coordination with the others doing so concurrently. OSPF-TE updates will continue to be generated as the available capacity state changes on each link as CR-LDP seizures of capacity occur. This causes other CR-LDP instances to fail or be initiated with out of date resource information and destined to have to crank back. The overall dynamics of possibly thousands of concurrently activated signaling, capacity seizure, and update dissemination protocol instances, and how they will interact to allocate the available capacity for protection, and how fast the whole process would settle down is quite uncertain. Even without considering signaling contention and fall-back dynamics, it is theoretically impossible to say what restoration level will be achieved because a finite-capacity multi-commodity rerouting problem is being attempted by a greedy (and mutually interfering) set of routing instances. The theoretical issue, treated in [Chapter 4](#) and further in [Chapter 6](#), is that of "mutual capacity"—where one path-finding instance with several choices may, when acting independently, take capacity that makes paths for many other pairs infeasible. This can occur even when there is sufficient capacity for full restoration under more coordinated routing. Thus, GMPLS auto reprovisioning may provide a useful built-in reaction for isolated path failures, but because there can be no assurances about the overall level or distribution of the recovery pattern for the set of paths that fail simultaneously under a cable cut, we do not treat it as a restoration method for use in the logical layer. This technique is more suited to use in the service layer.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

3.6 Service Layer Survivability Schemes

Service layer techniques are the last safeguards before physical failures become apparent to user applications and are usually worth having in addition to a lower layer scheme. Unlike lower layer schemes in which costs are incurred for extra ports and explicit protection capacity, service layer schemes are usually software-based implementations that attempt rerouting within the working, but only partly utilized, capacity that is visible at the service layer. A service layer rerouting response can also complement a lower layer response if the latter is incomplete, by logical reconfiguration of its paths, and/or application of service priorities to reduce delay or packet loss.

In addition, a service layer *node* failure, or interface failure on a switch or router, can be best dealt with among the peer layer network elements in the same service layer itself. Unlike methods at the logical or system layers which tend to be very fast-acting but all-or-nothing in terms of their benefit for any given path i.e., they either fully protect the traffic-bearing signals (so the effect is invisible) or do not (so the effect is total outage), service layer methods are generally more gradual and provide a shared "graceful degradation"-type of network response. Typically, blocking or congestion and delay levels may rise, but a basic functionality continues. Thus, except for extreme cases, service layer restoration methods tend to prevent hard outage per se, trading a performance degradation instead.

[Table 3-6](#) identifies a number of options for service layer survivability. Dynamic routing in circuit switched networks and link-state adaptive routing schemes, such as OSPF in the Internet are the two most traditional service layer schemes. With the advent of an IP-centric control plane, several of the logical layer schemes, in particular SBPP and *p*-cycles, have direct correspondents for use in the service layer as well. The main difference is only that a physical circuit like path entity in the data plane is replaced with a virtual path construct such as a VP or LSP. With IP-centric protocols an essentially identical control plane implementation can establish these service layer constructs, just as GMPLS constructs transport layer constructs. In an MPLS/IP service layer, label-switched paths just replace lightpaths in the prior descriptions of logical layer SBPP and *p*-cycles. Other more service-specific forms of restoration are also possible in the services layer. For instance, circuit switched telephony networks have long-used centralized adaptive call routing (called dynamic routing or dynamic non-hierarchical routing (DNHR)) to re-calculate routing plans in the face of congestion [[WoCh99](#)], [[Topk88](#)], [[Ash91](#)], [[IEEE95](#)]. And, of course, in all data networks, message retransmission and adaptive routing protocols apply. These, and the basic ability of OSPF to update its routing tables following link withdrawal LSAs, are all possible forms of service layer restoration mechanisms, as well as GMPLS auto reprovisioning of MLPS paths. The same basic proviso applies that, by itself, mass independent reprovisioning attempts by every affected end-node pair will have no assured or predictable outcome. But when used in the services layer to complement a logical layer restoration response (if needed), there is much less concern, because any auto reprovisioning activity in the service layer is, in that context, understood as only a best-efforts activity to improve performance following a logical layer response.

Table 3-6. Schemes for Service-Layer Restoration or Protection

Scheme or Principle	Short Description	Notes
MPLS <i>p</i> -cycles	IP-link protecting <i>p</i> -cycles formed using LSPs	Conceptually same as span-protecting <i>p</i> -cycles in logical layer but formed in MPLS layer and amenable to oversubscription based planning (Chapters 7, 10)
Node-encircling <i>p</i> -cycles	<i>p</i> -cycles formed as LSPs to protect against node (router or LSR) failure	<i>p</i> -cycles for which all flows through a node are straddling flows hence restorable in the event of <i>node</i> loss (Chapter 10)
MPLS SBPP	Equivalent to ATM Backup VP Protection	Oversubscription based capacity design (Chapter 7)
MPLS SLSP	SBPP on redefined sub-path segments	Short leap shared protection on LSPs with overlapped SBPP sub-path setups
OSPF (for routed IP flows)	Routing table reconvergence	No assured recovery level if used without a lower layer scheme, uncontrolled oversubscription
OSPF-TE / CR-LDP (for label-switched paths)	Independent LSP "redial"	No assured recovery level if used without a lower layer scheme, uncontrolled oversubscription

Scheme or Principle	Short Description	Notes
Dynamic call routing	Centrally recomputed alternate routing policies	Minimizes circuit-switched trunk group blocking

Dynamic routing schemes for circuit switched networks are an evolution of alternate routing in teletraffic networks wherein a direct "high-usage" trunk group would be supplemented by shared overflow "final routes." The routing of individual calls is determined at call setup time by first testing the direct route and then possibly one or more alternate routes, subject to loop-avoidance constraints. Dynamic routing schemes today follow this basic pattern but are centrally managed with a typical period of about 10 seconds between updates to a central site on the traffic levels on each trunk group from each site. Centralized algorithms can then update the outgoing first and/or second choice trunk group recommendation at each node based on the destination of calls it is handling. The centralized recommendations are able to take into account the current congestion states in various parts of the network, thereby inherently diverting traffic flows around areas of failure. A main benefit of adaptively updating the routing tables is exploitation of the non-coincidence of busy hour loads in the network.

Note that such updates to the routing plan do not imply rerouting existing connections. The aim is to improve the situation for *new* call (or packet or LSP) arrivals only. This is a natural approach in a pure data or telephony service layer where calls or sessions come and go on a minute-by-minute time-scale, and where users can re-establish their calls if need be and where data protocols retransmit lost data packets. Thus, the aim and approach is to seek adaptations that improve aggregate performance, without too much concern about the fate of any particular call or session. This is in contrast to the lower layer restoration environment where the emphasis is on re-establishment of existing paths, which may be in existence for years, and which may bear the entire traffic between two cities, rather than a single call or data session.

In circuit-switched services efforts may also be made to split the realization of the single logical trunk group between two nodes over physically diverse paths. In addition, optimization algorithms can be used to slightly overprovision the trunk quantities in each group as a general margin against failures or congestion. Other attractive aspects of service layer restoration in general is that different priority statuses for various users or services may be much more easily established and finely assigned. In addition, capacity is managed at a much finer scale so that small amounts of available capacity in larger working channels units manipulated by the lower layer schemes can be accessed to enhance performance.

Service layer schemes may also involve establishment of a full logical mesh of trunk groups or MPLS paths, or a full mesh over a subset of key nodes, so that routing of any connection is not through more than one other intermediate node of the same service layer. Such high-degree logical connectivity is possible because the resources to support this in service layer networks are essentially virtual, i.e., VPI numbers or LSP labels, etc., not physical cables and routes. A consequence of this, however, is that when a high-degree logical network is established over a sparse physical network there can be escalation or expansion of one physical cut into a large and hard to predict number of logical link failures in the service layer, making it rather uncertain to rely solely on a service layer restoration scheme. This is when it is especially useful to make sure a system or logical layer scheme is in place to hide the whole event from the service layers. This is later referred to as the "fault escalation" issue.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

3.7 Comparative Advantages of Different Layers for Survivability

The layered view we have just worked through allows us to see that survivability measures at each layer are for the most part complimentary, not competitive. Physical layer measures are essential and service layer measures always help. And we should always have at least one technique implemented at the system or logical layers but there is really no need to employ both, especially if cost is considered. An important planning decision is thus whether to employ a system layer or a logical layer recovery scheme. Two of the main factors in this decision are flexibility and efficiency. With rings, or 1+1 diverse routing, there will be an investment of over 100% in redundant transmission capacity because (by definition) both diverse routes cannot be equally shortest routes. With logical layer mesh alternatives this may often be reduced to 50-70% redundancy. In addition, complete flexibility exists with an OXC-based (i.e., logical layer) implementation (1) to adapt the protection stance to changing demand patterns, (2) to evolve the entire protection strategy from one scheme to another and/or, (3) to implement prioritized protection service classes. A summary of other comparative aspects is offered in [Table 3-7](#).

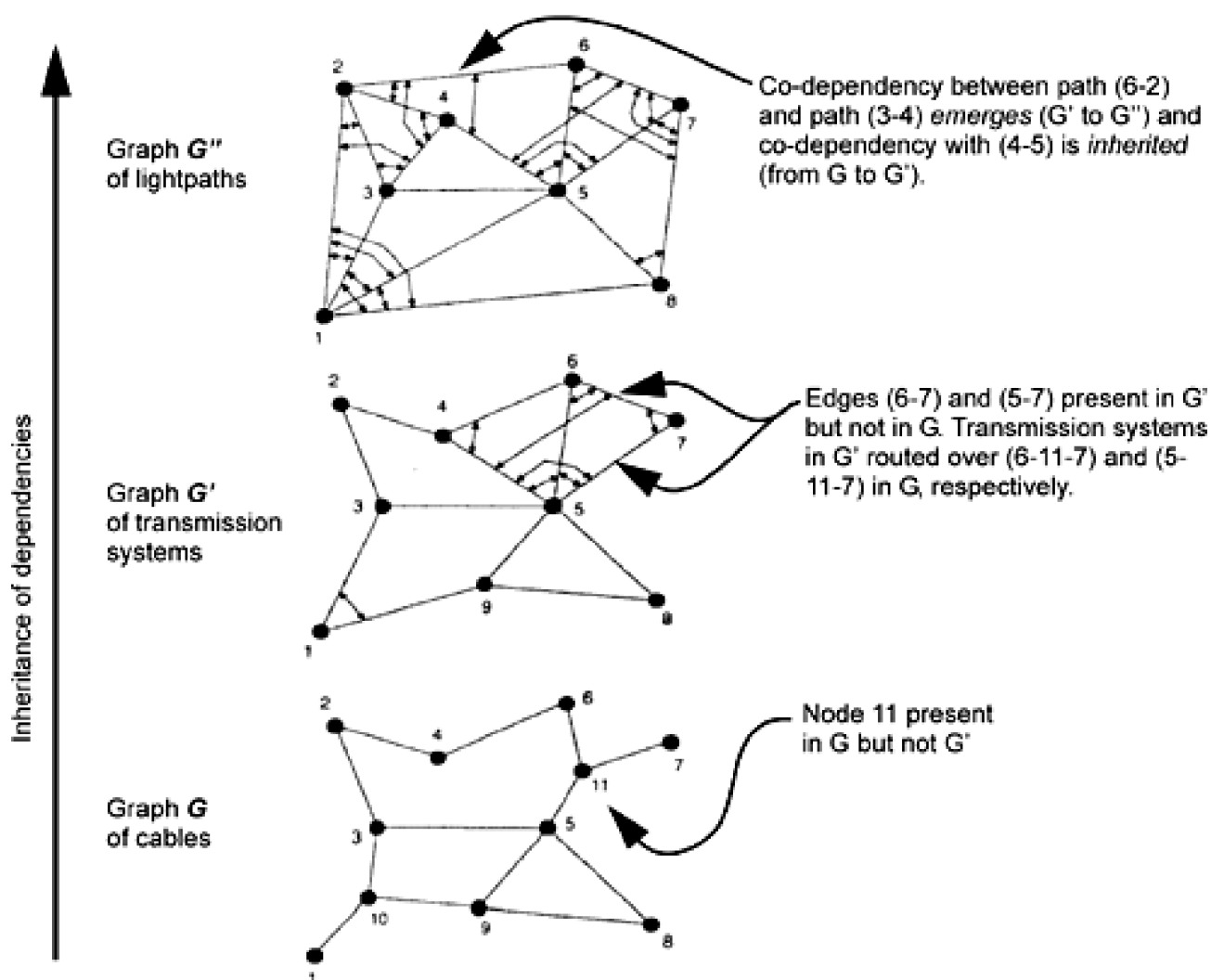
Table 3-7. Comparative Strengths and Weaknesses of Layers for Survivability

Attribute	Transmission System Layer	Logical Cross-Connection Layer	Services (or IP Transport) Layer
<i>example:</i>	<i>BLSR Rings</i>	<i>Span Restoration</i>	<i>MPLS SBPP</i>
Capacity Required	Highest	Middle	Least
Speed	Highest (~50 ms)	High (~ 100–300 ms typ.)	Slowest (seconds–minutes)
Certainty / predictability	Highest	High	Lower
Multiple Quality of Protection (QoP)	None	Easily supported on per path basis	Easily supported
Provisioning view (working)	Ring-constrained shortest path	Shortest path	Shortest path coordinated to be disjoint with protection
Provisioning view (survivability)	Inherent once routed	Checked upon shortly after routing	Coordinate protection sharing arrangements network-wide
Degradation characteristics (if restoration fails)	Abrupt and total outage	Abrupt on affected channels, may be partial	More graceful degradation; congestion not outage
Oversubscription strategies	No	SONET or WDM: no ATM VP: yes	Yes
Customer control	Least	Through VPN services	Most
Database and protocol dependencies	Least: a "hardwired" implementation	Little: event-driven protocols in firmware interacting on overhead bytes, network state is database	Highest—large databases of global network state, dissemination protocols, software dependent
Susceptibility to SRLG effects and fault escalation	Least, controlled during planning	Low, especially with adaptive distributed restoration	Highest vulnerability to SRLG effects and physical-to-logical fault expansion

Multi-Layer Protection: Containing the Inheritance of Dependencies

In thinking about the different layers where we can implement survivability, the issue of physical to logical fault multiplication is critical. Adequate knowledge of SRLG relationships may be extremely difficult to obtain (or maintain) if there are several steps of the emergence and inheritance of failure dependencies, to use the terms introduced in [OePu98]. At every layer of routing abstraction, new fault dependencies *emerge* and are *inherited* by all higher levels. The growth in complexity of determining physical diversity between paths as one goes higher up the hierarchy from physical toward service layers is conveyed in Figure 3-14, based on [OePu98]. Graph G shows the layout of cables which in this case involves some degree 1 nodes. As mentioned, a first step is to create a biconnected physical graph. Doing so in this example would remove some of the dependencies in G' emerging from G, but not all. Even when G is biconnected, dependencies between transmission systems are impossible to avoid as long as the systems are allowed to pass through nodes without terminating. They are especially frequent if least-cost routing of each system is desired. For example, we could close G with respect to stub-node 7 by adding a cable (7-8) but transmission systems (6-5) and (7-5) would likely remain dependent because span (5-7) in G is on both of their shortest routes. Observe also that node 11 is a junction in the cable graph but has no corresponding appearance in the higher level logical graphs. This is the classic case of a common duct (here, 5-11) creating dependency between what are otherwise viewed as separate transmission spans (6-5) and (7-5) in G' . When one routes lightpaths over these transmission systems, still further dependencies emerge where lightpaths share transmission systems and the prior dependencies from the cables to systems layer are inherited.

Figure 3-14. Illustrating the fault dependencies that emerge and are inherited by higher levels (adapted from [OePu98]).



The example in Figure 3-14 goes up only two layers above the cables and considers only eight top-layer nodes. In practice if the G'' shown is the lightpath service layer, then service paths at the STS-3c level routed over them have at least one or two more layers of dependency emergence and inheritance. The overwhelming impression, extrapolating from Figure 3-14 as a simple example, is that it may be difficult to give a robust assurance of full survivability against a cable cut if operating higher up in the hierarchy. Diverse STS-3c level paths would be able to protect the corresponding service against same-level failures or a single lightpath failure (one layer down), or perhaps also a single transmission system failure (two layers down). For example, STS-3c level 1+1 diversity can protect against an STS-3c interface

port failure on the host router or against a lightpath failure (including access multiplexing) one layer below. But at three layers of reach-down (to the cables) it seems far less plausible that we would always be certain that STS-3c primary and backup paths would have no inherited dependencies.

In practice this a compelling reason to use protection strategies at the service layer and at either the system or logical layers. Diversity measures at one level can realistically be expected to protect against single failures with known dependencies one or two levels below, or at the same level, but it is probably unrealistic to expect a services layer diversity mapping to retain complete physical disjointness more than two layers below. One set of options to consider is rings, p -cycles, or mesh protection implemented with whole-fiber cross-connects directly over a biconnected physical cable graph, G . In this case there are no emergent dependencies to be inherited by higher layers since each cable span becomes a directly protected single-failure entity. This is simple, robust, and requires relatively low-cost devices for whole-fiber protection switching. It is not very fine-grained, however, and the devices used for protection have no secondary use such as for dynamic service provisioning (other than provisioning dark fiber services).

Alternately, logical layer measures implemented in G' using cross-connects for mesh-protection at the channel level are more agile, multi-purpose, and fine-grained in capacity-handling and only have to cope with one level of known dependencies, such as arise in G' from the cable junction node 11 in G . Using SBPP in G'' (instead of a protection scheme in G') is not infeasible, but we see a major complexity associated with this alternative because now we are two layers above the level at which physical faults occur—so the complete map of dependencies is far more complex. To go yet another layer up and rely on MPLS autoreprovisioning or MPLS-level SBPP, it becomes hard to imagine that we could support a claim of protection (implemented at that level) against failures stemming from the physical layer, because G is a full three levels below the MPLS layer. This all suggests a practical principle that the emergence and inheritance of SRLG-like effects may need to be "contained" by an appropriate protection arrangement every *two* layers. If followed, this leads to a strategy of choosing some basic "infrastructure" protection scheme at the system or logical layers and complimenting it (possibly only for high priority service paths) with an additional technique at the corresponding service layer itself. For example system level p -cycles and service level 1:1 APS would be one combination. Lightpath level SBPP complimented by MPLS layer p -cycles would be another viable combination, and so on. Note in this regard that even in an ideal "IP over WDM" network, the *three* layers (G , G' , G'') in [Figure 3-14](#) all still exist. Where the reduction of levels occurs in IP over WDM is actually in the levels above the lightpath layer.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

3.8 Measures of Outage and Survivability Performance

Let us now introduce various quantitative measures of failure impact, given a failure occurs, and of intrinsic survivability performance in terms of the ability to resist failures in the first place. Given the impact of failures, there is growing regulatory interest in attempts to quantify the magnitude of the impact of various failures that occur. The notion is that hurricanes, tornadoes and earthquakes each have a system for classification of their severity, so why not network failures too? Network operators are also interested in such standardized measures for quality improvement and competitive processes. A second sense of "measuring survivability" is to ask about those intrinsic properties of a network that by design make it less likely to sustain an outage in the face of failures within itself. These are the basic notions of reliability and availability and what we define as the restorability. Let us touch on these in sequence.

3.8.1 McDonald's ULE

McDonald [[McDo94](#)] was perhaps the first to advocate development of quantifiable measure of network outages. McDonald's argument was that any drive toward a standard method for quantification for network failures would focus attention on the issue and inevitably lead to improvements in avoiding outage. His proposed measure is the "User-Lost Erlangs" (ULE) defined as:

Equation 3.3

$$ULE = \log_{10}(E \cdot H)$$

where E = average historical traffic intensity (in Erlangs) through the outage period and H = outage duration in hours. The measure is logarithmic, like the Richter scale. 10 Erlangs blocked for one hour is 1 ULE. 10 ULE is equivalent to an hour-long outage affecting 100 Erlangs of normally offered traffic, or 6 minutes of outage on 1000 Erlangs, etc. The logarithmic nature is a key idea for its utility. McDonald argues a logarithmic measure discriminates well between events of major and minor consequences. And it reflects a plausible belief that the overall societal impact somehow scales with the exponent of the total outage. We would add that it is also appropriate to avoid false precision: the data going into a ULE calculation will at best be estimates, so what is important is indeed the order of magnitude, not linear differences.

3.8.2 The (U,D,E) Framework for Quantifying Service Outages

The ULE notion was developed further with an eventual aim toward standardization in [[T1A193](#)]. In this framework the impact of a failure is assessed in terms of: Unservability (U), Duration (D) and Extent (E), called a (U, D, E) triple. The three parameters of the (U, D, E) framework are:

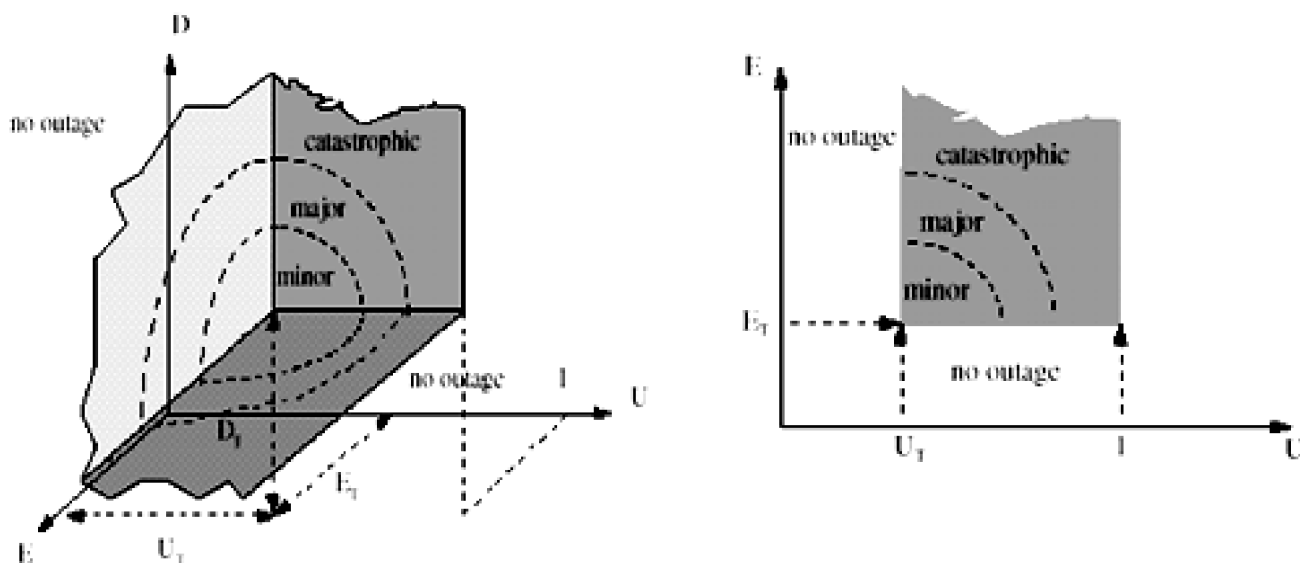
Unservability (U): is defined in terms of a basic capability and unit of usage appropriate to the application. For example, in a circuit switched network, this would be the ability to establish connections with acceptable blocking and transmission performance. The unit of usage is a call attempt and the unservability is the percentage of call attempts that fail. In a packet network, the unit of usage is a packet and the unservability is the percentage of packets that were not delivered within a stipulated delay. In a leased line network, unservability is defined as the percentage of DS-0, DS-1 or DS-3 leased signals that are *not* available.

Duration (D): is the elapsed time interval during which performance falls below the threshold for defining unservability.

Extent (E): reflects the geographic area, population affected, traffic volumes, and customer traffic patterns, in which the unservability exceeds a given threshold.

The idea is not to operate on U, D, E values further boiling them down to a single measure but to preserve them as a three-dimensional characterization of any outage event. A (U, D, E) triple can thus be plotted in the corresponding 3-space for classification of the event as catastrophic, major, or minor, depending on which predefined "volume shell" the (U, D, E) vector enters. [Figure 3-15](#) illustrates. It seems reasonable that some vector weighting scheme might also be agreed upon for definition of the qualifying regions. Or, conversely, a general (U, D, E) classification model would not necessarily have simple spherical shells for defining classifications unless the intent is to give strictly equal weight among U, D, E.

Figure 3-15. (U, D, E) concept for classification of network outages (source: [T1A193](#)).



(U, D, E) shells can be used both to categorize events as well as to lay down prescriptive policy for what might constitute an event requiring a company review of an incident or methods. For example, a Local Switch failure may be defined to have occurred whenever 500 subscriber lines (the extent E) are totally isolated (the definition of unservable, U) for 2 minutes or more (the duration, D), i.e., $(U, D, E) = (100, 2, 500)$.

Outage Index

Later work in the T1A1.2 committee that produced [T1A193](#) considers an approach leading to a single *Outage Index*. It is conceptually equivalent to formation of a vector weighted magnitude of the (U, D, E) triple but involves predefined nonlinear weighting curves for D, E and discrete multipliers for time of day, type of trunk affected (inter- or intra-LATA, 911 etc.) Such weightings are ultimately arbitrary but nonetheless can be fully detailed in a standardized method and then be of valuable service when applied industry-wide.

[Team LiB](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

3.9 Measures of Network Survivability

Measures of outage are different than measures of survivability. An outage is an event that arises from a failure that has actually occurred. Survivability is, however, the ability of a network to continue to provide service in the event of a failure that might arise. Survivability is, thus, an inherent attribute of the network design or technology employed regardless of if, or how often, failures actually occur. The term survivability itself is not usually given quantitative meaning but is used in general to refer to the quality of being able to keep on functioning in the face of either internal failures or externally imposed damage. The quantitative measures of survivability are necessarily more specific.

One class of such measures are called *conditional* or *Given Occurrence of Failure (GOF)* models. In these measures, each relevant failure scenario is first postulated to have occurred, then an assessment of survivability is made. These tend to be design-oriented measures since they reflect the merit of a survivable design over a pre-specified set of failure scenarios against which coverage is to be ensured but do not depend on knowledge or assumptions about how often such failures may actually occur. They deal with questions such as, "If failure x occurs, how well are network services protected from it?"

The other general class of survivability measures aim to take into account the probability of failure onset as well as the survivability response or capability of the network. These are called *Random Occurrence of Failure (ROF)* models. In contrast to the GOF orientation, ROF measures typically ask questions such as: "How likely is it that a path between nodes has an outage over x minutes in any given year?" ROF models are usually based on the assumption that failures can be characterized by random variables with given probability distribution functions and are thus closely related to the fields of reliability and availability which we will review.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

3.10 Restorability

A simple GOF-type measure that is widely used in design and characterization of transport networks is the *restorability*, also sometimes called the restoration ratio. Restorability is the most basic indication of survivability because it directly reflects the extent to which "single points of failure" have been removed as outage-causing circumstances. The biggest single step toward survivability is to eliminate single-span failures as a cause of service outage. This has a quantum effect on improving service availability as service outage can then only arise from much less frequent dual failures or node failures. As most commonly used the restorability is defined as the fraction of payload-bearing (i.e., "working") signal units that are subsequently restored, or that are topologically capable of being restored by replacement routes through the network. That is, for a specific failure scenario X ,

Equation 3.4

$$R_X \equiv \frac{\sum_{(i,j) \in X} \min(w_{i,j}, k_{i,j})}{\sum_{(i,j) \in X} w_{i,j}}$$

where (most generally) $w_{i,j}$ is the number of service paths between nodes i,j that are failed in the failure scenario X . This way of stipulating a failure scenario is totally general; any number of span and/or node failures can be represented in terms of the set X of i,j node pairs that simultaneously have one or more failed paths in scenario X . Thus the denominator of [Equation 3.4](#) can be thought of as a "total damage" sum in terms of the number of transport signal units that are severed in the failure scenario X . The numerator is the sum of what is restored (or can be restored) for each subset of failed signal units corresponding to a damaged span. $k_{i,j}$ represents the number of replacement (restoration) paths that can be provided for (i,j) . The $\min(-)$ operator ensures that no credit is given for providing more restoration than is actually needed for any subgroup of failed working signals.

One set of failure scenarios that is of particular practical interest is the set of all single and complete span failures. That is the set of all X which just one (i,j) . In this case the restorability for any one scenario $m = (i,j)$ simplifies to:

Equation 3.5

$$R_m = \min(w_m, k_m) / w_m$$

and the *network restorability* is defined as the average restorability of all working paths that are failed under each single-span failure scenario. That is:

Equation 3.6

$$R_n = \frac{\sum_{m \in S} \min(w_m, k_m)}{\sum_{m \in S} w_m} = \frac{\sum_{m \in S} R_m \cdot w_m}{\sum_{m \in S} w_m}$$

where S is the set of all spans in the network. $R_n = 1$ is often referred to as a "fully restorable" network. It is the mark of a network that can withstand any single-span failure without any service path outage. As a single figure of merit for network survivability R_n is of considerable practical interest because:

- a. The likelihood of failure scenarios containing more than one (independent) span failure at a time is much lower than a single failure.
- b. It is generally considered economically feasible (or at least necessary and reasonable) to design for $R_n = 1$ whereas it may be economically infeasible to protect against all possible multi-span or node failures by design.
- c. R_n is a property of the network design, or current network state, that is independent of any knowledge or assumptions about actual failure frequencies or mechanisms.
- d. Given the much higher failure rate of cables (outside plant structures in general) relative to node failures, achieving $R_n = 1$ by design is the most significant single step that can be taken in practice toward improvement of service availability.

A variety of purpose-specific variants from the basic definition of restorability are common. Examples are the "prompt restorability" which is the restorability level arising before a certain elapsed time from failure onset, or the "dual-failure restorability" which is as the name suggests and is considered further in [Chapter 8](#). Other measures can include prioritized demand weightings R_n . These are all valid measures as long as their specifics are fully stipulated in terms of the specific set of failure scenarios being considered and the criteria being employed to define survivability against those failures.

Restorability, and GOF measures in general, are relatively simple to compute and to understand, because they reflect simple measures of recovery levels for a specific set of assumed failure scenarios. In contrast, ROF measures can be much more involved and/or require simulation. A grounding in reliability and availability theory is required for their appreciation. Let us therefore now cover the basic concepts of reliability and availability which underlie ROF measures, and are also highly relevant to work in network survivability in general.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

3.11 Reliability

In ordinary English, "reliable" is a qualitative description, meaning that something or someone is predictable, usually available when needed, follows through on promises, etc. But the technical meaning of reliability is quantitative and much more narrowly defined [[BiAI92](#)], [[OCon91](#)]:

Reliability is the probability of a device (or system) performing its purpose adequately for the period of time intended under the operating conditions intended.

In other words, reliability is the probability of a system or device *staying* in the operating state, or providing its intended service uninterrupted, as a function of time since the system started in a fully operating condition at $t=0$. A reliability value can also be thought of as answering a mission-oriented question of the form: If the device or system was started in perfect working order at time $t=0$, and the mission takes until $t=T$ (with no opportunity for external intervention or repair), then what is the probability that the system will work failure-free at least until the end of the mission?

Reliability is thus always a *non-increasing function of time* with $R(0) = 1$ and $R(\infty) = 0$. When someone says the reliability of a system or device is a specific number, 0.8, say, there is usually some understood interval of time that is implicitly assumed. They are really saying the the probability of the device or system working that long without any failure is 0.8. More formally, the *reliability function*, also called the survivor function is:

Equation 3.7

$$R(t) \equiv \text{prob}\{\text{no failure in interval } [0,t]\}$$

and the cumulative failure distribution is its probabilistic complement:

Equation 3.8

$$Q(t) \equiv \text{prob}\{\text{one or more failures in interval } [0,t]\} = 1 - R(t)$$

Another way to think of the reliability function is as the complimentary cumulative distribution function (CDF) for the random variable that gives the time between failures. That is:

Equation 3.9

$$R(t) = 1 - \int_0^t f(t)dt = \int_t^\infty f(t)dt ; \quad Q(t) = 1 - R(t) = \int_0^t f(t)dt$$

where $f(t)$ is the *failure density function* which is the probability density function of time to failure from a known-good starting point.

Also useful is the "age-specific failure rate" otherwise known as the "hazard rate", l , in reliability. Given a population of systems or components that may fail, the hazard rate is the rate of failure per member of the group *given that the member has already survived this*

long. This itself is may be a function of time $I(t)$. An example is the classical "bathtub curve" of infant mortality, useful life, and wear-out phases for devices, which reflects age-specific failure rates). Much useful work is, however, based on the assumption of a constant hazard rate I which reflects systems during their useful life phase. The term "failure rate" or "hazard rate" are then equivalent and both terms are often used for I . But more generally I is the *age-specific failure rate per unit*, i.e.;

Equation 3.10

$$\lambda(t) = \lim_{\Delta t \rightarrow 0} \left[\frac{\text{number of failures in } [t \pm \Delta t/2]}{\text{number of systems exposed to failure} \cdot \Delta t} \right]$$

where Δt is a unit of elapsed time. Thus the hazard rate is strictly only the same as the failure rate if there is no age-dependency. When one is considering a single unit or system it follows that the hazard rate is the derivative of $Q(t)$ (which is $f(t)$) because as soon as there is one failure, there is no remaining pool from which to generate more failures). If there are a group of items being observed, however, we have to reflect the conditional probability nature of the fact that for a failure to arise in time $t \pm \Delta t/2$ the sample of the elements being considered only contains those remaining units that *already have survived* until time t ; the probability of which is by definition $R(t)$. Therefore, the hazard rate (or *age specific failure rate for per unit*) is in general:

Equation 3.11

$$\lambda(t) = \frac{d}{dt}[Q(t)]/R(t) = -\left(\frac{1}{R(t)}\right)\frac{d}{dt}R(t),$$

which is a simple differential equation from which it follows that:

Equation 3.12

$$\begin{aligned} \log R(t) &= -\int_0^t \lambda(u) \cdot du \\ \Rightarrow R(t) &= e^{-\int_0^t \lambda(u) \cdot du} \end{aligned}$$

and this applies for *any* hazard rate function $I(t)$. Also, because $R(t)$ is the probability of a unit surviving to time t (i.e., not failing in $[0,t]$) then over a population of items or a succession of trials where one item is repeatedly repaired and allowed to run again to failure, it is meaningful to think about the expected time between failures or mean time to failure (MTTF).^[6] This will be the expected value of the failure density function:

^[6] Note that MTTF is not exactly the same as the more often used MTBF of a repairable system. Although usually very close numerically, MTBF is strictly $MTTF + MTTR$ because the time between failures includes the time of repair following the last failure, whereas MTTF is the time to the next failure, following completion of the repair.

Equation 3.13

$$MTTF = E(f(t)) = \int_0^{\infty} t \cdot \left(-\frac{d}{dt}R(t)\right) \cdot dt.$$

Much practical analysis of network or equipment reliability assumes a constant failure rate for equipment items in service. This is not necessarily accurate but it is an accepted practice to characterize failures in service paths arising from a large number of possible independent failures over a large pool of operating equipment in service and independent external events each with individually low probabilities per unit time. Early life stress testing of critical components such as lasers helps eliminate the "infant mortality" portion of the non-constant hazard rate, improving the validity of the assumption somewhat. In addition, if external hazard mechanisms such as cable dig-ups are assumed to be unsynchronized with the equipment deployment, the overall hazard rate from cable cuts can reasonably be modeled as constant on average. A practical justification is also that while mathematical methods do exist to take the non-constant hazard rate curves into effect for each piece of equipment, doing so in real network calculations would imply tracking of the exact type, installation date, and every maintenance date in the life of each individual piece of equipment in each specific network path. Finally, there is recognition that what a planner is often doing with reliability or availability methods in the first place is making *comparative* assessments of alternate networking strategies or broad technology assessment studies of adopting new equipment or operating policies. In these contexts it is seldom the absolute numbers that matter, but the relative ranking of alternatives and these are unaffected by idealization of a constant failure rate. Thus, we have a special practical interest in the case where $\lambda(t) = \lambda_0$ (a constant), for which we get the special results from above that:

Equation 3.14

$$R(t) = e^{-\lambda_0 \cdot t}$$

Equation 3.15

$$MTTF = 1/\lambda_0$$

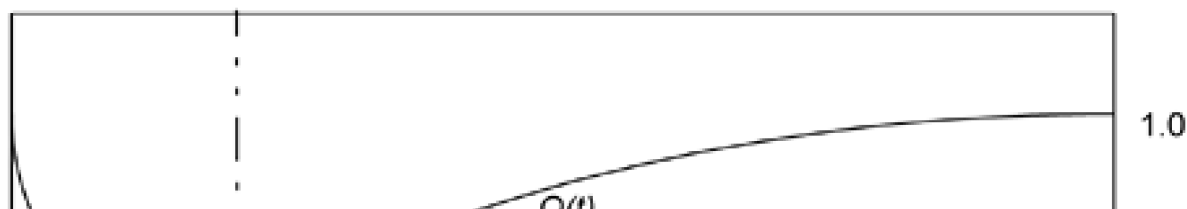
Equation 3.16

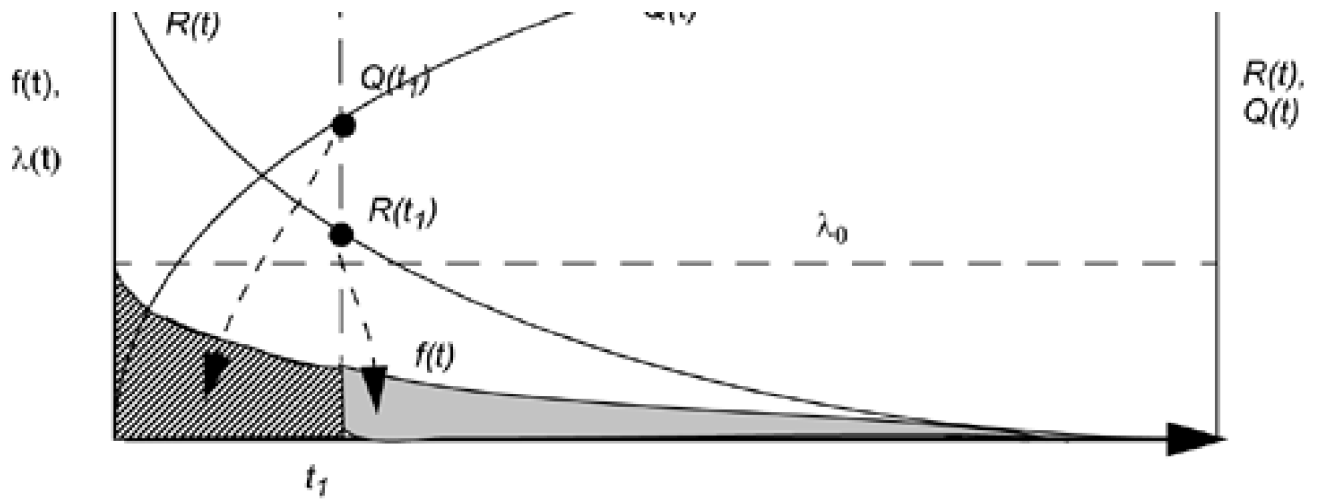
$$\text{and } P(\text{exactly } k \text{ failures in } [0,t]) = \frac{(\lambda_0 \cdot t)^k}{k!} \cdot e^{-\lambda_0 \cdot t}.$$

The last result is otherwise recognized as the Poisson distribution.

The relationships between reliability and failure density functions, and its complement, the cumulative failure distribution function are illustrated in [Figure 3-16](#). The dashed arrows linking function values on $Q(t)$ and $R(t)$ to areas under $f(t)$ show the integral relationships involved. In effect the fundamental function is the failure density $f(t)$. The cumulative failure distribution $Q(t)$ is its integral and the reliability is just the probabilistic complement of $Q(t)$.

Figure 3-16. Reliability $R(t)$, cumulative failure $Q(t)$, and failure density curve $f(t)$ relationships for a constant hazard rate, λ_0 .





[[Team LiB](#)]

◀ PREVIOUS NEXT ▶

3.12 Availability

In this section we review the basic concept of system availability. In the design of the book, the overall material on availability is split between this section, which gives a generic introductory treatment, and [Chapter 8](#), where the topic of availability is developed further in the specific context of determining the availability of paths through mesh-restorable networks.

To appreciate how availability differs from reliability, notice the "mission-oriented" nature of the definition above. Reliability is concerned with how likely it is for the system to operate for a certain time *without* a service-affecting failure occurring. There is no presumption of external repair or maintenance to recover from failures. A pump in the shuttle booster engine must run perfectly for three minutes during launch; there is no chance of repair, so the most relevant measure is reliability: what is the probability of operating flawlessly for three minutes?

This is a different orientation than required to characterize continuously operating systems which are subject to repair when failures occur. An Internet user does not care, for instance, about when the ISP had its first server or router failure, nor would he even directly be concerned with how often such failures occur. If such failures are promptly repaired the user will find the probability of being able to get access at any time suitably high enough to be satisfactory. In other words, *availability* can be high, even if failures are frequent. Availability is concerned with the steady state *probability of finding the system in an operating state at any time we want its service*. We are not concerned with whether its history of operation has been flawless to that point or not.

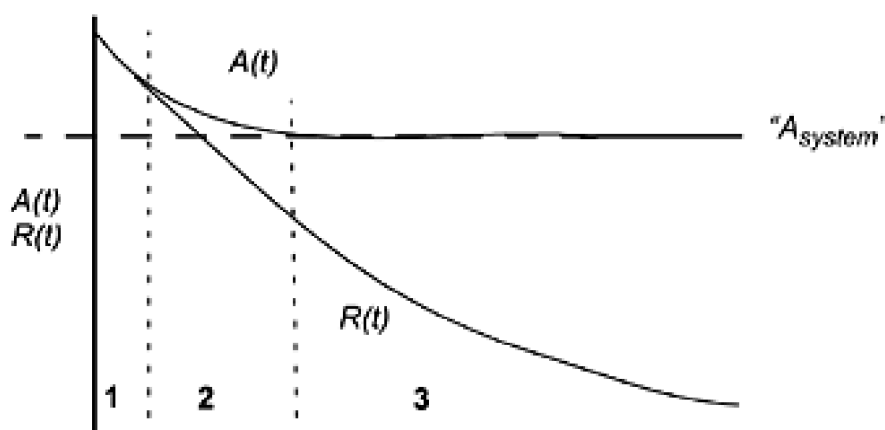
Availability is the probability of the system being found in the operating state at some time t in the future given that the system started in the operating state at time $t=0$. Failures and down states occur but maintenance or repair actions always return the system to an operating state. [\[BiAI92\]](#)

Note that finding a system in the up state at time t_1 is quite different from requiring or expecting that it has stayed continuously in the operating state from $t=0$ to t_1 . When considering availability of repairable systems, a statistical equilibrium is reached between the failure arrivals process and the repair process, both characterized by respective rates, and resulting in a fraction of total time that is "up" time. The fraction of all time that is up time is the system availability, more particularly the steady-state availability. In general a system is biased toward being in the up state for a short time after having a known start in that state. With time, however, failures arise, are repaired, and the system reaches its steady-state equilibrium. [Figure 3-17](#) illustrates these relationships for the case of a single repairable component with a constant failure rate and repair rate. It is not the case that reliability is undefined for a repairable system. It continues to be the probability that the system operates failure-free for the interval $[0, t]$. Whether the system is repaired or not at the point of first failure only makes a difference to the availability. Without repair, however, reliability and availability are identical and both trend monotonically to zero with time. By far in practice it is the steady-state availability that is of interest. Nonetheless, touching on the concept of time-dependent and steady-state availability helps clarify the nature of the phenomenon and also its relationship to the reliability of the same system.

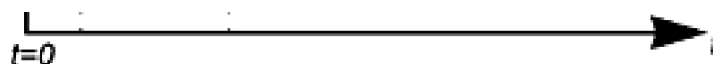
Figure 3-17. Relationship between reliability and steady-state and time-dependent availability for a single (non-redundant) repairable component or system.

Region:

- 1) Reliability and availability are same shortly after known-good starting point.
- 2) Repair actions begin to hold up the availability while $R(t) = \text{prob. (no failures yet)}$ goes on decreasing.
- 3) Equilibrium reached between failure and repair processes; availability reaches steady-state, $R(t)$



goes on dropping to zero.



Fundamentally, the steady-state (as opposed to time-dependent) availability is:

Equation 3.17

$$A = \lim_{T_{obs} \rightarrow \infty} \left\{ \frac{\text{Uptime}}{T_{obs}} \right\}$$

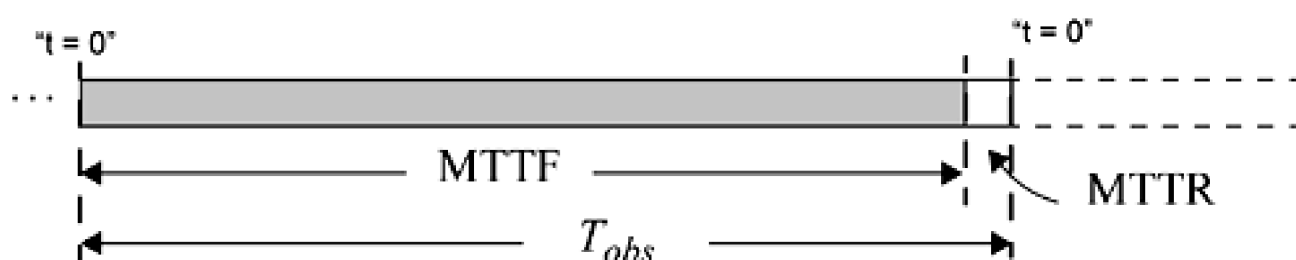
where T_{obs} is a total observation time. Henceforth, we refer to this simply as the system availability. It is the proportion of time that a system is providing its intended service or function observed in the limit as time increases toward infinity.

The most widely known equation for this limit is based on the intuitive picture of the life of a repairable system as a succession of cycles around a loop of operating-failure-repair-operating states. If we assume that each repair episode restores the system to its fully nominal operating condition, the time t in the reliability function is effectively reset to $t=0$. Consequently the expected time to the next failure is the MTTF (Equation 3.15). By definition at $t=MTTF$ following each repair, another failure occurs on average. This is followed by another repair time whose average duration we denote MTTR. Thus the long-term life of a repairable system is comprised of repeated cycles as shown in Figure 3-18. The actual times to failure and actual repair times for each individual cycle are random variables, but for the limiting case we need to know only their averages. In other words, the failure-repair-operating cycle is not regular and repeatably timed as the diagram might seem to suggest. Rather, this is the conceptual limiting average failure cycle. Once this mental image of the "failure cycle" is obtained, it is easy to remember or derive the most widely used expression for availability whenever it is needed. This is:

Equation 3.18

$$A = \frac{\text{uptime}}{T_{obs}} = \frac{MTTF}{MTTF + MTTR}$$

Figure 3-18. Time-line illustration of the failure cycle of a repairable maintained system.



Often the term MTBF ("mean time *between* failures") appears in this expression instead of MTTF. As mentioned in the footnote of page 155, it rarely makes a significant numerical difference, but conceptually MTTF is the correct form. If repair times are a significant fraction of average operating times, this distinction can become important, however.

Note that Equation 3.18 applies on the means of the failure density and repair time functions regardless of their distributions as long as the system is statistically stationary (expectations and other moments are not time-varying). Under these conditions Equation 3.18 is also equivalent to

Equation 3.19

$$A = \frac{\mu}{\lambda + \mu}$$

where $m = 1/MTTR$ is the "repair rate" and $\lambda = 1/MTTF$ is the "failure rate."

3.12.1 Concept of "Unavailability"

The probabilistic complement of availability A is the *unavailability* U ,

Equation 3.20

$$U = 1 - A.$$

In a much availability analysis for communication networks, we work with *unavailability* quantities or expressions because of some simplifying numerical assumptions which we will now examine. These assumptions often make the analysis or modeling of complex systems feasible, with acceptable numerical accuracy, in cases where the exact availability model would be intractably complex. The most enabling simplification is the concept of *adding unavailabilities instead of multiplying availabilities* for elements in series. As long as we are considering sub-systems or components that each do have relatively high absolute availability, then from [Equation 3.18](#) and [Equation 3.20](#) it follows that

Equation 3.21

$$U = \frac{MTTR}{MTTF + MTTR} = \frac{\lambda}{\lambda + \mu}$$

from which, for the many practical cases of interest in which $MTTF \gg MTTR$ (e.g., years versus hours is typical), we can write

Equation 3.22

$$U \approx \frac{MTTR}{MTTF} = \lambda \cdot MTTR.$$

In other words, unavailability is approximated as simply the repair time times the frequency of failure, or the failure rate expressed in the appropriate inverse-time units.

FITS

The FIT is an internationally used unit for measuring or specifying failure rates. Because individual components or subsystems are generally highly reliable in their own right, the convention has arisen of using a period of 10^9 hours as a time unit or time scale on which to quantify failure rates (or conversely MTTFs):

Equation 3.23

a failure rate of 1 failure in 10^9 hours = 1 "FIT"

Thus,

Equation 3.24

$$MTTF = 10^9 / FITs$$

gives the MTTF in hours, if the FIT rate is known, and

Equation 3.25

$$U = (MTTR * FITs) / 10^9$$

is the unavailability if MTTR is given and the failure rate is given in FITs. The following examples give a feel for some typical failure rates, MTTRs, and common "constants" involved in typical communication network unavailability analyses^[7].

^[7] References [\[ToNe94\]](#), [\[ArPa00\]](#), [\[MaCo03\]](#), [\[Free02\]](#), [\[Free96b\]](#) provide additional failure rate and availability data. Selected numbers given here are from [\[ToNe94\]](#).

- 1 year = 8766 hours
- 1 failure/year = 114,155 FITs
- 1 FIT = 1 failure in 114,155 years
- Typical FITs for a logic circuit pack of medium complexity = 1500, i.e., MTTF = 76 years
- FITs for an optical Tx circuit pack = 10,867, => MTTF = 10.5 years
- FITs for an optical receiver circuit pack = 4311
- "Three nines availability" $U = 10^{-4}$ -> $A = 0.99900$ => 8.76 hours per year outage
- "Five nines availability" $U = 10^{-6}$ -> $A = 0.99999$ => 5.26 minutes per year outage
- Typical cable cutting (failure) rate = 4.39/year/1000 sheath miles => 5000 FITs/ mi
- Typical cable physical repair MTTR = 14.4 hours
- Typical plug-replacement equipment MTTR = 2 hours

Note that while failure rates have "units" of FITs, A and U are inherently dimensionless as both are just time fractions or probabilities of finding the system up or down, respectively.

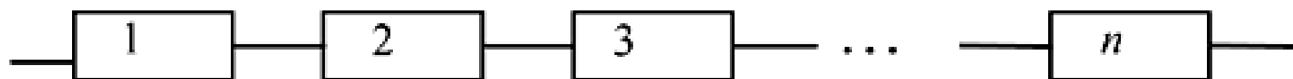
3.12.2 Series Unavailability Relationships

If a system is comprised of n components (or subsystems) in a "series" availability relationship then (like the proverbial Christmas tree lights) *all* components must be operating for the system to be available. Figure 0-1 shows the availability block diagram for elements in a pure series relationship. For elements in series the overall reliability function becomes:

Equation 3.26

$$R_s(t) = \prod_{i=1}^n R_i(t)$$

Figure 0-1. Elements in a series availability relationship.



and the exact form for unavailability and availability are

Equation 3.27

$$A_s = \prod_{i=1}^n A_i ; \quad U_s = 1 - \prod_{i=1}^n A_i = 1 - \prod_{i=1}^n (1 - U_i)$$

As the term "series" implies, the reliability reflects the product of probabilities that any one might fail by time t , and the availability of the whole requires that every series element must also be available.

Adding Series Unavailabilities

Let us now show why, as mentioned above, one can numerically approximate Equation 3.27 by a corresponding sum of unavailability values. Specifically the assertion is that:

Equation 3.28

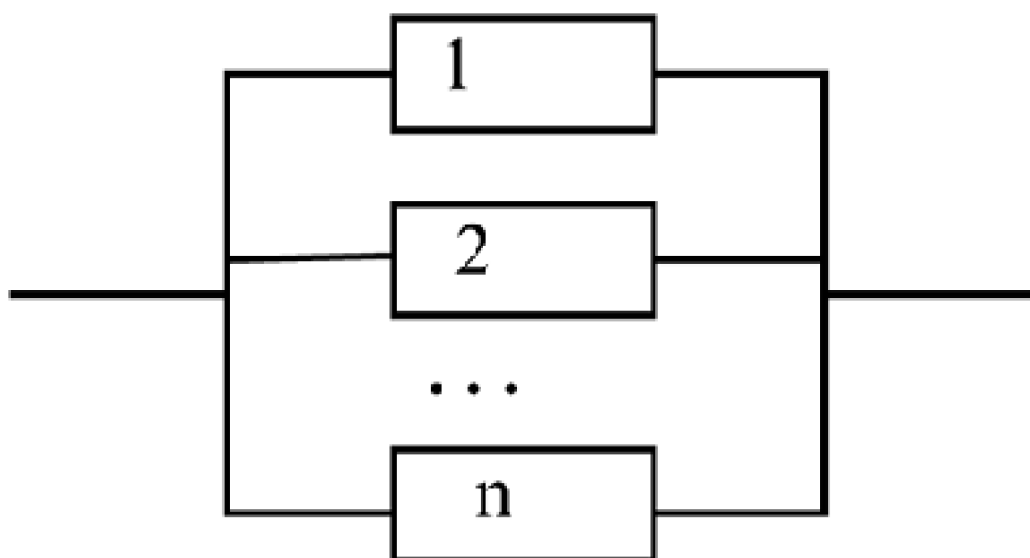
$$\prod_{i=1}^n A_i \approx 1 - \sum_{i=1}^n U_i$$

where A_i is the availability of the i^{th} set of N elements in a series relationship, and U_i is the corresponding unavailability, $1-A_i$. A simple example with two elements in series will show the basis of this useful numerical approximation. Consider two identical elements A and B in a series availability relationship, each with elemental unavailability U . If we assume A and B fail independently, the exact result would be $A = P(AB) = P(A)P(B)$. Or, therefore $A = (1-U)(1-U) = 1 - 2U + U^2$. In contrast, [Equation 3.28](#) would give us $A = 1 - 2U$. Thus we see that in "adding unavailabilities" of two elements in series we are only ignoring the square of an already small number U^2 . Moreover, the error in the approximation is toward being conservative (in an engineering sense) because to the extent we err numerically, it is in the direction of underestimating the actual availability. The typically high accuracy of this approximation is illustrated in [\[Free96b\]](#) with an example having six elements in series with U_i from 10^{-5} to 10^{-3} . The accuracy of the approximation is better than 0.5% which Freeman notes is also "typically far more precise than the estimates of element A_i 's".

3.12.3 Parallel Unavailability Relationships

To say that n elements or subsystems are arranged in a parallel availability relationship means that only one of them has to be working for the system to be available. As long as one element is working, the system as a whole is providing its intended service or function. [Figure 3-19](#) shows the availability block diagram for elements in a pure parallel relationship.

Figure 3-19. Elements in a parallel availability relationship.



For elements in parallel the overall reliability function becomes

Equation 3.29

$$R_p(t) = 1 - \prod_{i=1}^n (1 - R_i(t)),$$

and the exact form for the unavailability is

Equation 3.30

$$U_x = \prod_{i=1}^n U_i$$

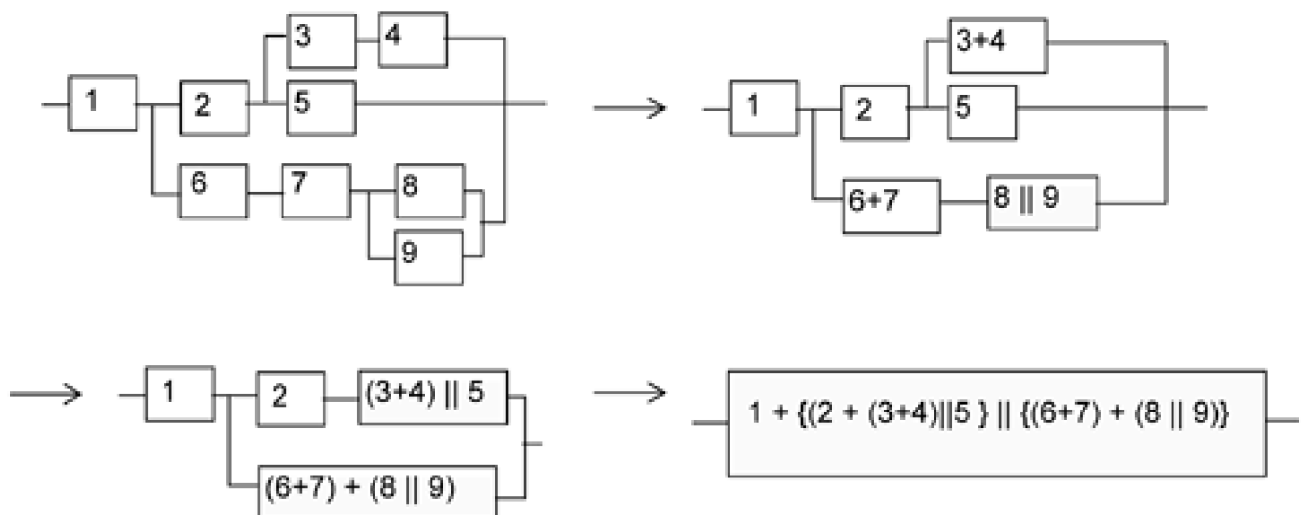
In summary, two useful basics for simplified availability analysis using the unavailability orientation are:

1. *Unavailabilities add for elements in series.* This is an approximation (but numerically conservative) and quite accurate for $U_j \ll 1$.
2. *Unavailabilities multiply for elements in parallel.* This is exact.

3.12.4 Series-Parallel Reductions

The first step in more detailed evaluation of system availability is often to apply repeated series-parallel reductions to the availability block diagram of the system. This involves repeated application of the two main rules: unavailabilities add in series, unavailabilities multiply in parallel. For relatively simple problems, a suitable series of series-parallel reduction steps can completely solve the problem of computing system availability. Figure 3-20 shows an example of this type. As a convenient shorthand in Figure 3-20 we denote element unavailabilities simply by the element numbers and "A + B" means the addition of the unavailabilities of blocks A and B. Similarly the notation "A || B" means the unavailability of element A in parallel with element B, i.e., the product of their unavailabilities. In the example, three stages of reduction lead directly to a simple algebraic expression for the system unavailability as a function of the elemental unavailabilities. The problem is simple because there are no cross-coupling paths in the availability model. The approach to calculation of availability for more complex system availability block diagrams is facilitated by first introducing the topic of network reliability.

Figure 3-20. Example of series - parallel reductions.



[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

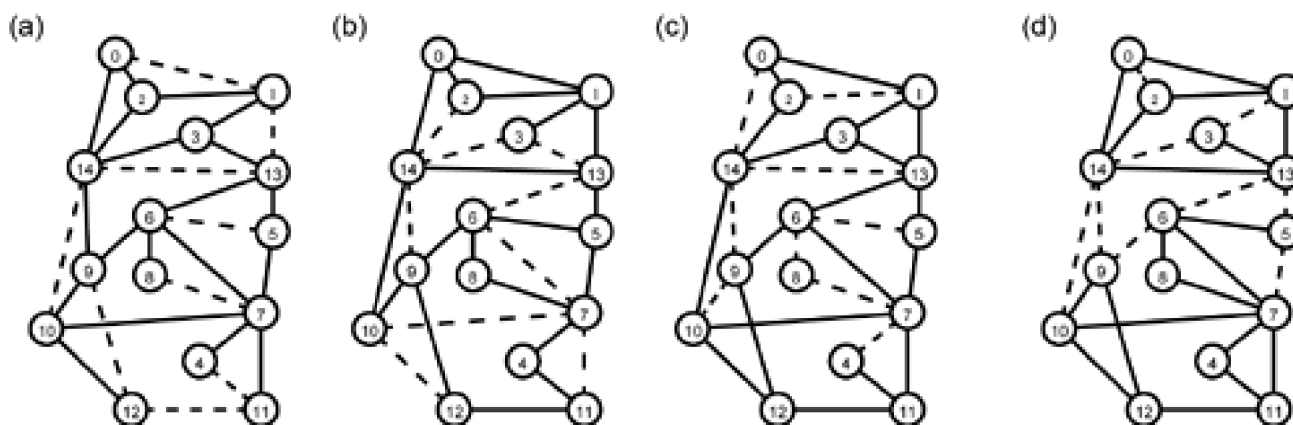
3.13 Network Reliability

The field of "network reliability" is concerned with questions of retaining graph connectivity in networks where edges have a non-zero probability of being in a failed state. The central issue is simple-sounding but in fact it is quite difficult to exactly compute the probability that a graph remains connected as a whole, or if a path remains between specific nodes or sets of nodes, in a graph with unreliable edges. Specific measures that are studied in this field are questions of " $\{s,t\}$ " or "two-terminal" reliability, k -terminal reliability, and all-terminal reliability. These are all various measures of the purely topology-dependent probability of graph disconnection between pairs of nodes points. Rai and Agrawal [RaAg90] provide a complete survey of this field. Here we try only to extract those basic ideas of network reliability that form part of a grounding for work on transport network survivability and feed into the problem of availability block diagram reduction.

Figure 3-21 illustrates the basic orientation for the network reliability problem. Four equally likely states are drawn for an assumed $p_{link}=0.32$ (i.e., out of 28 links present we expect 9 of them down at any one time on average). A solid line is a working link, dashed is a failed link. If we pick nodes 0-11 we see that in (a)-(c) despite the failures there is always still a route between them. Inspection shows in fact that none of the randomly generated states (a)-(c) contributes any two-terminal unreliability: there is still at least one topologically feasible route between all node pairs. Equivalently, we can say that none of these failure combinations has disconnected the graph. Case (d), however, is an equally likely state but has a dramatically different effect. Four of the nine failure links form a cut of the graph across edges (14-19), (14-9), (6-13) and (13-5). The two-terminal reliability of all node pairs separated by the cut are thus affected by this failure state. This not only illustrates how abrupt and discontinuous network behavior is in general but it also conveys why numerical enumeration of all link state combinations, followed by tests for graph connectivity, is not feasible for this type of problem on a large network.^[8]

^[8] Note also that with $p_{link}=0.32$, 9 (out of 28) is the only most likely number of failures at any time, but that all states with fewer or greater number of failed links at the same time still occur with lesser but finite probability, and may also contribute to graph disconnection. The complete calculation of network reliability has to consider all such combinations, not just states with exactly the expected number of failures, as used for the example.

Figure 3-21. Network reliability: How likely is it that at least one route exists between nodes? In this example there are 2^{28} link-state combinations to consider.



Of course in a real network, there may also be outage due to finite capacity effects in Figure 3-21 (a) though (c) but this is not in the scope of the basic "network reliability" problem. Basic network reliability (in the sense of [Colb87],[HaKr95],[Shei91]) presumes that there are no routing or capacity constraints on the network graph. If at least one route exists topologically between $\{s,t\}$, then it is assumed the signal (or packet train) will discover and use it. With this limitation understood, however, its methods and concepts can provide tools for use in other means for more encompassing considerations availability analysis. The problem of most relevance to the availability of a service path through a network is that of two-terminal reliability.

3.13.1 Two-Terminal Network Reliability

Two-terminal reliability^[9] is the complement to the likelihood that every distinct path between $\{s, t\}$ contains at least one failed (or blocked) link. Exact computation of the two-terminal reliability problem is NP-complete for general graphs even when the link failure probabilities are known. The computational complexity of trying to enumerate all networks states and inspect them for connectivity between nodes $\{s, t\}$ has led to the approach of more computationally efficient bounds. A widely known general form is called the *reliability polynomial*:

^[9] In addition to the treatment given here, interested readers are referred to the excellent tutorial paper on algorithms for terminal pair availability calculation by A. Iselt [[Ise100](#)].

Equation 3.31

$$R(G, \{s, t\}, p) = \sum_{i=0}^m N_i(G, s, t) p^i (1-p)^{m-i}$$

where $G = (V, E)$ is the network graph, $m = |E|$ is the number of edges in the graph, $\{s, t\}$ is a specific terminal pair and p is the link *operating* probability.

This form prescribes either exact or approximate (bounding) estimates of $R(-)$, depending on how $N_i(-)$ is obtained. In its exact form $N_i(-)$ is the number of subgraphs of G in which there are exactly $(m-i)$ failed links but the remaining graph contains a route between nodes $\{s, t\}$. Of course this just defers the problem of calculation $R(-)$ to that of counting or estimating $N_i(-)$. Two simple bounds are conceptually evident at this stage. One is to enumerate (for each $i \in 1 \dots m$) only those $m-i$ failure link combinations that constitute cuts of the graph between $\{s, t\}$. A cut-finding program can thus enumerate a large number of cuts and their associated weights (in terms of number of edges) for insertion into [Equation 3.31](#). Obviously for $p \approx 1$ the smallest cuts are the most likely and hence numerically dominant contributors to $R(-)$.

Assuming not all of the highest order cuts are enumerated^[10] the result will be an upper (i.e., optimistic) bound on the exact $R(-)$. i.e.,

^[10] Note that enumerating all cutsets is no less demanding computationally than the original prospect of generating all link-state combinations, i.e., $O(2^{|E|})$ but each cutset is a direct prescription for connectivity loss so the goodness of the bound and computation time can often benefit from using direct knowledge of most likely failure combinations and can be tailored by systematically increasing the size of the cutsets considered.

Equation 3.32

$$R(G, \{s, t\}, p) \leq 1 - \sum_{i=c}^m C_i(G, s, t) p^{m-i} (1-p)^i$$

where c is the minimum cut of the graph between $\{s, t\}$ and $C_i(-)$ is the number of $\{s, t\}$ cutsets found comprising exactly i edges. The exact reliability will be lower than this because network states involving i failures but containing a cutset of fewer than i edges are connectivity-failure states that are not counted.

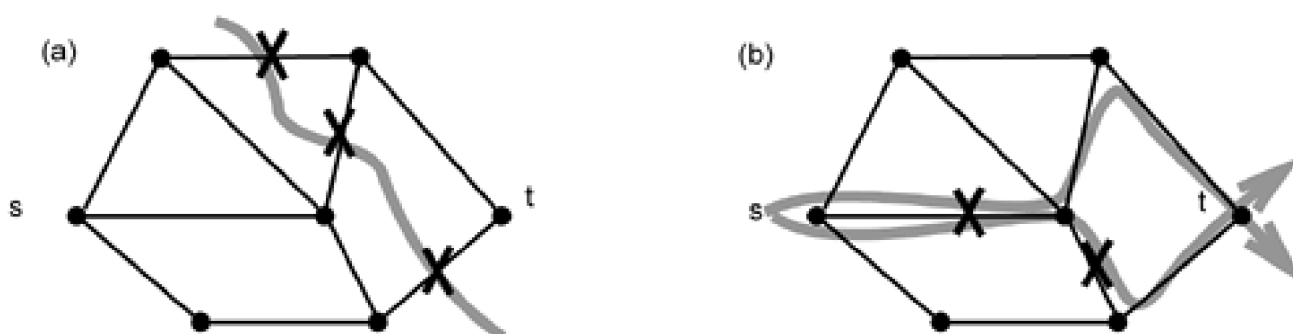
A converse viewpoint for assessing $N_i(-)$ is from the standpoint of network states that contain at least one working route among the set of all distinct routes between $\{s, t\}$. (The two are conceptually the same as the notion of "cuts and ties" in more advanced analysis of system availability block diagrams.) Here, all of the k -successively longer distinct (non-looping) routes on the graph between $\{s, t\}$ are generated and each recorded with its associated length (number of edges in series en route). Then a simple upper (i.e., optimistic) bound on $\{s, t\}$ reliability is:

Equation 3.33

$$R(G, \{s, t\}, p) \leq 1 - \prod_{i=1}^k (1 - p^{L_i})$$

where L_i is the length of the k^{th} distinct route between $\{s, t\}$. [Figure 3-22\(a\)](#) portrays the basic notion of $\{s, t\}$ reliability being viewed in [Equation 3.33](#) as the probability that *not every* possible route is blocked and implicitly treats routes as independent entities. In contrast, [Figure 3-22\(b\)](#) shows how several distinct routes may actually share single link failures in common, illustrating why [Equation 3.33](#) is an optimistic bound.

Figure 3-22. Orientations to the network reliability calculation: (a) failures that together create an $\{s, t\}$ cut set, (b) failures that defeat all routes between $\{s, t\}$.



More precisely the route-based formulation is dependent on union of the probabilities that all edges in route i are operating, i.e.,

Equation 3.34

$$R(G, \{s, t\}, p) = \bigcup_{i=1}^k (1 - p^{L_i})$$

which calls for application of the inclusion-exclusion principle for the union of non-disjoint sets [\[GaTa92\]](#) (p.90). Denoting $P(i) = (1 - p^{L_i})$ as the probability that all links in the i^{th} route are operating,

Equation 3.35

$$R(\) = \sum_{i \in 1 \dots k} P(i) - \sum_{i < j, i, j \in 1 \dots k} P(i)P(j) + \sum_{i < j < n, i, j, n \in 1 \dots k} P(i)P(j)P(n) - \dots + (-1)^{k+1} \prod_{i \in 1 \dots k} P(i)$$

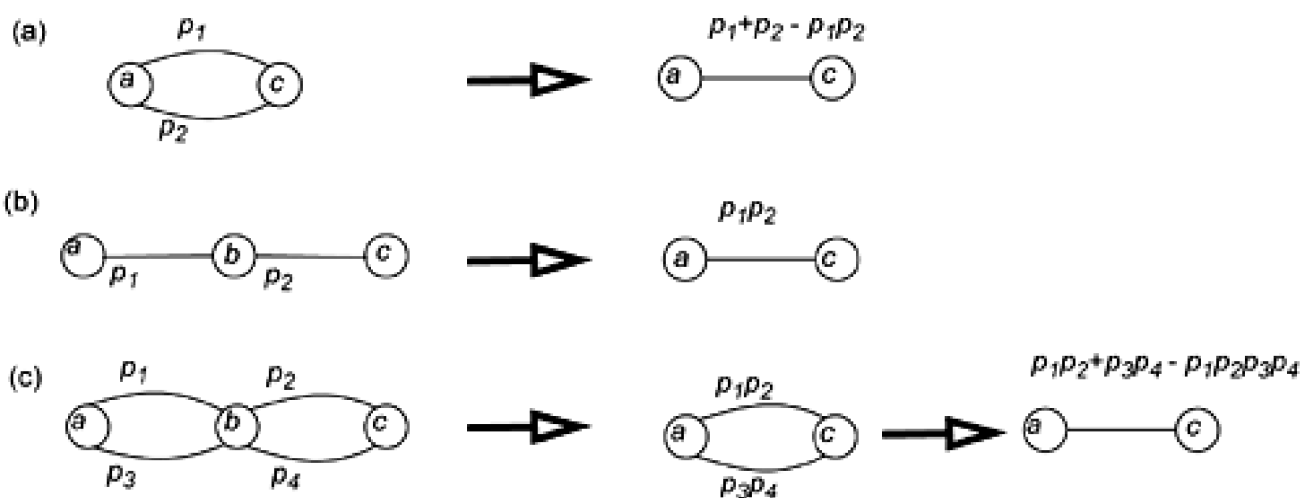
In [\[Shei91\]](#) the application of the inclusion-exclusion principle for probability union is treated further, showing that there are always certain cancellation effects between terms of the inclusion-exclusion series that give further insights (the concept of irrelevant edges) and that can be exploited to simplify the expansion process.

3.13.2 Factoring the Graph and Conditional Decomposition

Let us now return to the problem of calculating system availability in cases where basic series and parallel relationships do not completely reduce the model. This is where the link to network reliability arises. If a network is completely reducible between nodes $\{s,t\}$ by repeated application of simple reductions into a single equivalent link, the network is said to be *two-terminal series-parallel*. In such a case the resultant single reduced edge probability is $R(G, \{s,t\}, p)$. But many realistic cases are not two-terminal series-parallel in nature because of some edge that cross-couples the remaining relationships in a way that halts further application of the series-parallel reductions. In the approach that follows, which is also based in network reliability, such an edge is used as a kind of pivot point on which the problem is split into two conditional probability sub-versions that apply when the particular edge is in one case assumed available and in the other case where it is assumed to be down.

[Figure 3-23](#) summarizes the basic series-parallel reduction rules in a canonical form on the edge probabilities (probabilities of the link being up, equivalent to the elemental availability). Cases (a) and (b) are the previous basic parallel and series relationships, to which case (c), called a "two-neighbor reduction," is added. When applied to either a network graph or an availability block diagram, these transformations are exact or, in the language of network reliability, they are "reliability preserving." To use these reductions, element failures must be statistically independent, and in cases (b) and (c) in [Figure 3-23](#) node b must have no other arcs incident upon it. Node b also cannot be either the source or target. While single arcs are shown the rules apply to any block that is similarly reducible to a single probability expression, so that, for instance, p_1 in [Figure 3-23\(a\)](#) may already be the result of a prior set of series-parallel reductions.

Figure 3-23. Reliability-preserving graph reduction rules: (a) parallel, (b) series, (c) two-neighbor reduction.



In general the application of series-parallel reduction rules will be exhausted before the original network is completely reduced. This will usually manifest itself through some edge that cross-couples between remaining subgraphs, i.e., one or more nodes will be like node b in [Figure 3-23\(b\)](#) but with the presence of more than just two arcs, so that another application of a series reduction is not possible. At this stage the graph can be "factored" to continue the reductions. Graph factoring is based on Moscovitz's *pivotal decomposition formula* [\[Shei91\]](#) (p.10). The key idea is that:

Equation 3.36

$$R(G, s, t, p) = p \cdot R(G|e) + (1 - p) \cdot R(G - e)$$

where p is the probability the edge is available, $G|e$ means graph 'G given e', and $\{G-e\}$ is graph G without edge e. Thus the whole is considered as the conditional probability decomposition of the two states that the confounding edge e may be in, with probability p and $(1-p)$

respectively. $G|e$ is represented by graph G where edge e is contracted or "short circuited." The probability-weighted sum of the two conditional probability decomposition terms is the two-terminal graph reliability. In practice the idea is to recognize a key edge e that will decouple the two resulting conditional subgraphs in a way that allows another round of series-parallel reductions. A complete graph may thus be decomposed through a series of series-parallel reductions, splitting to two conditional subgraphs, series parallel reductions on each, splitting again in those as needed, and so on. The real computational advantage of the decomposition steps is to overcome the situations where no further series-parallel reductions are possible. Were it not for this use of decomposition to link between subproblems that are further series-parallel reducible, it would be of little practical value because by itself it is equivalent to state-space enumeration by building a binary tree of all two-state edge combinations. More detailed treatments can be found in [Colb87] (p.77) and [HaKr95].

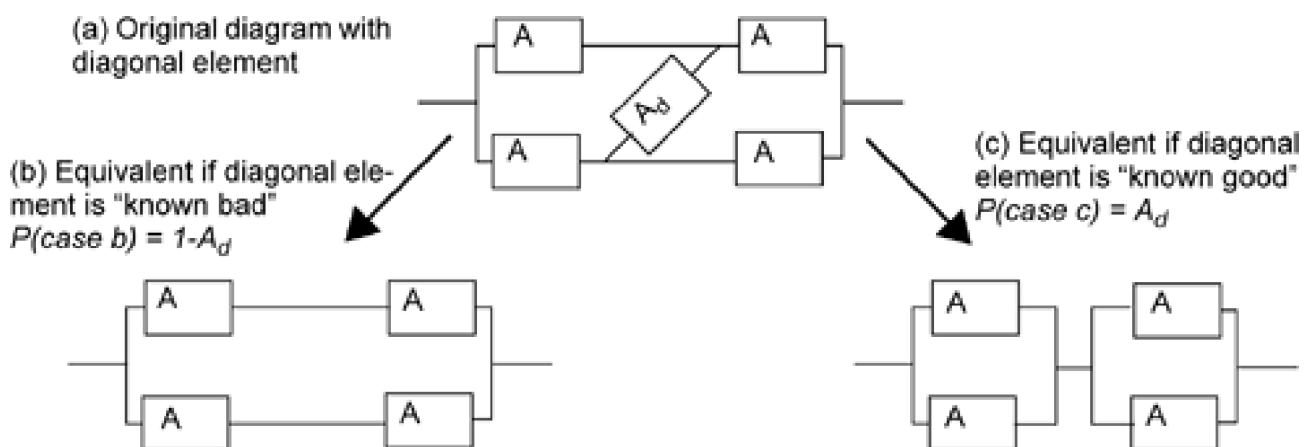
To illustrate the application to availability problems, however, consider the availability block diagram in Figure 3-24(a). Because of the "diagonal" element, it is not amenable to series-parallel reduction. We do, however, obtain two subgraphs that are each easily analyzed if we presuppose the state of the diagonal element. In (b) we presume it is failed. In (c) we presume it to be in a working state. Thus the resulting subgraphs are conditional probability estimates of the system availability. To get the overall availability we weight the result for each subgraph by the probability of the decomposed link state that lead to that subgraph. Therefore, for this example:

Equation 3.37

$$A_{sys} = (1 - A_d) \cdot A_{\text{case(b) subgraph}} + A_d \cdot A_{\text{case(c) subgraph}}$$

$$A_{sys} = (1 - A_d) \cdot [1 - \{(1 - A^2)\}^2] + A_d \cdot [1 - (1 - A)^2]^2$$

Figure 3-24. Example illustrating conditional decomposition of an availability block diagram.



3.14 Expected Loss of Traffic and of Connectivity

Let us now return to look at the two most commonly used ROF-type measures of survivability. These are the *expected annual loss of traffic (ELT)* [ArPa00] and the *annual expected downtime of connection (AEDC)* [1A193]. An appreciation of these measures is aided by the background above on availability and network reliability. ELT is like a traffic-weighted path availability and AEDC is a certain type of two-terminal network reliability measure. Both of these are path-oriented measures in that they pertain to a transport signal path between a stipulated pair of nodes. The path orientation follows the philosophy of using hypothetical reference paths (HRP) for various performance assessments. This type of measure allows comparisons between network designs and/or various restoration techniques to be made on the basis of one or a few path models of specific relevance and that are agreed by stakeholders to be representative of either worst or average cases. In principle the corresponding path measures can be computed for all possible end-node pairs to obtain statistics of all individual paths in the network. A path in this context is constituted by a demand unit between nodes i, j that is a contiguous transmission signal.

3.14.1 Expected Loss of Traffic (ELT)

For a given pair of nodes exchanging demand over possibly several paths through a network, ELT asks what the expected number of lost demand-minutes will be over a year. The total demand between a node pair need not follow a single route. The total demand between i

$$\sum_{p \in P} d^p_{i,j} = d_{i,j}$$

and j , denoted d_{ij} may be realized by routing over a set of diverse routes P as long as $d^p_{i,j} = d_{i,j}$. One way of implementing the ELT calculation is then:

Equation 3.38

$$ELT_{i,j} = \sum_{p=1 \dots P} \left[d^p_{i,j} \cdot \sum_{\forall k \in (S, M)} \delta^p_{i,j}(k) \cdot U_k \right] \cdot M_0$$

where $d^p_{i,j}$ is the amount of (i,j) demand assigned to the p^{th} route and $\delta^p_{i,j}(k) = 1$ if span k is in the p^{th} route for demand pair (i,j) , and zero otherwise. The constant $M_0 = 5.26 \times 10^5$ gives ELT units of demand-minutes/year of "traffic" loss. Thus ELT is the sum of the demand-weighted unavailability of each distinct (not disjoint) path employed to bear the total demand between nodes i, j . In the inner sum every node and span in the network is indexed by k and the routing function $\delta^p_{i,j}(k)$ answers the 1 / 0 question: is network element k a constituent of the p^{th} diverse route used for demands between i and j ? If so, the unavailability of that element, U_k , contributes to the ELT. The benefit of ELT beyond a single path availability analysis is that it reflects the size and number of demand units affected, allowing apples-versus-apples comparison across alternatives involving different signal levels, diversity routing and/or restoration techniques. Note that Equation 3.38 has some of the same numerical approximations and assumptions as mentioned above for availability analysis. In particular it strictly overestimates the total by virtue of the addition of unavailabilities for each distinct path as if they were independent.

Also, as written, the ELT formula is most applicable to "passive" point-to-point transmission networks where the network elements in each path directly contribute their true U_k values to the total (although the U_k for a transmission span may include the built-in benefit of co-routed APS against internal failures). In a network that embodies active restoration mechanisms and designed-in spare capacity, however, the U_k values should be those already reflecting any net benefit in equivalent unavailability terms of the designed-in survivability methods. For instance in a span-restorable mesh network, the U_k values for spans could be the equivalent unavailability of spans defined

in [Chapter 8](#). Alternately the native path availabilities could be presented but a simple extension to [Equation 3.38](#) used so that unavailability is not contributed to the sum if *either* a path or its known backup are available. This implicitly recognizes the switch-over from working to protection that happens to avoid "lost traffic" in a survivable network with active restoration or protection. Ultimately, however, if the details of modeling the effects of active restoration measures into the framework of [Equation 3.38](#) become unmanageable, [Equation 3.38](#) still guides how one would approach the calculation of ELT by simulation.

3.14.2 Annual Expected Downtime of Connection (AEDC)

AEDC is a Random Occurrence of Failure (ROF) measures of survivability defined in [\[T1A193\]](#). AEDC is more like a network reliability measure. It asks how often during a year would one expect *total disconnection* of all the paths for communication between nodes *i* and *j*. [\[T1A193\]](#) states:

"Connection between two nodes is lost if, for all paths, there exists no working or protection channels able to carry the demand. The consequences of losing connection between two nodes can be more serious than the consequences of losing an equivalent amount of traffic throughout the network without losing connectivity. Loss of connectivity can lead to the loss of important emergency and high priority traffic or create a situation of isolation."

Thus AEDC specifically relates to the two-terminal reliability of a network. The main difference is that classical network reliability addresses whether there exists *any* topologically possible route between nodes, but the intent with AEDC is to consider details or limitations of the actual restoration mechanism and network capacity constraints that would be involved in determining how many, individual prefailure paths between *i* and *j* could feasibly be restored. The point is that while the graph may remain connected it is possible that specific rerouting mechanisms may be starved of capacity or constrained in routing so that they cannot emulate the routing generality in classical network reliability. It is safe to say, however, that the two-terminal reliability of the network graph (multiplied by the number of minutes in a year) would be a lower bound on the AEDC (in minutes).

A more exact evaluation of the AEDC can also be conducted with the availability analysis methods above. The approach is as follows for a given node pair:

1. Identify all distinct routes between the nodes which would be eligible for use for either the normal working path or a protection or restoration path.
2. Represent the set of distinct routes as a series-parallel availability block diagram.
3. Apply series-parallel availability reductions to the current availability block diagram.
4. When step 3 halts, but the reduction is not complete, select a cross-over element in the block diagram and apply a conditional decomposition.
5. Repeat steps 3-4 until the block diagram is completely reduced.
6. The AEDC is the resulting unavailability value times the number of minutes in a year.

The basic process is illustrated in [Figure 3-25](#) for a small network where the AEDC for nodes A to D is calculated. In the availability block diagram the dashed block names (such as node B') represent the unavailability of the network node and the physical span leading up to it. i.e., B' represents node B and link A-B as a single block. A way to develop the availability block diagram (c) from the network diagram (a) is to start from the list all distinct routes. The first route becomes the pure series path model seen at the top (A B C D) in [Figure 3-25\(c\)](#). If the next route listed was fully disjoint from the first then it too is drawn in full, in perfect parallelism to the first route represented. More generally where the routes are not disjoint, one tries to draw the path in parallel but has to obey a rule that if an element has been drawn already it cannot be drawn again. Thus, for the second route listed we drop down from A to represent E and F, then return to up to D. The third route above also adds no new elements, just the vertical link between rows joining the output of E to the input of C. Note this link is in effect unidirectional in that it represents the route option going from E to C but does not imply a corresponding direct linkage between B and F. The last route lays down the link from the joining the output of C back to the input of E to represent the route (ABC)-(EFD).

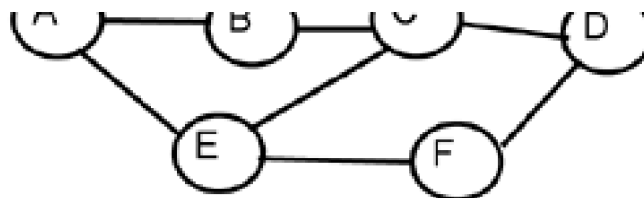
Figure 3-25. Example using availability analysis to estimate AEDC.

(a) Sample network

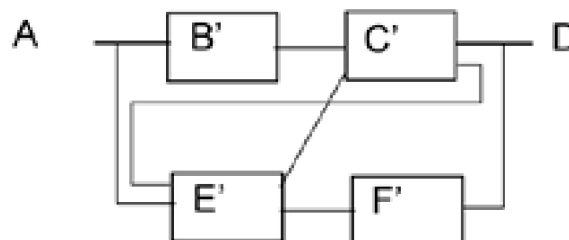


(b) Distinct routes between A-D

- (i) A B C D
- (ii) A E F D
- (iii) A E C D
- (iv) A B C E F D



(c) Corresponding availability block diagram



As the example shows, the primary complication is that the "diverse" paths between A and D are not *disjoint* paths. There is thus a correlation of failures affecting each path. This is true for ELT as well, but ELT is an expected sum of demand weighted outage on all paths, a probability union type of construct which is numerically insensitive to this at typical Uk values. In contrast AEDC is a question of probability intersection. That is, all paths down defines the condition for i,j "disconnection." In this case any common elements among one or more paths can drastically alter the probability of all paths failing together. Consider for instance a case of $|P|=5$ but all paths sharing on node in common (span disjoint but not node disjoint). Then the AEDC would be almost literally just the Uk of the one node common to all paths. The general case of the paths not being fully disjoint does not lend itself to direct analytical expression. Rather, the methods of series-parallel reduction and decomposition can be used to address the question and any specific limitations to the rerouting capability of the restoration method are reflected in the set of distinct routes represented.

If a set of fully disjoint paths between (i,j) is identified (or span disjoint paths where node failures are not being considered) then the AEDC can be lower-bounded (i.e., an optimistic bound) as:

Equation 3.39

$$AEDC_{i,j} = \prod_{p \in P} \left\{ \sum_{\forall k \in S} \delta^p_{i,j}(k) \cdot U_k \right\} \cdot M_0.$$

This is "exact" (i.e., except for the series addition of elemental unavailabilities on each path) for the fraction of time during the year that the set of all mutually disjoint paths between i and j would all be down simultaneously.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

Chapter 4. Graph Theory, Routing, and Optimization

Having familiarized ourselves with transport networking in [Chapter 1](#), with recent IP and optical networking developments in [Chapter 2](#), and having had a preliminary overview of survivability principles and methods in [Chapter 3](#), this chapter completes the "preparatory" phase of the book by providing a toolkit of basic concepts, algorithms and optimization techniques that can be brought to bear in planning, design, and research related to transport networking. We use this chapter to establish a common set of concepts and methods as a foundation for the more advanced treatments of mesh-networking topics in [Chapters 5](#) through [11](#). First we look at a number of simple concepts and definitions from graph theory. We then look at a number of basic concepts and algorithms for routing, cycle finding, and tests of graph properties such as biconnectivity, etc. The last part of the chapter is devoted to optimization methods and concepts as applied to network design problems.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.1 Graph Theory Related to Transport Networking

It is natural to represent transport networks as a set of nodes and connections between these nodes. Typically, the nodes are buildings or equipment enclosures with the connections between them representing transmission facilities or cable ducts. This makes useful terms, concepts and results from graph theory available to us. This section introduces some basic notation and concepts from graph theory. Where appropriate we also explain the somewhat parallel terminology that is more particular to the specific context of a transport network. An important basic concept is that a graph may represent certain aspects of a network, usually its topology, and often an abstraction of its capacity, but a graph and a network are not the same thing. We can speak of the graph of a network but a network itself comprises the actual equipment, signals, cables, fibers, and so on.

4.1.1 Set Concepts and Notation

Sets arise frequently in formulating and solving network planning problems. In particular optimization problems are often stated as a series of relationships between sets and operations on set members. Thinking in terms of operations on sets, and relationships between sets (instead of sequences and sequential processing) is especially important when working with a high-level optimization programming language such as AMPL [\[FoGa93\]](#). The objects that form a set are called its *elements*. The elements of a set are drawn from a population known as the *base type* for that set. A set is completely specified by the elements that belong to it, without regard to the relations between the elements. For example, a set containing five objects arranged in circle is the same as a set containing the same five objects arranged in a line. Thus, there is no concept of order in a set (unless specifically stated). There is also no concept of duplication in a set; all elements in a set are distinct from one another and an any element either belongs to the set or not. Some of the symbols used to express sets and their relationships are shown below.

- $M = \{a, b, c\}$ means the set M is composed of elements a , b and c .
- $X = M - \{c\}$ means the set X is composed of the elements of M except for c .
- $a \in M$ means a is an element of set M .
- $n \notin M$ means n is not an element of set M .
- $\forall m \in M$ means for every element m in set M . \forall is read as "for every...".
- \emptyset is the null or empty set.
- $|M|$ is the *cardinality* or number of elements in set M .
- $\{x | x \text{ is a positive integer}\}$ is a set *former* meaning all x such that x is a positive integer. $|$ is read as "such that...".
- \exists is the "existential qualifier" meaning that some specified object must exist as a condition for some qualification or condition to be met.
- $A \subseteq B$ and $B \supseteq A$ both mean that set A is a subset of B and set B is a super-set of set A .
- $A \cup B$ indicates a *union* of sets: all elements in either A or B .
- $A \cap B$ indicates an *intersection* of sets: all elements in both A and B .
- $A - B$ indicates a *difference* of sets: all elements in A not in set B .

- $A \oplus B$ indicates a *symmetric difference* of sets: all elements in either A or B , excluding those elements in both A and B , i.e., $\{x \mid x \in A \cup B, x \notin A \cap B\}$.
- If A and B are both sets, then $A \times B$ represents a set C with cardinality $|C| = |A| \cdot |B|$ that contains all distinct pair-wise combinations of elements from A with those in B . This may be extended to generating all pairs or triples etc. of a set itself, denoted as A^2 or A^3 .

To illustrate use of the notation consider two sets $M = \{a, b, c\}$ and $N = \{b, e, f, g\}$. Because M has three elements, $|M| = 3$. The union of M and N is $M \cup N = \{a, b, c, e, f, g\}$. The intersection of M and N is the set of elements that appear in both M and N , which is $M \cap N = \{b\}$. The set difference of M and N is $M - N = \{a, c\}$. The symmetric difference of M and N is the set of elements appearing in M and N but not in both, which is $M \oplus N = \{a, c, e, f, g\}$. As an example of more complex set-related notation that we see later consider the expression $\forall (i, j) \in M^2 (i \neq j)$. This would be read as "for every (i, j) pair in set M by M , such that i is not equal to j ," etc.

4.1.2 Graph Theory as it Relates to Transport Networks

A *graph* $G = (V, E)$ consists of a finite set of *vertices* $V = \{v_1, v_2, \dots\}$ and a set of *edges* $E = \{e_1, e_2, \dots\}$, such that each edge in E joins a pair of vertices in V . A tacit convention in transport networking is to revert to the graph theoretic terms of "vertex" and "edge" (or arc) when the point or argument being made is a general one, or the discussion refers to a result based in graph theory itself, and so on. When referring more directly to a transport network context, however, one finds the terms *node* and *span* (or *link*), and corresponding sets N and S used instead of vertices and edges. Often the usage is synonymous, but the difference in usage arises from the context of discussing something purely within the domain of graphs in general, or based on reference to actual networks.

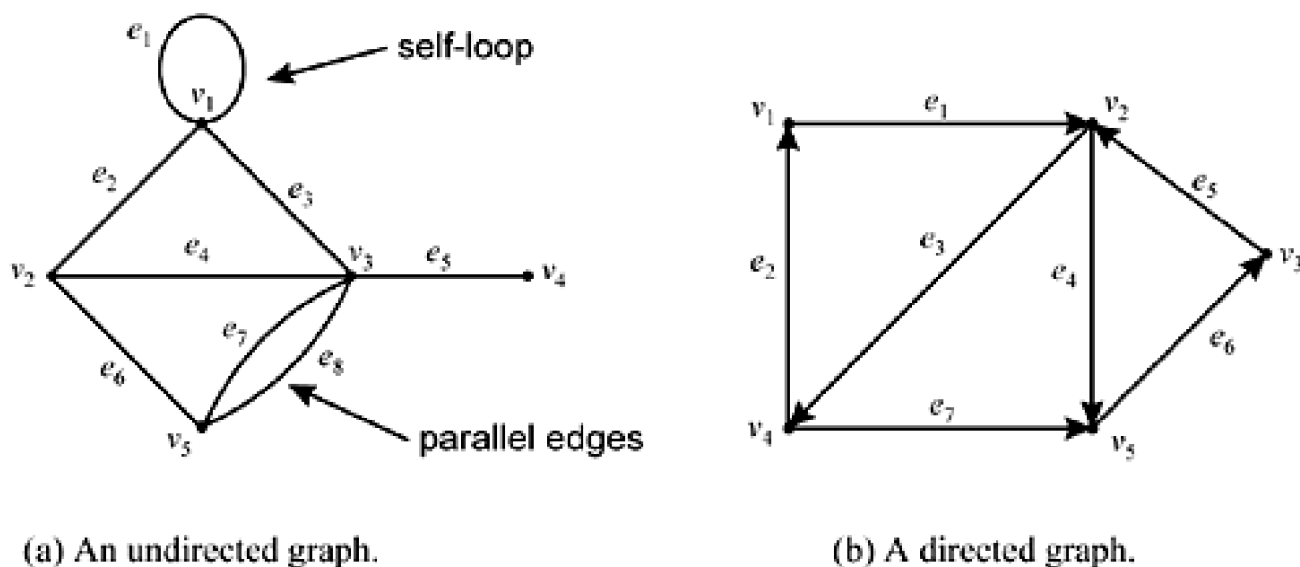
Vertices $\{u, v\}$ of a graph are *adjacent* if they are joined by an edge $e = \{u, v\} \in E$. Such an edge is said to be *incident* on the vertices u and v , which are also called the *end* vertices of e . Similarly, two edges are adjacent if they are incident on a common vertex and two edges are *parallel* if they are incident on the same pair of end vertices. While ordinary English would have it that all nodes in a connected graph are in a sense "neighbors," in graph terms vertices are *neighbors* only if they are adjacent.

A graph is *simple* if it has no parallel edges or self-loops. A *self-loop* is an edge that begins and ends on the same vertex. A graph with more than one edge in parallel between one or more pairs of vertices is called a *multigraph*. A graph in which a number w_{uv} is associated with every edge $\{u, v\}$ is called a *weighted graph* and the number w_{uv} is the *weight* of edge $\{u, v\}$. In transport networks these weights often represent cost, distance, utilization, or capacity. A graph where the edge weights represent capacities is also called a *capacitated graph* and problems involved with determining capacity—or solving some routing problem with given capacities—are called *capacitated* problems. This is in contrast to problems on simple graphs where capacities are not considered, only connectivity aspects of the graph itself. A capacitated graph and a multi-graph are close cousins. When the capacity is strictly integer we have two ways of representing the problem: either as an integer-capacitated simple graph, or, as a multigraph where the numbers of parallel edges on each edge correspond to the discrete capacity of each edge. Problems in data routing typically adopt a simple graph view, whereas most transport network problems require a capacitated graph or multigraph representation. In transport networking the *network graph* refers just to the topology of spans and nodes. A variety of weighted graphs can be employed for routing problems depending on the metric to be observed or minimized in routing. In other contexts the capacity units of a capacitated graph may be real-valued and thus present a weighted graph where the weights are edge capacities.

An edge is *directed* if the order of stipulating its vertices $\{u, v\}$ changes some property of the edge. Most commonly the direction of transmission is implied by the ordered nature of the vertex pair, if the edge is a directed edge. A directed edge is drawn by a line segment with an arrowhead indicating the direction. A graph with directed edges is called a *directed graph* or *digraph*. A graph is *undirected* if none of its edges are directed. A graph where every adjacent vertex pair has anti-symmetrical pairs of directed edges joining them is one way of representing networks with fully bidirectional transmission. More often networks where transmission is bidirectional (and of equal capacity) on each span are treated simply as undirected graphs. Thus, for transport networking the an undirected graph and an (anti-symmetric) *bidirectional* graph are equivalent concepts. Unless stated otherwise, our default is to treat transport networks as undirected graphs, in which all transmission capacity is implicitly bidirectional and equal in each direction on each span. For some problems and algorithms, explicit representation of the anti-symmetric directed graph view can be essential or advantageous. [Chapter 3](#) showed, for example, that Generalized Loopback and Oriented Cycle Double Cover methods benefit from certain efficiencies that arise from a directed graph orientation, that are not obtainable in the undirected representation. We will also see that a certain special structure of optimization problem to follow (the "transportation" problem) requires us to use an anti-symmetric directed graph model. Therefore, depending on the problem, switches from directed or undirected views even if all capacity is actually bidirectional. In the directed bidirectional graph every directed $\{u, v\}$ edge has an associated parallel $\{v, u\}$ edge. Unless a directed graph orientation is stated it is usually implicit that a

statement about the capacity of an edge (or span) refers to its transmission capacity in each direction. In the future, with more multi-cast traffic and Internet applications in general, a directed orientation for transport graphs may be more common if traffic flows themselves also become highly asymmetric. But a directed orientation is a fairly easy extension to any of the undirected design models to follow. [Figure 4-1](#) illustrates some of the terms so far defined.

Figure 4-1. Some graph terminology.



The number of vertices and edges in graph $G = (V, E)$ is denoted by $|V|$ and $|E|$, respectively. In graph theory the number of individual edges incident on a vertex is called the *degree*, or sometimes the *valency*, of the vertex. For example, v_3 in [Figure 4-1\(a\)](#) has a degree of five. If, as an example, the multigraph representing individual lightwave channel connectivity is drawn, it may have nodal degrees of tens or hundreds. More often in transport networking we are concerned with the degree of a node in terms of the number of distinct physical transmission spans incident on the node. This is a number between 2 to at the most about 7 or 8 for real transmission networks. The context for a statement of nodal degree therefore has to be clear. Our default will be to refer the physical layer graph degree, also called the *span degree* of nodes or just the "nodal degree" for short. It is understood unless explicitly stated the degree of a node in the transport network is the number of physically disjoint spans incident to it. Thus the span degree of v_2 in [Figure 4-1\(a\)](#) is three, but in the logical layer each of the incident edges may have a discrete capacity of 10 units, making the degree of v_2 in that layer $3 \cdot 10 = 30$ if it were represented by the corresponding multigraph.

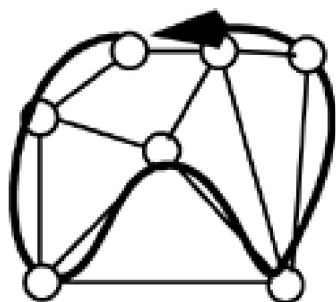
In graph theory, a *walk* is any sequence of adjacent edges in a graph. For example, $W = \{(v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)\}$. The vertices v_1 and v_k are called the *origin* and *terminus* of the walk, respectively. The number of edges in the walk, $k-1$, is called the *length* of the walk. A *trail* is a walk in which the edges are distinct. If all the vertices in a trail are also distinct, it is called *path*. If all vertices are also distinct (except possibly origin and destination), the path is called *simple*—i.e., it contains no loops. In transport networks we usually speak of *origin* and *destination* nodes and the paths between them over the graph. An origin node is sometimes also called *source* node, and a destination node can be called a *sink*. The matrix of all pair-wise requirements for transmission paths is called the *demand matrix*. Notation for referring to individual pairs varies. We refer to *O-D pairs*, *end-node pairs*, or "*relations*." In other literature readers may see *A-Z pairs*, *(s,t) pairs*, or *C-S pairs* referred to. On a simple graph, the term *path* in graph theory and *path* as commonly used in data networking correspond to each other. In practice, the terms *route* and *path* are often used interchangeably if the statement or context is in reference to a simple graph context—that is to say, without a specific structure of overlying capacity. Such a path is just a sequence of edges and essentially synonymous with the term *route*. In digital and multi-channel optical transmission networks, however, a *path* is a specific transmission circuit (in the electrical sense—or a contiguous lightpath) with a specific capacity and a specific structure in terms of the individual channels of which it is formed on each edge. A route and a path are thus *different base types* in a transport network, whereas they are often synonyms in other network contexts.

A walk (or a path) is *closed* if its origin and destination vertices are the same. In general, a closed walk is called *cycle* or *circuit*, and might figure-8 on itself and pass through the same vertex more than once. A closed simple walk is called a *simple* or *elemental cycle*. We will always mean simple or elemental cycles unless specifically stated. Traditionally in communication networks everything had to do with paths, often shortest paths, for the routing of the payload signals. With the addition of survivability this "linear" path orientation remains relevant for the working signals but research is showing that cyclic signal paths are highly relevant and seem to have a natural affinity for use in the protection role.

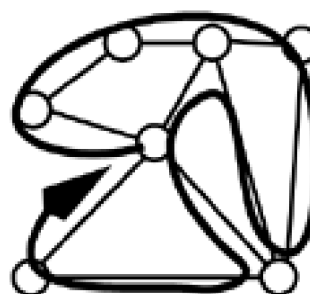
A cycle that connects all of the vertices in a graph, while passing through each only once, is called a *Hamiltonian cycle*. A (possibly

non-simple) cycle that traverses all edges of the graph exactly once (but may revisit the same vertices) is called an *Eulerian cycle*. Hamiltonian and Eulerian cycles are illustrated in [Figure 4-2](#). Eulerian cycles have special relevance to ring-based network planning because they protect every span in an on-cycle relationship. Hamiltonian cycles have a special relationship to p -cycle networks because a single Hamiltonian has a potential protection relationship to every span of the graph. (We do not, however, mean to suggest in either case that the network design problem consists of finding a *single* Eulerian ring or Hamiltonian p -cycle. An optimized ring or p -cycle capacitated network design will usually employ multiple cycles, not all of which will be either Eulerian or Hamiltonian. [Chapter 10](#) develops this point further).

Figure 4-2. Hamiltonian and Eulerian cycles.



(a) a Hamiltonian cycle visits all nodes once, not necessarily traversing all spans



(b) an Eulerian cycle crosses all spans once, but may revisit nodes

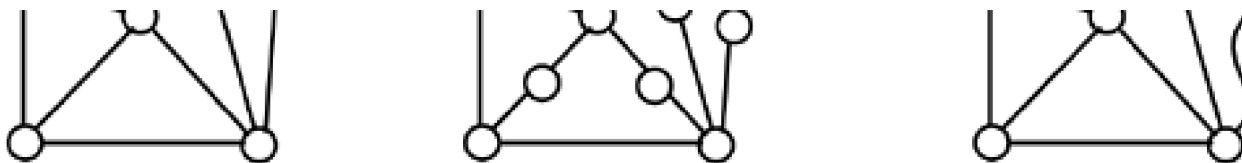
A graph as a whole is said to be a "Hamiltonian" (or "Eulerian") if it contains a Hamiltonian (or Eulerian) cycle, respectively. Eulerian graphs are easily identified in polynomial time because it is both necessary and sufficient that all nodes have *even* degree for an Eulerian cycle to exist in a graph. And yet the corresponding question "Does graph G contain a Hamiltonian cycle?" is an NP-complete decision problem. (NP-completeness concepts follow in [Section 4.2](#).) Hamiltonian cycles are, however, more likely to exist in highly connected networks. One result says that any graph of N nodes where every node has degree $d \geq N/2$ is Hamiltonian [\[GaTa92\]](#). An implication of this is that small well-connected networks (or subnetworks) are likely, or guaranteed to, contain a Hamiltonian cycle.

A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. Two vertices (not necessarily neighbors) are said to be *connected* if there exists a path between them in G . An undirected graph G is connected (as a whole) if there exists at least one path between every pair of vertices in G . Two graphs G and H are *isomorphic* if, despite different schemes of labeling nodes and spans, a 1:1 correspondence can be constructed between vertices in G and H for which, if there is an edge between two vertices in G if (and only if) there is also an edge between the corresponding nodes in H . Isomorphisms of a real fiber graphs are sometimes generated for use where the original geographically recognizable data is considered sensitive information. Tests to discover (or disprove) isomorphism include comparing the number of spans and nodes and the number of nodes at each nodal degree, and replacement of degree-2 nodes with single edges. If easily exhausted tests cannot distinguish two graphs, then a further procedure is to find two nodes, one in each graph suspected of being topologically the same vertex in the common isomorphic structure, and run Dijkstra's algorithm ([Section 4.3](#)) to develop the tree of shortest paths to all other nodes. Comparing such trees rooted at different source nodes may either establish complete node equivalences for the isomorphism or disprove it.

A closely related concept is *homeomorphism*. The difference between isomorphism and homeomorphism has only to do with degree-2 nodes. Two graphs are *homeomorphic* if a copy of one can be made from the other only by adding or deleting degree-2 nodes from any of its edges. Examples of homeomorphic graphs are shown in [Figure 4-3](#). The graph in [Figure 4-3\(c\)](#) is a homeomorphism of both other graphs (a) and (b) that is of special interest because it completely abstracts or summarizes what we call the true mesh connectivity of the other two graphs. It is a view only of the logical connectivity between nodes of degree 3 or higher. Some problems on mesh-restorable capacity and routing only depend upon properties of this abstraction. In other contexts we have to consider each degree-2 node in detail as well.

Figure 4-3. Three graphs that are homeomorphic.





A *plane graph* is a graph whose vertices are points in the plane $\mathbb{R} \times \mathbb{R}$ (where \mathbb{R} is the real number line) and whose edges are lines in the plane which have no intersection point other than at vertex points. Any graph that is *isomorphic* to a plane graph is called a *planar graph*. In practice this means there must exist some way of drawing the graph on a flat page (the plane), so that no edges cross. Physical layer transport networks tend to be almost, but not always, planar. This arises primarily from a tendency in the physical layer for nodes to be directly connected by facility routes only to geographical regional neighbor nodes. Furthermore, non-planarity would imply that two edges in the graph will intersect at some location other than a vertex, creating a single point of failure for both edges. Higher level logical transport graphs can be much farther from planarity, with logical edges between almost any pair of nodes.

One aspect of a connected planar graph is that it divides the $\mathbb{R} \times \mathbb{R}$ plane into separate non-overlapping regions called *faces*. Some faces are bounded (i.e., those interior to the graph) and some are unbounded (at an outside edge of the graph). *Euler's formula* relates the number of such faces $|F|$ to the number of vertices $|V|$ and edges $|E|$ of any planar graph:

Equation 4.1

$$|F| = |E| - |V| + 2$$

Note that $|F|$ includes the unbounded face in addition to all of the bounded faces. For instance, all of the graphs in [Figure 4-3](#) have 6 faces, and the graph in [Figure 4-1\(b\)](#) has 4 faces. To the extent that physical transport networks tend toward planarity, the number of interior faces can play a role in suggesting a number of rings used in ring-based designs. In fact there is a temptation in ring network design to lay one ring down as a bounding box of each face. While this is often stated to appear to be the "natural set of rings" for the network, such simple tiling of the plane with rings is usually inefficient because it makes for many ring-span overlaps. Although our scope is limited to mesh networks, p -cycles are in some senses ring-like. [Chapter 10](#), which is devoted to p -cycles, contains some illustration of this point.

[Figure 4-4](#) and [Figure 4-5](#) illustrate a few further points about planarity. First, a planar graph may not look planar upon first sight. Secondly, an interesting result called Kuratowski's theorem is that a graph is planar if and only if it contains no subgraph that is homeomorphic to $K_{3,3}$ or K_5 . K_n denotes what is called a *complete graph* on n vertices where every vertex is connected by a unique edge to all $(n-1)$ other vertices. $K_{n,m}$ denotes a complete bipartite graph on n and m vertices. This is a graph where the vertices are grouped into two sets $|V_1| = n$, $|V_2| = m$ and each vertex is connected by a unique edge to all vertices of the other vertex set (i.e., of which it is not a member).

Figure 4-4. Two graphs that are planar: (a) is evident, (b) is seen by pushing an edge out of the way to obtain (c).

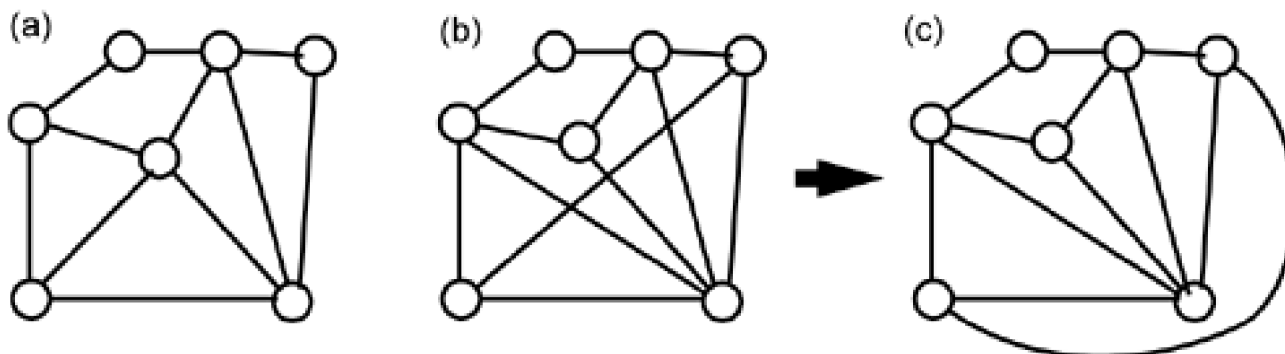


Figure 4-5. The two fundamentally non-planar graphs $K_{3,3}$ and K_5 .

(a) $K_{3,3}$



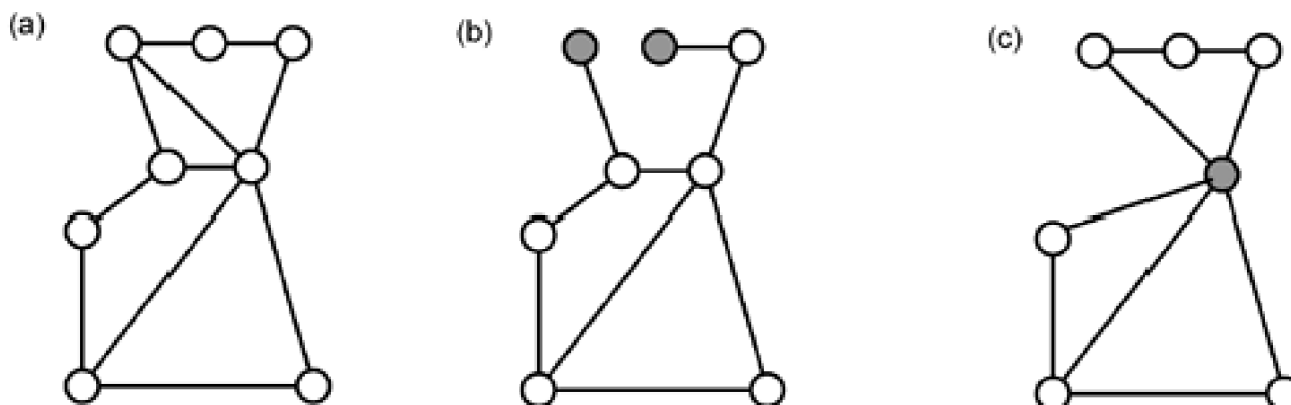
(b) K_5





A graph is *two-edge connected* if it has at least two edge-disjoint paths between every pair of vertices. A graph is *two-vertex connected* or *biconnected* if it has at least two vertex-disjoint paths between every pair of vertices. Graph connectivity properties are important for transport networks because to survive all single failures, the graph must be at least two-edge connected for it to be topologically possible to protect or restore failed spans. Characteristics of a biconnected network are quite recognizable by visual inspection. Such a network will have no degree-one or *stub* nodes and is *bridgeless*, or in other words, has no *articulation points*. Articulation points are also called *bridge nodes* or *cut-nodes*, and are nodes whose single failure would partition the graph into two disconnected parts. Figure 4-6 illustrates this. The property of two-edge connectedness is also referred to simply as *two-connectedness*. Either two-connectedness or bi-connectivity is suitable for survivable network design against span failures, but bi-connectivity is preferred to avoid there being any single node whose failure would break the network into two disconnected pieces. In general in following sections, we are not dependent on whether a graph is two-connected or biconnected, but it must have at least one of these properties. We refer to such graphs in general as *closed*. In practice, however, virtually all survivable fiber-based transport networks are biconnected in their physical layer graph.

Figure 4-6. A biconnected network (a) has no degree-one sites, as in (b), and no articulation points or bridge nodes, as in (c).



If we view transmission systems as providing pairs of symmetric bidirectional edge pairs, the directed network that results in this viewpoint is a *strongly connected digraph*. A digraph is strongly connected if for every ordered pair of vertices (v, w) , there is a directed path from v to w .

A *tree* is a connected undirected graph containing no cycles. A *forest* is a set of trees. In a connected undirected graph there is at least one path between every pair of vertices. The absence of a cycle in a connected undirected graph implies that there is at least one pair of nodes for which there only one disjoint path between the nodes. Therefore, in a tree there is only one path between every pair of vertices. A *spanning tree* is a subgraph of G that contains every vertex in G . If G is not connected, the set of spanning trees for each connected component is called a *spanning forest*.

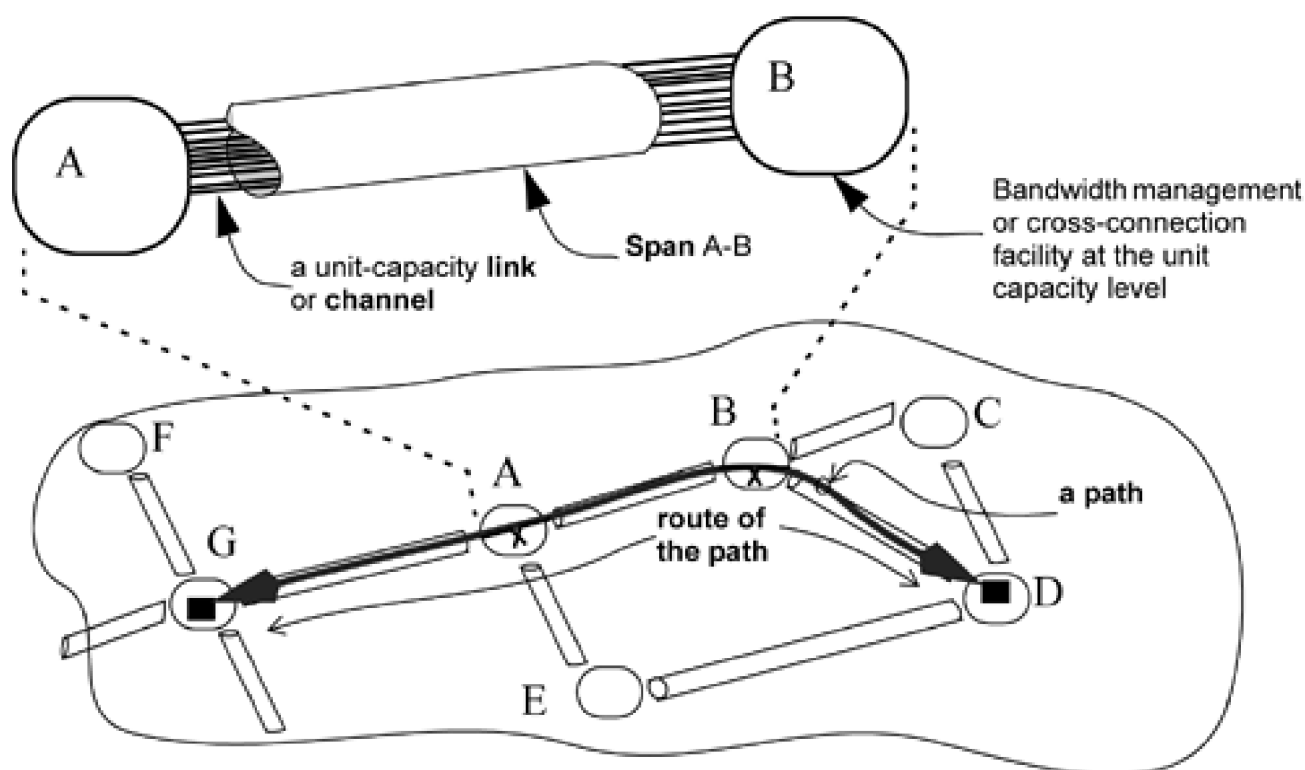
4.1.3 Transport Network Terminology

In graph theory, the terms arc, edge, or link are essentially interchangeable for the connection between adjacent nodes and terms path and route are synonymous in referring to a connected sequence of edges. In transport networking, however, we are always dealing with a capacitated graph or discrete multigraph view of the graph because each physical transmission system provides a discrete set of logical point-to-point capacity units in parallel between its transmission terminals. These are typically fibers, lightpaths or OC-n signal units. These unit-capacity transmission links or channels are cross-connected to form paths to bear the logical point-to-point requirements of the services layer between node pairs.

In this book, the basic unit of capacity on which switching devices operate to inter-connect capacity is called a *link* or a *channel*. The

physical entity that is the collection of all such unit-capacity links in parallel between neighbor nodes is called a *span*. A span is subject to physical damage that will affect all its links simultaneously. A *route* is defined as a concatenation of spans (i.e., simply a geographical route over the fiber map) and a *path* is a specific cross-connected sequence of individual links (or channels) on spans. Thus a path provides a specific unit-capacity signal propagation sequence embedded within, and cross-connected through, a sequence of transmission spans. [Figure 4-7](#) summarizes these basic concepts and terminology as used in this book. Note that every path has a route and multiple paths may follow the same route, but path and route are distinct concepts.

Figure 4-7. Concepts of span, link (or channel), route and path in transport networking.



Precise and consistent use of the terms *link*, *span*, *path* and *route* facilitates discussion of a wide range of arguments set in a generic transport network context. In a SONET context a link may correspond to a single fiber bearing an OC-48 and the span is the cable containing many such fibers. Paths are end-to-end OC-48 connections. If cross-connection is at the OC-48 level then a fiber bearing an OC-192 represents four links on the span. In a DWDM context with cross-connection at the lightwave level then each lightwave channel corresponds to the generic "link" entity and paths are end-to-end lightpaths. As we use them, the terms "span" and "link" have their origins in the transmission community. As Bhandari explains, *span* and *link* allow us to distinguish between the entities that map into logical links of higher levels and the physical transmission spans over which all end-to-end paths (logical links in higher layers) are established:

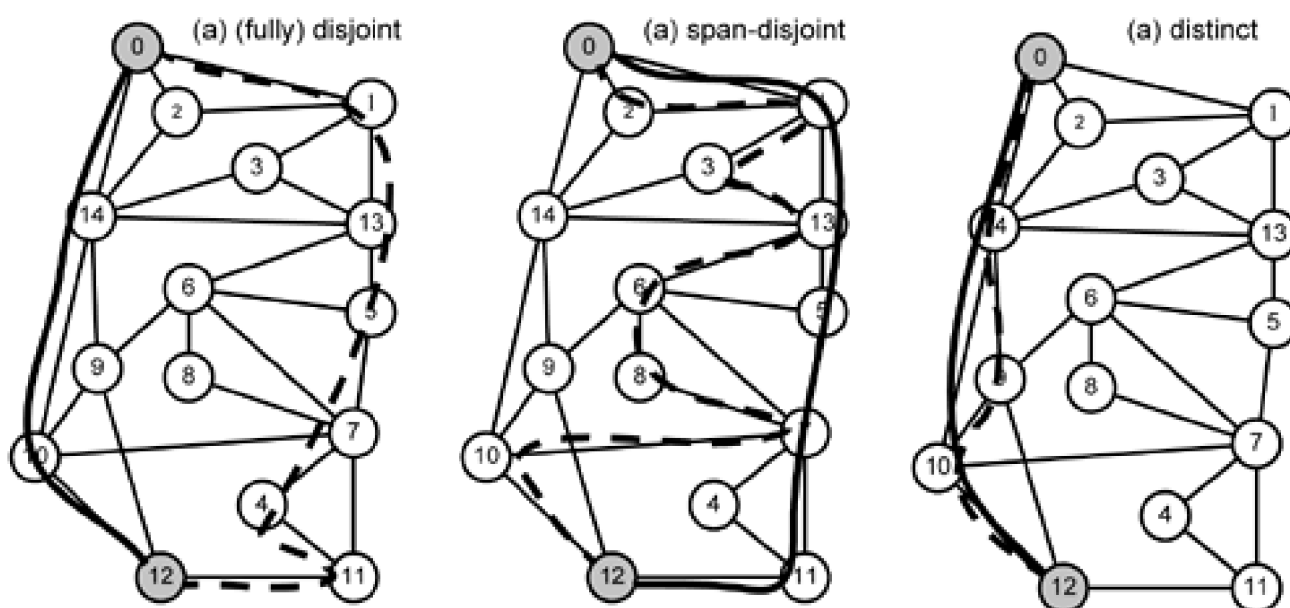
"In reality, networks such as the telecommunication fiber networks can be more complicated than the traditional graph-theoretic networks described by vertices and edges... spans are the actual physical links that comprise the physical network. Links (or edges) of the (logical) network are built from spans. As a result, a given span can be common to a number of links...The physical layout of a fiber network consists of links, where each link (also sometimes called a logical connection) connects a pair of nodes: a link is in general a series of contiguous physical connections called spans." [\[Bhan99\]](#) page 117.

There is, however, no standard usage of terms in the industry as a whole. *Duct* or *cable* is sometimes also used as a synonym for span. A span is also referred to as the "default SRLG" in the framework of shared risk link groups [Section 1.5.4](#)). But the main thing to be cautious of misinterpreting is "link." In many sources "link" refers to what we call a span and may even refer to a path when viewed from the service layer. As a practical matter the term link has lost all precision because it is used so generally for almost any notion of interconnection. We therefore increasingly rely on the term channel to replace it, but "link," as defined above, is still found in prior literature, where it refers to a managed unit of cross-connected capacity, not the entire physical connection between two nodes. If readers are generally aware of this confusion they can, however, usually infer the intent when authors use "link" for everything, from the context. Generally in past literature in the transmission engineering community, link has usually meant a single-channel capacity unit (as here). But in much current literature for optical networking, link means what we call a span. The point is not trivial when one considers the confusion it causes in reference, for example, to a "*k*-shortest link-disjoint paths" algorithm.

4.1.4 Distinct and Disjoint Routes

Two routes are *distinct* if they simply differ in any detail. Routes are *disjoint* if they have no elements (spans or nodes) in common. Sometimes for emphasis or clarity one sees "fully disjoint" or "node and span disjoint" but both are what "disjoint" strictly implies. An exception is commonly implied to allow the end nodes to be the same when talking about disjoint routes. For instance, a 1+1 protection scheme may be arranged between two nodes by established two paths with common end points following disjoint routes at all locations but the end nodes. Two routes are *span disjoint* if they have no span in common, although nodes other than the end nodes may appear in common between the routes. [Figure 4-8](#) shows the differences.

Figure 4-8. Paths having (a) fully disjoint, (b) span disjoint and (c) distinct routes.



4.1.5 Data Representations of Graphs

The information to specify a graph for computer or mathematical purposes is usually represented by an *adjacency matrix* or *adjacency list*. An adjacency matrix is an $n \times n$ matrix where non-zero entries represent the existence of an edge. For an undirected graph, a 1 in the matrix at row i , column j represents a bidirectional edge between nodes (i,j) (and the entry in row j , column i is ignored). If the graph is directed the matrix coordinates (i,j) and (j,i) contain independent indications of whether an edge exists. If the matrix entries are strictly 1/0 indications then the matrix encodes only edge incidence information and is called an *incidence matrix*. If an edge exists, however, it is easy to use the adjacency matrix to encode additional attributes such as capacity or edge distance.

A more efficient and compact representation of a network is an adjacency list. One form of adjacency list can be thought of as a set of lists where each vertex appears once, associated with a listing of the other vertices to which it is directly connected. Thus an edge (v,w) exists if w is a member of the v^{th} sub-list. For a graph with directed edges, no edge is specified more than once in the type of adjacency list just described, because directed edges (v,w) and (w,v) are distinct entities. For undirected graphs, however, (w,v) is redundant if (v,w) is listed earlier. Using linked-list techniques each sub-list entry can also be a data record describing the attributes of the edge such as its length, capacity, cost, and so on.

The same information may also be presented as a single flat list of the (v,w) vertex pairs for which edges exist. This format of a single list of spans that exist implicitly specifies all vertices that exist in the network as well as the edges between them.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

4.2 Computational Complexity

We are now heading into a series of sections that introduce useful algorithms for routing and other operations on graphs, followed by introduction to a number of basic network optimization problems. While treating each algorithm and the optimization problems it is natural to consider the inherent computational complexity of each. To do so, we must equip ourselves with some background on the concepts and conventions of algorithmic complexity.

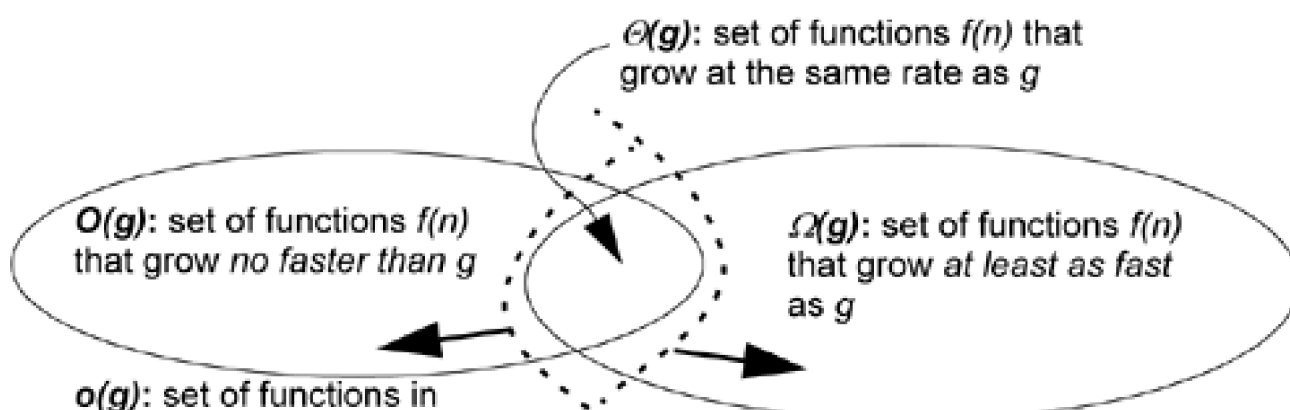
An important point is that there are many easily posed problems, especially combinatorial optimization problems, that computers simply cannot solve exactly. And it's not just a matter of getting a faster computer or more RAM because these are problems for which the solution space explodes exponentially (or even factorially) with problem size, making the number of possible solutions astronomical. A network planner can easily pose problems where the number of solutions to be checked exceeds the number of elementary particles in the universe. A large number of important real-world problems in fact behave this way. It is important, therefore, to identify and exploit special cases and classes of problems that *can* be efficiently solved. Other important basic problems such as shortest path routing or finding maximum flow are in the class of "polynomial time" problems and can be solved quickly and exactly in practice at almost any required size. Asymptotic notation provides a means for conveying and comparing properties of different algorithmic approaches and helps us assess the complexity of overall methods that might combine several such basic algorithms. An excellent introductory article to key concepts about combinatoric problems and algorithmic complexity principles is [\[Kar86\]](#),

4.2.1 Asymptotic Notation

The complexity of an algorithm is a measure of the execution time or memory needed as a function of the problem size. The two basic measures are referred to as time- and space-complexity, respectively. Complexity can be empirically observed or determined theoretically. Empirical analysis is used to predict the average case behavior of the algorithm by running it on a suitable number of problem sizes and observing the run time. Theoretical analysis more often predicts only the worst-case behavior the algorithm might exhibit. Complexity is expressed in a notation which makes statements about the rate at which complexity increases with problem size. More formally, the notation of asymptotic order is really making statements about the set of all possible functions to which an observed or predicted behavior belongs. The following treatment is based on that of [\[BaGe00\]](#).

If some function g is stated, then [Figure 4-9](#) shows conceptually how all other functions must be asymptotically related to g . The families of functions shown and the notation for them are discussed below. Note that we are talking about functional forms which are equivalent in the way they grow at large n , so we do not care about exact relative function values at small n .

Figure 4-9. Classification of functions based on growth rates for asymptotic order notation relative to some stated functional form $g(n)$.



$O(g)$ that become small relative to g

$\omega(g)$: set of functions in $\Omega(g)$ that become large relative to g

Mathematically, to make a statement that $f \in O(g)$ means that

Equation 4.2

$$0 \leq \lim_{n \rightarrow \infty} \left\{ \frac{f(n)}{g(n)} \right\} < C$$

where C is finite or zero. In other words, it says that at suitably large problem sizes $f(n)$ is growing *no faster* than $g(n)$. Thus, a statement that the run time, $f(n)$, of maximum flow is $O(n^3)$, really means that $f(n) \in O(n^3)$ and tells us that whatever the actual function describing max-flow run time is, its functional form is asymptotically *no worse than* n^3 . This is informally called "Big Oh" notation. More generally, it means that there exist constants C and n_0 such that

Equation 4.3

$$|g(n)| \leq C \cdot f(n)$$

for all $n \geq n_0$. $f(n)$ is some mathematical function of n such that a positive real multiple of $f(n)$ exceeds the amount of resources consumed by the given algorithm. n is the problem size, and n_0 is the threshold value for the problem size beyond which [Equation 4.3](#) is valid.

At the other extreme, if $f \in W(g)$, it means that

Equation 4.4

$$0 < \lim_{n \rightarrow \infty} \left\{ \frac{f(n)}{g(n)} \right\} \leq \infty.$$

In other words, $W(g)$ is the set of all functional forms that are ultimately smaller in magnitude than the actual functional behavior. These are functions that are *definitely more slowly growing* than $f(n)$. For example, in practice we might be uncertain about the actual asymptotic growth rate of some algorithm, but know from first principles, say, that it certainly must exceed that of max-flow. Then a statement that " $f(n) = \Omega(n^3)$ " or $f(n) \in \Omega(n^3)$ would correctly state that whatever the actual function describing max-flow run time is, i.e., that its functional form is asymptotically at least as strong as n^3 . This is called "Omega notation."

This leaves the case of $f \in \Theta(g)$ which, as [Figure 4-9](#) suggests, is the subset of functions in $O(g)$ that grow exactly as g in the limit. (This is not-surprisingly called "theta notation.") [Equation 4.2](#) applies again because $f \in \Theta(g)$ implies $f \in O(g)$, but $f \in \Theta(g)$ is the special case where $0 < C < \infty$ is a definitely non-zero finite constant. In practice the difference in usage is that $f \in O(g)$ only states an upper bound on the asymptotic form, whereas $f \in \Theta(g)$ is the stronger statement that $f(n)$ grows at exactly the same rate as $g(n)$ in the limit.

The formal differences in the definition of $\Omega(g)$ and $O(g)$ are subtle. The range of the limit in [Equation 4.2](#) includes zero and the constant may be arbitrarily large but is definitely finite whereas in [Equation 4.4](#), the limit is unbounded in size and strictly cannot be zero. Notionally near these limits (when $C \approx 0$ or very large but not ∞) we are dealing with functions that are only barely within $f \in W(g)$ or $f \in O(g)$. Sometimes we might want to make reference to functional forms that are much more solidly within the respective sets. This is where the

notations $o(g)$ "little o of g" and $w(g)$ "little omega of g" are used, implying the following, respectively:

Equation 4.5

$$f \in o(g) \rightarrow \lim_{n \rightarrow \infty} \left\{ \frac{f(n)}{g(n)} \right\} = 0; \quad f \in \omega(g) \rightarrow \lim_{n \rightarrow \infty} \left\{ \frac{f(n)}{g(n)} \right\} = \infty.$$

Thus $f(n) \in o(g)$ allows that $f(n) \in O(g)$ is still a true statement but that $f(n)$ is also no border-line case: it is solidly within $O(g)$ and more specifically must be one of the smaller growing functions in $O(g)$. $w(g)$ expresses the corresponding notion about $f(n)$'s membership in $\Omega(g)$.

4.2.2 P and NP

An algorithm that solves a certain problem is said to be *efficient* if there is any polynomial $f(n)$, even $f(n) = n^{100}$, that upper-bounds the worst case run times as problem size increases, for a suitable constant C . The class of all problems for which polynomial algorithms exist or can exist by transformation from other existing algorithms is called P . A wider class of problems (that includes P as a subset) is the set of problems, called NP , for which a proposed solution can be *checked* (i.e., recognized as a solution if it is one) in polynomial time. Note that NP does not stand for non- P , it stands for the class of *nondeterministic polynomial problems*. These are problems for which you could recognize a solution efficiently if a guesser presented it, but for which there is no guarantee that all possibilities may not have to be tested, or, if tricks are known to reduce the number of possible solutions to test, the number to be checked may still exceed any polynomial bound. NP thus includes P . An open question in mathematics is whether P and NP are equivalent.

A problem is said to be *NP-hard* if it is at least as hard as any problem in NP and is, strictly, a statement that can apply to problems both in NP and outside NP [BaGe00] (p.561). (There are other classes of problems outside NP but beyond our scope.) Many optimization problems of real-world importance are *NP-hard*. In practice, this means only that no efficient (i.e., polynomial time) algorithm is known for them (and likely does not exist). More theoretically it means that if a solution for the related *decision problem* version of the problem at hand is proposed (by a "guess generator") you can at least test the solution for validity in polynomial time. There is, however, no assurance that you would not have to exhaust and test all possible solutions in this way.

A related concept is *completeness*. Completeness of a problem is not simply a higher ranking of difficulty per se. Rather, it is a special type of problem. A problem X is complete if you could solve any other problem of this class (even problems not yet posed) in polynomial time, given that you had a polynomial time algorithm for X . Hence complete problems are hardest in their own classes. To the practicing network planner NP -hard and NP -complete are not different in their practical implications: both imply the possibility of exponentially increasing run-times with problem size. The distinction of complete versus hard problems is more important to researchers who need to choose an NP -complete problem to work on to advance solving techniques for the entire class. So NP -hard problems are not necessarily less difficult than NP -complete. Graph coloring, Knapsack, Satisfiability, Traveling Salesman and Hamiltonian cycle problems are but a few of the

classic NP -complete problems. These problems vary from being in $Q(2^{\sqrt{n}})$ to as bad as being in $Q(n!)$. Satisfiability was the first provably NP -complete problem. Since then, by transformation proofs to satisfiability, other provably NP -complete problems have been inventoried. A detailed listing of other known NP -complete problems is given in [GaJo79] which is also a valuable guide to NP complexity in general. Note that the definition of problems in NP is formally in terms of *decision problems*, not explicit algorithms for synthesis of the solutions themselves. Decision problems are cast as yes/no-answerable or "decidable" problems. The engineering context of a problem may be to find the shortest Hamiltonian cycle in a graph. But the corresponding decision problem to which complexity theory formally applies would be: Does a Hamiltonian cycle of length less than X exist? As a simple conceptual guide to practitioners we offer a quick summary of the various classifications in [Figure 4-10](#).

Figure 4-10. A practitioner's thumbnail guide to P, NP, NP-hard and NP-completeness.

Problem class:	Practical meaning or implications:
-----------------------	---

- *P* The set of problems for which a polynomial time algorithm exists, or can be developed, for exact solution of the problem. *Example:* The shortest path through a graph is in $O(n^2)$.
- *NP* The set of all problems (cast as the corresponding decision problems) for which a *guess* at the solution can be validated in polynomial time. *Example:* Is there a tour of this graph that is less than 20 miles in total length? NP *includes* all problems in P.
- *NP-hard* This is a statement that a problem is “at least as hard as any problem in NP.” Any solution algorithm will inevitably have to examine all possible solutions or still exhibit run-times that are not bounded by any polynomial function.
- *NP-complete* Same as for NP-hard but with the additional significance of having been proven as a “complete” problem for the class NP. If you could solve any NP-complete problem in polynomial time, then you would be able to solve any other NP-complete problem also in polynomial time.

Let us now look at algorithms and optimization methods for network problems. We will start with the useful basic ability to find shortest paths on a graph.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.3 Shortest Path Algorithms

A frequent basic function is to find the shortest, least cost, or other minimum-weight route between nodes in a network. This is generically called the *shortest path problem*. When we are considering graphs with non-negative edge-weights, we can use the original Dijkstra's shortest-path algorithm [Dijk59]. Negative edge weights do not arise in real networks directly, but they do occur in intermediate steps in the later max-flow and min-cost disjoint path pair problems. In these cases Dijkstra's algorithm may fail to find the shortest path that exploits the negative edges to reduce its cost. For negative edge weights we can use the Fulkerson-Ford shortest path algorithm [FoFu62] or the modified Dijkstra algorithm below by Bhandari [Bhan99]. Let us first describe the basic Dijkstra algorithm in depth. An in-depth understanding of Dijkstra's algorithm is worthwhile because it is a fundamental subproblem in many other algorithms and it also introduces the concepts of labeling and scanning as a basic algorithmic technique that can be easily modified for a variety of other search purposes. A look at the internals of the Dijkstra algorithm also reveals that in finding the single shortest path, the shortest-path tree from source to all other nodes is actually built. This can be exploited in contexts where full sets of routing solutions are actually needed. Specific advantage of this has been taken in the fast k -shortest paths algorithm outlined later [MaGr94].

4.3.1 Concepts of Labeling and Scanning

A fundamental aspect of Dijkstra's algorithm is the concept of *labeling* nodes. Labeling marks a node with the minimum distance so far discovered from the source to the node, without any record of the route followed, but with a pointer back to the neighbor node through which one can arrive at the given node with this distance. The neighbor node in this context is called the *predecessor* node. A label thus has two attributes: $Label = \{distance, predecessor\}$. Labels are initially *temporary* until a certain stage in the algorithm where a given label is found to have the lowest distance over all the nodes *scanned* in a given iteration. It then becomes a *permanent* label. Scanning is the process of looking outwards from other nodes to all their adjacent nodes that are not permanently labeled and updating their (temporary) labels with new distance and predecessor information if the updated distance is shorter. Thus, scanning is as follows:

```
procedure ScanFromNode( $i$ ) {
  for every adjacent node  $j$  not already in [P] {
    if  $D_j > D_i + d_{ij}$  {
      Label@[ $j$ ] := ( $D_i + d_{ij}, i$ )
    } else {
      make no change }}}

```

where [P] is the (ordered) set of permanently labeled nodes, d_{ij} is the distance of the edge between nodes i and j , and D_i is the distance part of node i 's label.

4.3.2 The Dijkstra Algorithm

The basic procedure for finding the shortest path from source node s to target node t is:

```
procedure Dijkstra(source  $s$ , target  $t$ ) {
  CurrentNode :=  $s$ 
  initialize each node's Label, [P], and [D] to null

```

```

append node s to [P] and  $D_s$  to [D]
do while node t not in [P]{
  ScanFromNode(CurrentNode)
  find node x not in [P] with minimum  $D_x$ 
  append node x to [P]
  append  $D_x$  to [D] (vector of  $D_j$  values for nodes in [P])
  CurrentNode := x }

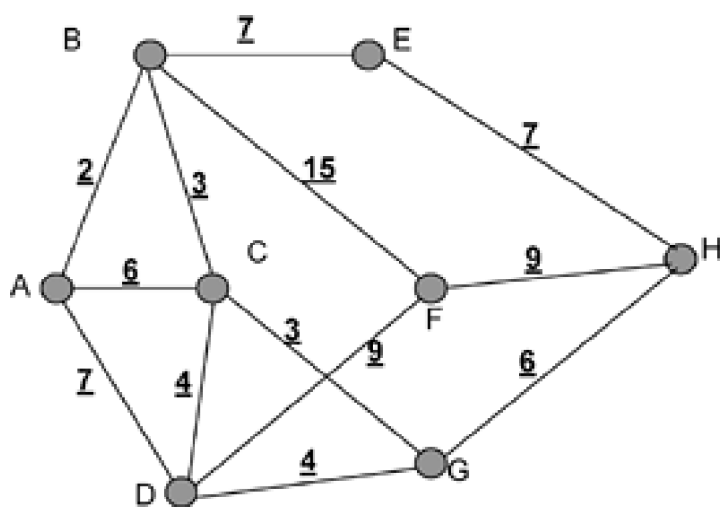
```

Upon exiting, the vectors (or ordered sets) [P] and [D] encode a shortest path tree rooted at the source node and including possibly all other nodes (but certainly the target node) as leaf nodes on the tree. From this information the route of the shortest path can be read out as well as the associated distance. If the shortest path tree to all other nodes, not just the target t is desired, the "do while node t not in [P]" condition is changed to continue iteration until all nodes are permanently labeled.

Example

[Figure 4-11](#) shows a graph with edges with distance weights associated with them. We will use this graph for an example illustrating Dijkstra's shortest path algorithm to find the shortest path from source A to target H. [P] will be the ordered set of permanently labeled nodes and [Dp] is the corresponding vector of final distances. Note in what follows that "+" means set addition. For example $[P] := [P] + C$ means node C is added to the existing contents of set [P]. To start out:

Figure 4-11. Graph for examples of Dijkstra, max-flow, and k -shortest paths algorithms.



Notes:

For the explanation of Dijkstra's algorithm in Section 4.3 the edge weights represent distances.

For the explanation of k -shortest paths in Section 4.5 the edge weights represent transmission capacities (i.e., unit link quantities).

$[P] := [A]$; $[Dp] := [all\ infinite]$

Step 1. Starting at the source, scan all neighbors: result is: $d[B,C,D] := [2,6,7]$

The temporary labels assigned to nodes B,C,D are $\{2,A\}$, $\{6,A\}$, $\{7,A\}$ respectively.

$d_{min} = 2$ (at node B) so node B becomes permanently labeled: $\{2, A\}$. The predecessor set has node B added to it $[P] := [P] + B$; i.e., now $[P] = [A, B]$, $[Dp] = [-, 2]$.

Step 2. Go to the last node to receive a permanent label, node B, and scan from there.

Scanning from B, nodes E, F receive their first temporary labels:

$d[E] := d[B] + 7 \Rightarrow$ label $\{9, B\}$;

$d[F] := d[B] + 15 \Rightarrow$ label {17, B};

and the existing temporary label at C {6, A} is updated, since a lower distance is discovered: $d[C] := d[B] + 3 = 5$, hence the new label for C is {5, B}.

Next, checking all temporary labels network wide, we find $d_{\min} = 5$ at node C. Hence node C becomes permanently labeled {5, B} and $[P] := [P] + C$,

$[Dp] := [Dp] + 5$,

$[P] = [A, B, C]$,

$[Dp] = [-, 2, 5]$.

Step 3. Go to node C and scan from there: nodes G, D are scanned. (Node A is already permanently labeled).

Node G gets its first temporary label: $d[G] := d[C] + 3 = 8$; label = {8, C}.

Node D already has a temp. label {7, A} and is now re-scanned from node C. The new distance would be $d[D] := d[C] + 4 = 9$ which is greater than that of the existing label, so no change is made to node D's label.

Next, checking globally for d_{\min} , we find $d_{\min} = 7$, at node D hence node D gets permanent label {7, A},

$[P] := [P] + D$,

$[Dp] := [Dp] + 7$.

Step 4. Go to node D and we scan nodes G, F. (Nodes C, A are already permanently labeled).

$d[G] := d[D] + 4 = 11$... exceeds existing temp label distance {8, C}, so no change.

$d[F] := d[D] + 9 = 16$... which improves on the existing label {17, B}, so node F gets a new temporary label: {16, D}.

$d_{\min} = 8$ found at node G; perm. label @ G = {8, C},

$[P] := [P] + G$,

$[Dp] := [Dp] + 8$.

Step 5. Go to G and scan H. (Nodes C, D are already permanently labeled).

$d[H] := d[G] + 6 = 14$,

$d_{\min} = 14$ (at H); perm. label @ H {14, G},

$[P] := [P] + H$;

$[Dp] := [Dp] + 14$.

At this point we would also note that H is the target node, and has received a permanent label, so we would have found the shortest path A to H as required, and we could exit at this point. However, continuing will show more generally how Dijkstra's algorithm can (and for any given source-target pair, may have to) find the complete tree of shortest paths rooted at the source node.

Step 6. Go to node H and scan nodes F, E:

$d[F] := d[H] + 9 = 23 \rightarrow$ no change from existing label {16, D},

$d[E] := d[H] + 7 = 21 \rightarrow$ no change from existing label {9, B},

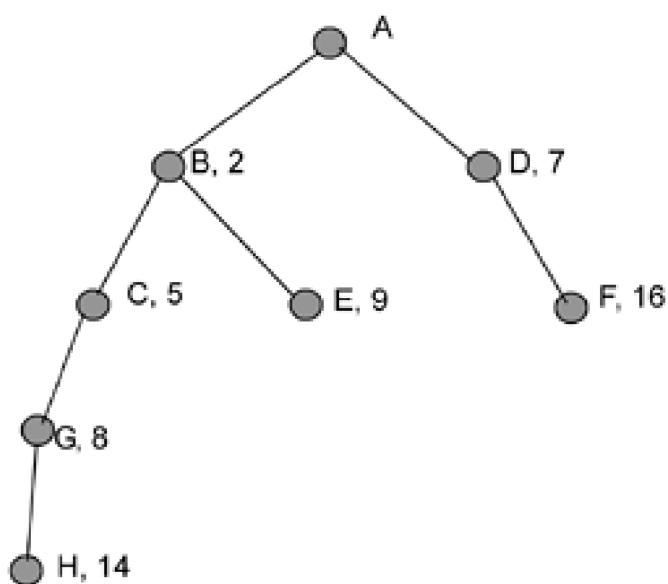
$d_{min} = 9 \rightarrow$ perm. label at E: {9, B}.

At this stage, only node F's label has not been made permanent. And there are no neighbors of node E to scan (B and H both have permanent labels). So at this point the "extended" algorithm halts and all labels can be considered permanent. Now if we look at the contents of the predecessor set and the distance vector, [Figure 4-12](#) shows how the complete shortest path tree is encoded by them. Thus, the answer to the original question is that the shortest path from A to H has a distance of 14 following route {A-B-C-G-H}.

Figure 4-12. The set of permanent labels at the end of an extended Dijkstra run encode the shortest path tree from the source node to all other nodes.

Shortest Path Tree

step	node	perm. label
0	A	(source)
1	B	{2, from A}
2	C	{5, from B}
3	D	{7, from A}
4	G	{8, from C}
5	H	{14, from G}
6	E	{9, from D}
	F	{16, from D}



Because the algorithm may have to complete the entire shortest-path tree on any given call, implying it has to visit every node once as a recently permanently labeled node from which to do the next scan, and because the scanning process could involve $n-1$ other nodes in the worst case, the time-complexity of Dijkstra's algorithm (as so far described) is $O(n^2)$ where n is the number of nodes. However, by use of various data structures operated upon by sort routines at each iteration, it is also possible to trade space or storage complexity for time complexity realizing versions of Dijkstra's algorithm that run in $O(n \log n)$ [\[MaGr94\]](#) or $O(S \log n)$ [\[Moy98\]](#) (pp. 97-98) time where S is the number of edges in the graph.

Note that while this example used the edge weights of the graph in [Figure 4-11](#) to represent distances, the same algorithm can be used to find minimum logical-hop distances. In the latter case the measure of distance is "1" for every edge traversed in a route. This "hop distance" measure of shortest path length is sometimes used in studies where distance data is not available or, more importantly, because general insights can more often be obtained from test cases where every edge has equal length. Purely logical topology-related effects of the graph structure are then more dominant, whereas in general with any specific set of edge distances, results can be influenced by the specific distance values employed, making it harder to obtain general insights about effects due solely to topological phenomena or interactions. In addition, hop length as a measure of distance is also used where the view is that nodal equipment costs dominate span distance-related costs or where some routing performance measure relates more to the number of queueing or switching points en route than to sheer distance.

Other shortest-path algorithms

In practice Dijkstra's algorithm serves well for most shortest-path finding purposes on networks. Dijkstra's algorithm is also a basic part of the OSPF protocol for routing in the Internet [[Moy98](#)]. The *Bellman Ford* or the *Ford-Fulkerson-Floyd* shortest-path algorithms are variations on Dijkstra's approach that permit them to handle edges with negative weights, as long as there is no directed cycle on the graph with overall negative weight around the cycle. In management planning problems the negative edge weights may have meaning in terms of scheduling or inventory issues, etc. Bellman's main variation is to scan the nodes in the order they are labeled, eliminating the requirement to find the smallest label, making it possible that a node may be scanned more than once in an iteration. Bellman's worst-case theoretical time complexity is therefore $O(n^3)$ but is reported in practice often to exhibit $O(S)$ behavior, especially where the average nodal degree is small and long paths tend to have more hops than shorter paths. The differences in the Ford-Fulkerson-Floyd algorithm make it more of a breadth-first search type of Dijkstra algorithm that is somewhat better suited to a decentralized implementation and may have advantages in finding all pairs of shortest paths as opposed to just one shortest-path tree as in Dijkstra. Kershenbaum [[Kers93](#)] gives a thorough comparative treatment of the two latter algorithms. Robertazzi [[Robe99](#)] also covers the Ford-Fulkerson-Floyd algorithm in more depth.

[[Team LiB](#)]

◀ PREVIOUS

NEXT ▶

4.4 Bhandari's Modified Dijkstra Algorithms

4.4.1 BFS Dijkstra

Bhandari [[Bhan99](#)] gives two useful variations on the basic Dijkstra algorithm. One, called *Breadth-First Search (BFS) Dijkstra* is a speed-up technique for use on relatively sparse non-negative graphs. The difference is that it uses scanning from all of the nodes where labels were updated in the prior iteration. This initially involves scanning from more nodes than regular Dijkstra, as the BFS effect "fans out." But as soon as the destination node itself has been reached and receives its first non-infinite distance, $d(Z)$, all nodes with labeled distances $d(j) > d(Z)$ are rejected as nodes from which to do scanning in the next iteration. This continues as $d(Z)$ itself improves, rapidly killing off the initial fan out. Although a node may be scanned from several other nodes in one iteration, the basic logic of scanning is the same: If the node i has label distance greater than the current $d(j) + l_{ji}$ when scanned from node j , then its label is updated. The order of parallel scanning within the iteration may affect which of equal-distance routes is reflected in a node's temporary label, but not that a current shortest distance will be left at the end of the iteration. The stopping condition is permanent labeling of the destination node as usual. Bhandari reports observing up to five-fold speed-up over regular Dijkstra in sparse graphs of about 100 nodes [[Bhan99](#)] (p.33).

4.4.2 Modified Dijkstra


Bhandari's other variation, called the *Modified Dijkstra* algorithm, allows shortest paths to be found in graphs with one or more directed negative edge weights but no cycles of negative weight. These circumstances arise in the intermediate steps of the following maximum flow and shortest disjoint path pair problems.

First, a word on how the basic Dijkstra algorithm fails in these circumstances. It is perhaps tempting to think that Dijkstra fails by looping on the negative edges to apparently reduce the path cost. In fact this is not what happens. (That would actually require that a cycle on the graph exists with net negative cost.) Rather, Dijkstra can simply miss the opportunity to construct a lower cost path by taking a route that traverses a negative edge because it never rescans nodes once they are permanently labeled. It is well worth knowing and being cautioned that in practice an ordinary implementation of Dijkstra's algorithm will not crash outright or fail in any other dramatic way when presented with negative edges. More insidiously, it will seem to function in an apparently normal way, and will produce path outputs; they just won't always be the true shortest paths that are possible with negative edges. This "silent" failure is important to be aware of as nothing else would necessarily give a warning that incorrect results are coming out.


Bhandari's modification is to scan *all* neighbors of the node that most recently received its permanent label, not just nodes that have not yet been permanently labeled. If a node is re-labeled as a result of this, it is no longer considered to be permanently labeled. The extended scanning and contingency to unlabeled a node are redundant and have no net effect on the outcome in a graph with no negative edges. In a graph with negative edges, however, it avoids missing the true shortest path involving negative edges. The reason it works also diagnoses why ordinary Dijkstra fails to find such paths: a node can be permanently labeled before a longer (hop-wise) route with lower total cost is found to it, using a negative edge. [Figure 4-13](#) gives a small example in which node A is the source node.

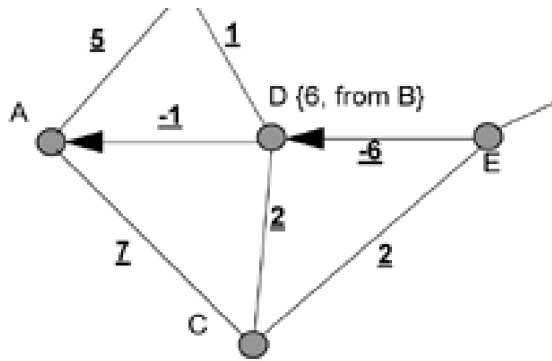
Figure 4-13. Why regular Dijkstra fails with negative edges and how Bhandari's modification overcomes this.

B {5, from A}

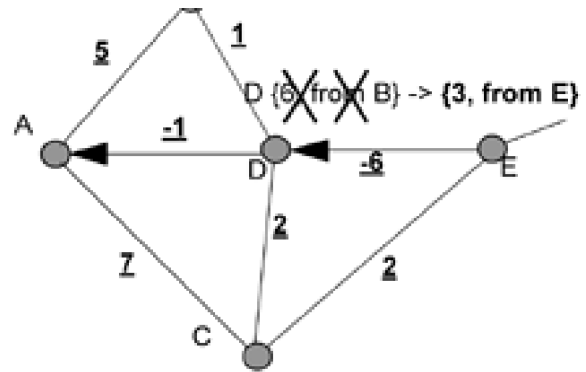


~~B {5, from A}~~ → {4, from D}





(a) Nodes B and D become permanently labeled with distances shown in the first two steps of the ordinary Dijkstra algorithm. Thereafter no shorter routes to D or B can be recognized.



(b) In the fifth and sixth iterations of modified Dijkstra nodes D and B are "rediscovered" and relabeled at new shortest distances due to the directed negative edge.

The first two steps of ordinary Dijkstra will permanently label nodes B and D, respectively, with distances 5 and 6. Some time later, node E will be visited and given the label {8, from C}. Ordinarily, node D will not then be rescanned, because it is permanently labeled. In Bhandari's method it will be rescanned, however, leading to an update for node D now that we realize that D is also reachable over the longer (hop-wise) path A-C-E-D that has the lower net cost of three. Similarly in the next step of the modified algorithm, the label at B is updated to reflect the longer but lower cost route using the negative edge E-D.

In summary, the modified algorithm is:

procedure ModifiedDijkstra(source s, target t) {

```

    CurrentNode := s
    initialize each node's Label, [P], and [D] to null
    append node s to [P] and  $D_s$  to [D]
    do while [P] does not contain all nodes{
        ModifiedScanFromNode(CurrentNode)
        find node x not in [P] with minimum  $D_x$ 
        append node x to [P]
        append  $D_x$  to [D] ([D] is the vector of  $D_i$  values for nodes in [P])
        CurrentNode := x }}

```

procedure ModifiedScanFromNode(j) {

```

    for every adjacent node j {
        if  $D_j > D_i + d_{ij}$  {
            Label@[j] := ( $D_i + d_{ij}$ )
            if j in [P] {
                remove j from [P] and  $D_j$  from [D]}
        } else { make no change }}}

```

4.5 *k*-Shortest Path Algorithms

It is easy to see that if one can find the shortest route through a graph one could also build up a succession of routes with increasing length. By removing or disqualifying the edges of the shortest path and repeating the process one would find the next-shortest path that does not use any edges of the first path, and so on. The family of successively longer paths thereby obtained is an example of one notion of *k-shortest paths* (ksp).

There is an extensive literature on slightly different notions of *k-shortest paths*, each of which relates to different problems. Topkis, for instance, is interested in a set of *k-shortest paths* through a network of trunk groups [Topk88](#). Sheir gives an algorithm for *k-shortest paths* where paths may contain loops as a means of enumerating a succession of increasing path lengths through a graph, for applications in project scheduling etc. [Shei76](#). Other variants of *k-shortest paths* derive their differences from issues such as whether they pertain to simple graphs or multigraphs and whether the *k*-successive paths are disjoint in certain ways or simply distinct. A classification system and notes on dozens of different *k-shortest path* algorithms is contained in [Gro89](#). Our purpose now is not to survey that whole domain, but to flag for readers that the simple phrase "*k-shortest paths*" is by no means a unique specification. There are, however, three particular classes of ksp algorithm that are of interest here:

1. ***k-shortest link-disjoint paths***: This is the set of *k*-successively longer paths in a multigraph where paths may share spans among the set of routes they take, but are disjoint at the link level (i.e., any one link on a span is an element of only one path at most), and the set of paths as a whole respects the finite capacity on every span of the network.
2. ***k-shortest distinct routes***: This is the set of *routes* over the spans of a network where each is merely distinct from the others and they can be ranked in increasing order of either geographical or logical (hop count) distance.
3. ***k-shortest span-disjoint routes***: This is the set of *routes* over the spans of a network where each is span-disjoint from the others and they can be ranked in increasing order of either geographical or logical (hop-count) distance.

In all cases we are interested only in non-looping paths so for simplicity this is not included in each of the criteria stated, nor do we restate this in the subsequent discussion of ksp. As an example to work with in developing these ksp concepts let us reconsider [Figure 4-11](#) as a simple graph where the edge weights given are now interpreted as distances. In this case ksp criteria 1 produces:

Shortest path: A-B-C-G-H (distance = 14)

Second path: A-D-F-H (distance = 25)

Third path: infeasible: graph is disconnected through cut {(A-B), (B-C), (D-F), (G-H)}.

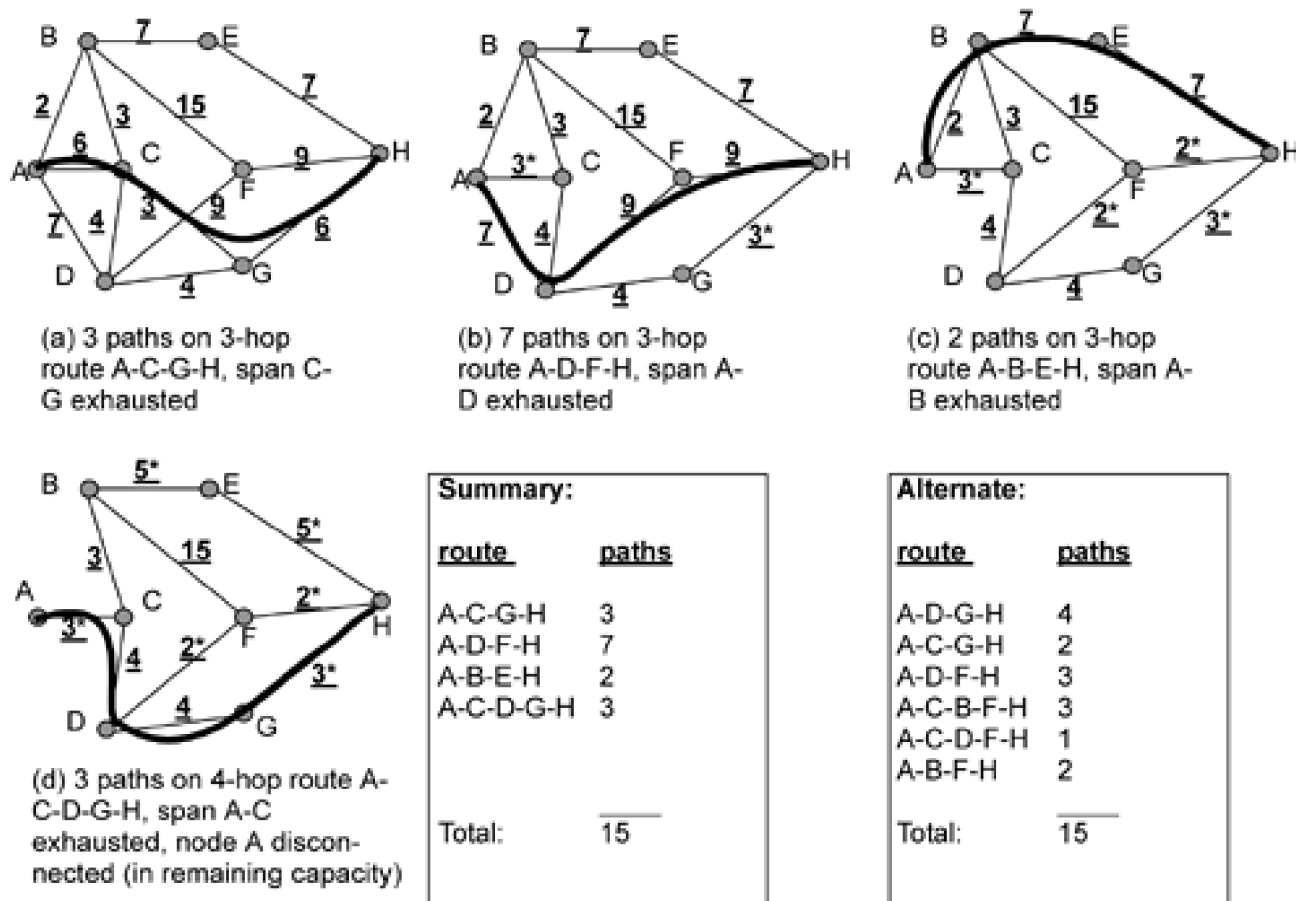
But because we are working with a simple graph in this case there is no distinction between a route and a path. If we interpret the simple graph as representing a transport network with non-zero link quantities on every span, then this result is also an instance of the *k-successive span-disjoint routes*. Not surprisingly the graph becomes disconnected quickly due to insisting on span disjointness among the succession of routes. One use for this set of routes is as a simple (but non-optimal) way of finding a span-disjoint route pair for a 1+1 APS setup or more generally to arrange a *k*-way split of the demand flow between a pair of nodes over mutually span disjoint routes.

Let us next consider the set of *k-shortest link-disjoint paths* in a multigraph (criteria 1 above). This path-set is one that can be efficiently used in mesh-restorable networks or for closely modeling the behavior of a distributed span-restoration algorithm. To provide an example of this type of ksp we now let the edge weights in [Figure 4-11](#) become the span capacities; that is, the number of unit-capacity links in parallel on each span. At the same time, for simplicity we will shift our measure of path length to that of logical hop count so that shortest and next shortest routes will be more apparent by inspection. (Note, however, that with hop length there can be more than one equal-length route existing at each increasing hop length.)

[Figure 4-14](#) steps through the process of developing this particular ksp path-set. In (a) we note that the shortest route is of length 3 and that there are actually five different 3-hop routes from A to H in the graph. In such cases one can apply an arbitrary rule or simply leave it up to the internal order of execution of a program (such as follows) to determine the order of claiming paths off of the equal length routes. Let us consider capacity on the route A-C-G-H first. On this route we find span capacities of 6, 3, 6 respectively so the feasible number of paths to follow this route is $\min(6, 3, 6) = 3$ and span C-G is effectively removed from the graph at this point since all of its capacity is used.

The updated remaining capacities on other spans are marked by *. Next, in (b) seven paths are realized over route A-D-F-H, seven being the limit set by the capacity of span A-D. The process goes on until no more paths are feasible on any route. The graph of remaining capacity is always disconnected at this stopping state. In practice it is often one of the end nodes that is simply isolated, as is the case here (node A), when all of its local egress capacity is consumed.

Figure 4-14. Illustrating the concept of k -shortest link-disjoint paths in a capacitated graph.



As mentioned, when working with hop count as the measure of length, there may not be a single unique route of each successive length, so the k sp outcome is dependent on the internal program details of choosing over equal-hop-length routes. To illustrate, or [Figure 4-14](#) we also show the outcome of another of the possible sequences that could be followed. In this case it makes no difference to the total path count, but the path details differ. A detailed study has shown in fact that although the path construction details vary with the order taken, it is quite rare for this process to yield a total number of paths that is less than the theoretical maximum number of such paths (the "max-flow" measure below) as a result of the order-dependency in the relatively low average degree that characterizes transport networks [\[DuGr94\]](#).

This dependency of the path-set details only arises from the use of logical hop count as a distance measure. With unit-hop totals as the cost measure, different routes can often have exactly the same total costs, so which is chosen then depends only on the execution order or other particulars of the program implementation. In practice, if span distances are real-valued (or, say, integer but resolved to within 1 km lengths), then every path length usually has a unique distance total of its own and the "successive length" hierarchy defined by k sp becomes uniquely defined. Highly symmetric networks or contrived assignments of real-valued distances to edges can, of course, be devised as counter-examples to this—where two paths will also have the same real-valued length—but in practice which typical variations in span lengths drawn from the real-world, or if a small random value is added to each spans's distance, the k sp flow and detailed path-set construction become uniquely defined and repeatable.

We summarize with the following pseudo-code of the k -shortest link-disjoint paths algorithm. The approach is to run a single-shortest path algorithm in a simple graph abstraction of the multigraph at each stage. As long as a span in the multigraph has unused capacity remaining (edge weight not zero) then an edge is retained in the simple graph to represent the possibility of a route that traverses that span.

```

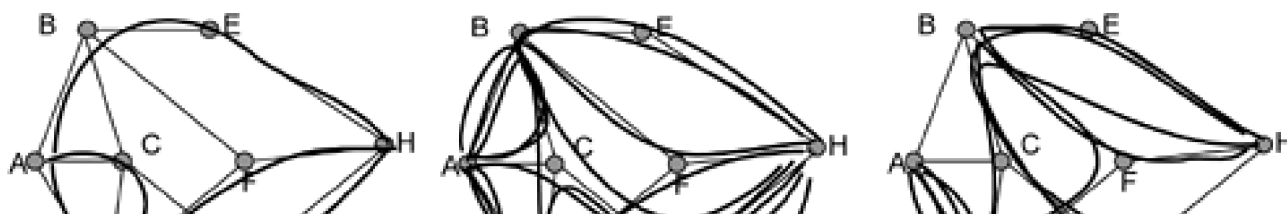
procedure ksp(capacitated graph G, source s, target t) {
  CreateSimpleGraph(G, G*) {
    for every edge i in G {
      if capacityi > 0 {
        create a corresponding single edge in G*
      }
    }
  }
  k := 0
  repeat {
    k := k + 1
    route(k) := ShortestPath(G*, s, t)
    no.paths(k) := min(set of capacities on route(k))
    for every span i in route(k) {
      capacityi := capacityi - no.paths(k)
      if capacityi = 0 {
        remove i from G*
      }
    }
    output route(k), no.paths(k)
  } until route(k) cannot be found
}

```

The complexity of this procedure can be stated initially as $O(kn^2)$ where k is the number of distinct routes in the resultant path-set and n^2 arises from each call to the basic shortest path algorithm. But k is really an output, not an input to the problem. So to make a proper statement of computational complexity one needs to somehow bound k . One approach is to say k is at most the number of distinct routes between source and target. That however is an extremely loose bound as there are $O(2^S)$ distinct routes in the worst case. S is the number of spans, hence this is also the same as $O(2^N)$. If an explicit hop limit, H , is applied to limit the length of any route considered this improves the bound on k to $O(2^H)$ but this is still very high. The reason these are wildly high bounds for is they do not reflect the elimination of remaining distinct route possibilities as spans are exhausted at each iteration. A better bound for k is based on recognizing that at termination of this procedure the network is always disconnected. In fact, the process usually discovers a minimal-capacity cut-set of the graph. Since there can be at most S spans in any cut-set, it follows that k is $O(S)$: that is, that in the worst case $k=S$ iterations would be required to disconnect the graph, reaching the stopping condition. Thus the procedure above has a theoretical time complexity of $O(Sn^2)$ which is also equivalent to $O(n^3)$ because S is related to n through the network average node degree. Additional optimizations can, however, lead to an extremely fast ksp algorithm that is theoretically and experimentally $O(n \log n)$ [MaGr94].

Let us now consider ksp criterion 2 on the list above—that is; k -shortest distinct routes. First let us illustrate the set of such routes on our example network, so as to contrast this criterion with that of disjoint routes and k -shortest paths. Figure 4-15 again uses the network of Figure 4-11 and A, H as the end-nodes. In Figure 4-15(a), the complete set of (fully) disjoint routes at all lengths is shown. In (b) the set of distinct routes that exist only at hop lengths of 3 and 4 are shown. Figure 4-15(c) shows the additional distinct routes that exist at 5 and 6 hops. The main point is that there are vastly more distinct routes than there are disjoint routes. A distinct route only has to differ in one span from previously enumerated routes, whereas each disjoint route differs from others in its set on every span on its route. One approach to an algorithm for finding the set of simply distinct paths is to extend the ksp-like aspect of iteration of a single shortest route finding routine on a simple graph, with appropriate span exclusions on each entry. For example if, on the first iteration, the shortest route found is $\{S1-S2 \dots S_m\}$ then subsequent iterations would find the remaining shortest route first with S_1 excluded, then with S_2 excluded, and so on. But it can be seen then that each of the latter paths is itself then a template for a series of further exclusions and shortest route finding calls. In other words, it leads to a search tree for all distinct routes that is effectively a depth-first search (DFS) rooted on each span of the initial shortest route. As a practical matter, therefore, it is easier to obtain the k -shortest distinct routes by length sorting the outcome from an all-distinct routes routine, which is based on a general DFS, and is a separately required capability in any case. In other words, rather than develop an extension to ksp to find the k -shortest distinct routes, we will move on to a DFS for all distinct routes. With only a length sort to be added this will also provide the k -shortest distinct routes.

Figure 4-15. k -shortest disjoint and distinct routes (up to six hops) in the example network.





(a) There are two *disjoint* routes of 3 hops and 1 more found at 4 hops



(b) There are nine *distinct* routes possible at 3 and 4 hops



(c) and four more *distinct* routes at 5 and 6 hops

4.5.1 All Distinct Routes

The set of *all distinct routes* on a graph up to a certain hop (or distance) limit H , is needed in certain problem formulations for mesh network capacity design. As with ksp, our interest is only in non-looping routes. Finding all distinct routes can be a challenging task, however, simply because there are $O(2^n)$ such routes possible between node pairs in a graph of n nodes (and $Q(2^n)$ in a fully connected mesh on n

nodes). More specifically, if we have a network of S spans and are using a hop limit $H < S$ we could expect a bound of $O\left(\binom{S}{H}\right)$. In many planning problems, however, the set of such routes needs to be computed only once for a given topology and may then be stored and used as a sort of database for different optimization studies that need various subsets or filtered selections of routes from the set of all distinct routes for their formulation.

The algorithm for all-distinct routes (equivalently all distinct paths in a simple graph) is a type of depth-first search (DFS) algorithm. This algorithm thus also serves as an example to introduce DFS as a basic method underlying other algorithms to follow. In DFS in general, one starts at a source or root node that is problem-specific. Here, it is one of the end nodes of the node pair between which all distinct paths are desired. From this node one takes an excursion out on any span incident at the source, marking that span as having been visited. At the node thus reached, this is repeated. At each node the rule is to take an unmarked span, mark it, and head further down to a next node if possible. For all distinct routes, whenever the target node is discovered, the process records a route and backs up to the penultimate node in the route. If all spans are marked at that node, it backs up again. At the first such node found during this backtracking, where there is an unmarked span, the DFS search branches down that span and continues as above. Any time the search depth reaches H hops (or an accumulated distance limit) without discovery of the source, or if a loop is discovered in the route, the process also backs up. A potential loop is discovered when the unmarked span currently branched upon leads to a node that is already a parent of the node from which branching just occurred. In this case the span is marked and the branch is not taken. An important detail is that as the process backs up from any node, all spans at the node that it is leaving, except the one to the parent to which it is returning, are *unmarked* upon departure from the node. This is necessary so that spans that are part of an early route discovery are left free to be incorporated in distinct routes later discovered from different approaches. But the current search path will not be explored any further in that direction. Thus, when a whole region of the search tree is completed, all spans are left unmarked except the one incident on the source from this region. A pseudo-code version of the algorithm follows.

procedure AllDistinctRoutes(graph G, source node, target node) {

```

node := source
depth := 0
[route] := [null]
Repeat {
  all booleans := false
  StepAhead(done_at_this_node)
  if not done_at_this_node {
    if node in [route] {
      loop := true }
    if node = target {
      foundroute := true }
    if depth = H {
      hoplimit := true }}
  if done_this_node or loop or hoplimit {
    Backup
  } else if foundroute {
    record_route
    Backup
  }
} Until node = source and unmarked_spans(source) = null}

```

```

procedure StepAhead(done_at_this_node) {
  done_at_this_node := false
  span := next([unmarked_spans(node)])
  if span = null {
    done_at_this_node := true
    return
  } else {
    node := neighbor(node, span)
    mark(span)
    depth := depth + 1
    [route] := [route] + [node]
  }
  return
}

```

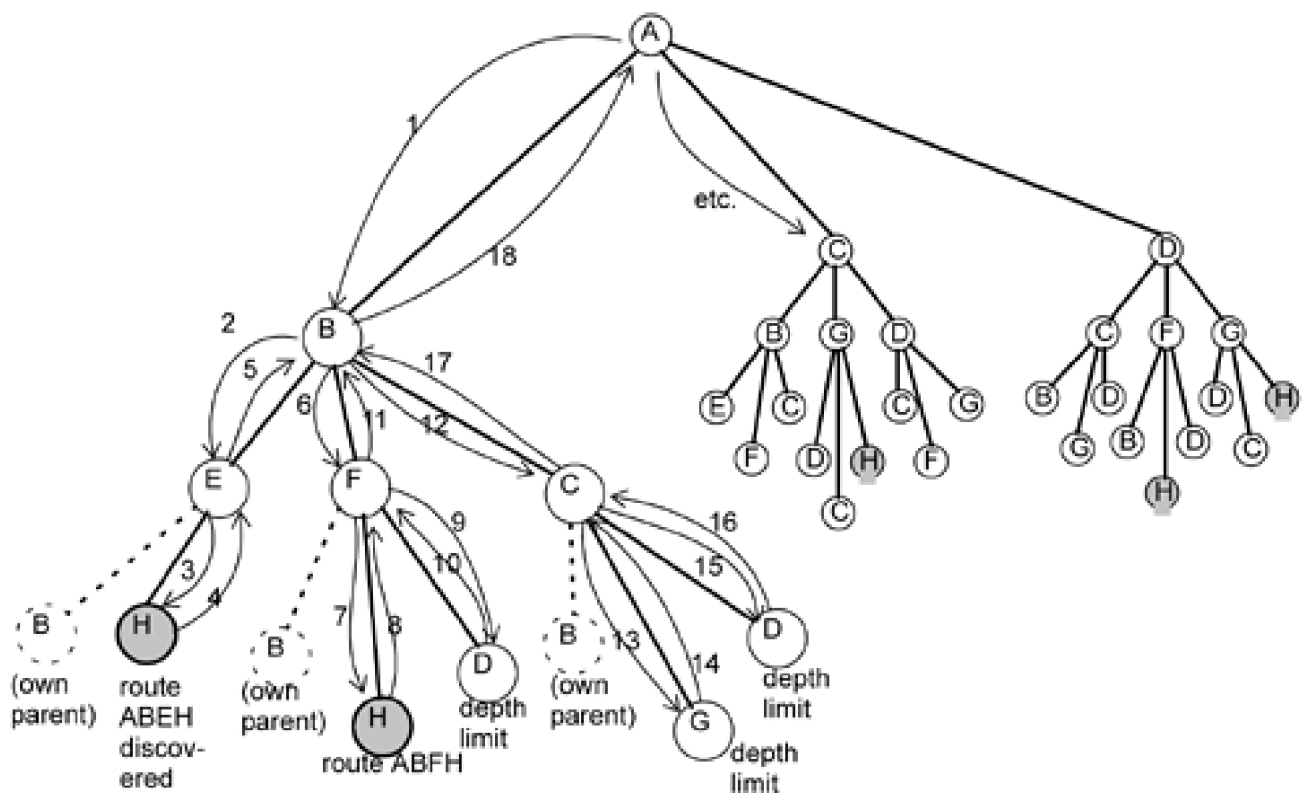
```

procedure Backup {
  find_parent([route])
  unmark([spans(node)] - [(node,parent)])
  [route] := [route] - [(node,parent)]
  depth := depth - 1
  node := parent
  return
}

```

Figure 4-16 illustrates the DFS operation on the example network of Figure 4-11 (Section 4.3.2) for a hop limit of $H = 3$. The graph structure of Figure 4-11 is reflected in the DFS search tree structure in that each node (other than the source node) appears at each level sub-tended by the other nodes reachable from it. The blown up region with numbered arrows shows the sequence of "StepAhead" and "Backup" operations that explore the branch from the source A having first span (A-B).

Figure 4-16. Operation of the depth-first search to find all distinct routes.



4.6 Maximum Flow: Concept and Algorithm

Let us now consider finding the maximum flow between two nodes in a capacitated network. First we will look at the important *max-flow min-cut* theorem and then an algorithm to find the maximum flow quantity, and the role of *oksp* in providing path sets that closely approximate the max-flow quantities.

4.6.1 The Min-Cut Max-Flow Theorem

A *cut* in a graph is a set of edges which, if removed, disconnect the graph. Alternately a cut in graph G can be thought of as specifying a partition of the vertices of the graph into set P and its complement \bar{P} . The set of edges (v, w) with $v \in P$ and $w \in \bar{P}$ is the corresponding cut (or cut-set) and the *capacity of the cut*, $C(P, \bar{P})$ is the sum of the capacities of each edge in the cut.

Equation 4.6

$$C(P, \bar{P}) = \sum_{v \in P} \sum_{w \in \bar{P}} c_{v, w}$$

The *min-cut max-flow theorem* states that the largest flow that can be sent between two nodes cannot exceed the capacity of the minimum cut between the two nodes. A bit more formally, if $F_{i,j}$ is a flow (over possible many routes) between nodes i, j , then

Equation 4.7

$$F_{i,j} \leq C(P, \bar{P}) \quad \forall i, j | i \in P \cap j \in \bar{P}$$

and, specifically, if $F_{i,j}$ is the maximum feasible flow, then there is a cut which has $C(P, \bar{P}) = F_{i,j}$ and is the cut which has minimum capacity over all cuts that separate i, j . In other words:

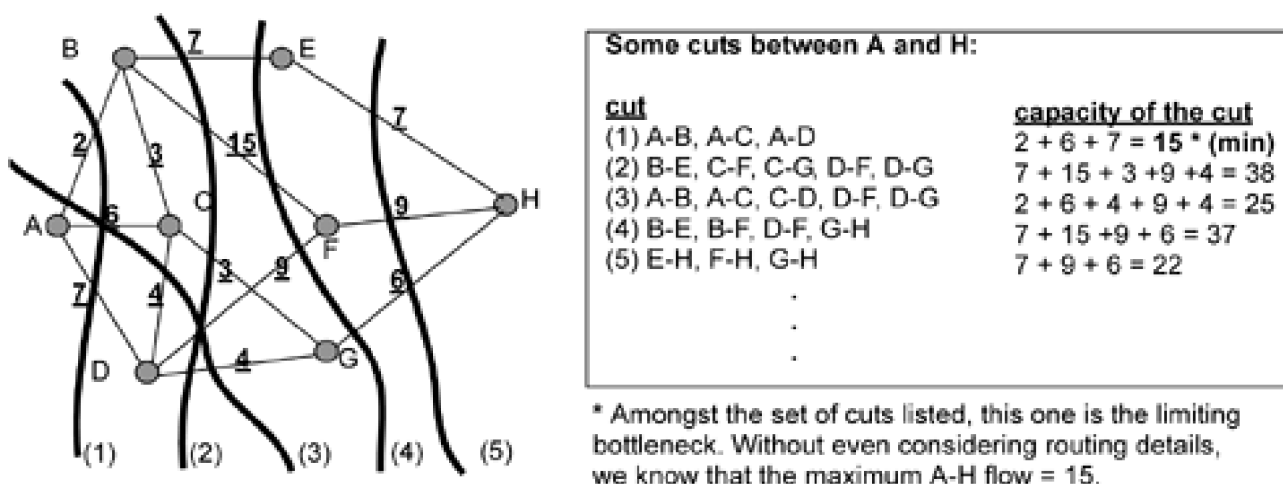
Equation 4.8

$$F_{i,j} \Big|_{\max} = C(P, \bar{P}) \Big|_{\min}$$

The min-cut max-flow theorem is easily understood if one thinks about bottlenecking arguments. If we think of it as a kind of *bottleneck theorem* it states equivalently that between any two nodes of a network, some set of edges over which flow may be sent will be saturated and act like a bottleneck, limiting the maximum flow. Let us again use the graph of [Figure 4-11](#) for an example. In [Figure 4-17](#) a sample of five cuts are shown which separate source and target nodes A, H. There are many other cuts that also separate A, H but the point that this sample illustrates is the same for the set of all such cuts. The point is that among the capacities of cuts 1 through 5, which are 15, 38, 25,

37 and 22, respectively, the minimum is 15. Thus, regardless of any other the details, there can be *no* construction of a path-set between A-H in this graph that has a capacity of *more than* 15. This is not dependent on routing details or cleverness of the routing algorithm: the max-flow can't exceed the min-cut. In the example it is evident that the edges adjacent to node A themselves only support a total of 15 flow units. All the other edges in the graph could be given 100 or 1000 units of capacity but it would make no difference to the maximum feasible flow between A, H (or H, A) because of the 15-unit capacity bottleneck defined by the {A-B, A-C, A-D} cut. Note that in finding the maximum flow based on min-cut considerations we do not obtain a routing solution that realizes the max-flow, we only learn the max-flow quantity for which some feasible path construction must exist.

Figure 4-17. Max-flow min-cut theorem says there is always a bottleneck somewhere that limits the flow between nodes. The bottleneck is the min-cut.



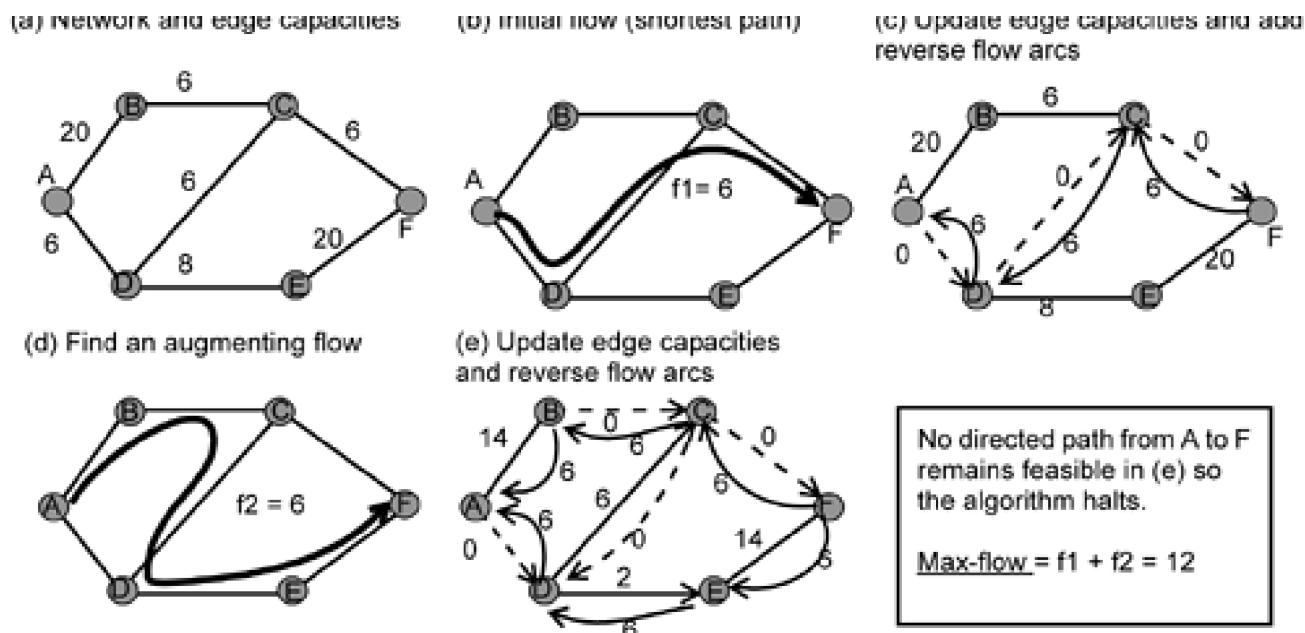
4.6.2 Maximum Flow Algorithm

The max-flow min-cut theorem itself gives us, at least in principle, a way to compute the maximum feasible flow between nodes: find all cuts of the graph (that partition the nodes of interest), compute the capacity of each cut, and take the smallest value. The trouble with this approach is that the number of cuts grows exponentially with graph size. For a fully connected graph there are $(2^n - 2)$ possible cuts. There is, however, a polynomial time algorithm ($O(n^3)$) to find the maximum flow called the *flow augmentation algorithm*. It finds the maximum flow quantity and in doing so identifies a minimum-weight cut.

The procedure is to find any initial feasible flow from source to target, say the flow over the shortest route, and then *augment* or *improve* the flow iteratively until no further improvements are possible. The key to finding the flow augmenting opportunities is the adoption of a directed orientation to edges of the graph as the algorithm proceeds. Basically, as we route an initial flow over an edge we give it the direction that is implied by the initial flow and create an equal flow on an imaginary parallel edge with *opposite* direction. The imaginary reverse direction flows in effect provide a means to represent that we may be willing to "take back" some of the initial flow assignment on certain edges, if the overall flow can be improved. These adjustments effectively sense and saturate all edges in the min-cut. The process is most easily seen by stepping through an example.

In choosing an example, however, [Figure 4-11](#) will no longer serve us well because the min-cut of that network is indeed just the one identified around node A with capacity of 15, which is also the capacity found by *k*-shortest paths. A more useful example to portray the flow augmentation technique is one where the min-cut is located out in the "body" of the network, and where the ksp procedure would not necessarily match the max-flow result. Hence let us work with the network in [Figure 4-18](#) to find the max-flow from A-F. All edges have unit length except B-C and D-E, which have an arbitrarily high length. The capacities are as shown in [Figure 4-18\(a\)](#). The topology is contrived so that the ksp result and max-flow will not be identical to support later discussion of the *trap topology* concept.

Figure 4-18. Finding the max-flow quantity by the flow augmentation algorithm.



Because edges B-C, D-E are "long" the shortest path is A-D-C-F with a flow of 6. And yet if we route such a flow we will find that no further flow between A-F is possible, because the graph of remaining capacity is disconnected. At the same time, however, we can directly identify the min-cut {B-C, A-D} with a capacity of 12. Thus, we start out seeing that ksp is quite different than max-flow in this example and that at completion of the max-flow procedure we should expect a total flow solution of 12.

Step 1. Let us say a shortest path algorithm is used to find the initial flow of six. In [Figure 4-18\(b\)](#) we represent this initial flow, f_1 .

Step 2. In (c) we draw the updated edge capacities (i.e., the remaining capacities on the edges after taking the initial flow) and we create reverse-direction arcs of the same capacity as that in the forward direction associated with each initial flow over each edge. Arcs with zero capacity are drawn as dashed lines.

Step 3. Next we search for an augmenting flow in the graph of (c), respecting the directionality of arcs now present. There is only one route feasible: A-B-C-D-E-F with a directed flow, f_2 , of six.

Step 4. We again update the remaining capacities, treating the artificial arcs the same as any other, and adding any new reverse arcs associated with the augmenting flow. The result is (e).

Step 5. In (e) there are no more paths feasible, so we halt. The max-flow result is the sum of the initial flow f_1 plus the one augmenting flow f_2 found. The corresponding min-cut is also identifiable at this stage among the set of forward arcs left with zero-capacity. Partition {A,B} {C,F,E,D} is identified as the min cut (corresponding to cutset A-D, B-C). Other cutsets are contained within set of forward arcs having zero capacity upon termination, but all of these have higher cut magnitudes in the graph of actual capacities. For example, C-F, D-C, A-D is also a cutset comprised of zero-capacity edges in [Figure 4-18\(e\)](#), but those edges have total weight of 18 in [Figure 4-18\(a\)](#).

A key point for understanding the max-flow algorithm is that at the end of the example in [Figure 4-18](#), f_1 and f_2 lead to identification of the max-flow quantity of 12, but the routes of f_1 and f_2 (in (b) and (d) above), are *not* the actual routes of a path-set that realizes a flow of 12 in this graph. One reason they cannot be is the edge C-D has capacity 6, not 12. In fact edge C-D is not used in a path construction to realize max-flow in this network. If one path takes it, the other is blocked out. The max flow algorithm in effect realizes this aspect of the graph through its reverse direction flow arcs. [Figure 4-19](#) shows the actual construction that yields max flow in [Figure 4-18](#). Pseudo-code for the flow-augmenting max-flow algorithm is as follows:

[\[View full width\]](#)

procedure MaximumFlow(graph G, source node, target node) {

```
i := 0
tot_flow := 0
find_shortest_path(G, source, target, [route])
flow := min(capacity of spans in [route])
repeat {
  split_new_flow_edges(flow, [route])
  find_shortest_path(source, target)
  update_capacities(flow, [route])
  if [route] not = [null] {
    flow := min(capacity of spans in [route])
    tot_flow := tot_flow + flow}
} until [route] = [null] }
```

procedure find_shortest_path(source node, target node){

This can be an implementation of Dijkstra's algorithm with the added consideration of
➔ each edge being treated as a directed pair of arcs, each with its own capacity.}

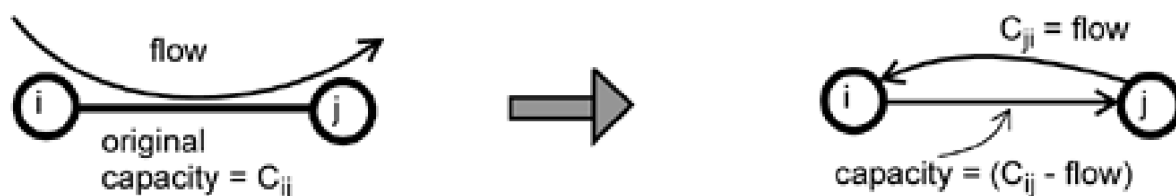
procedure split_new_flow_edges(flow, [route]) {

For any edge that is not yet "split" into a pair of anti-directional arcs that is
➔ traversed by an initial or augmenting flow, convert the original edge into one forward arc
➔ with capacity = (original capacity - flow value) and a reverse direction arc with capacity
➔ equal to the flow value.}

procedure update_capacities(flow, [route]) {

For all forward and reverse-direction arc pairs, decrease their capacity by the amount
➔ of any initial or augmenting flow that crosses them. }

Figure 4-19. In the max-flow algorithm edges are split into arcs representing the original flow and the prospect of reversing that initial flow decision in a later flow-augmenting stage.



(a) Intact edge gets a flow routed over it

(b) Max-flow algorithm splits the edge into anti-directional arcs

Note that the flow augmenting steps need not only involve flows over the reverse direction arcs. For instance, if the capacity of edge A-D was eight (instead of six), then an additional iteration would find additional "conventional" flow of two on route A-D-E-F for a total flow of 14. In addition note that edges are not split into a pair of opposing directional edges until such time that a flow is actually routed over them because it is only at that time that a "forward" sense is defined for the edge. This is illustrated in [Figure 4-19](#).

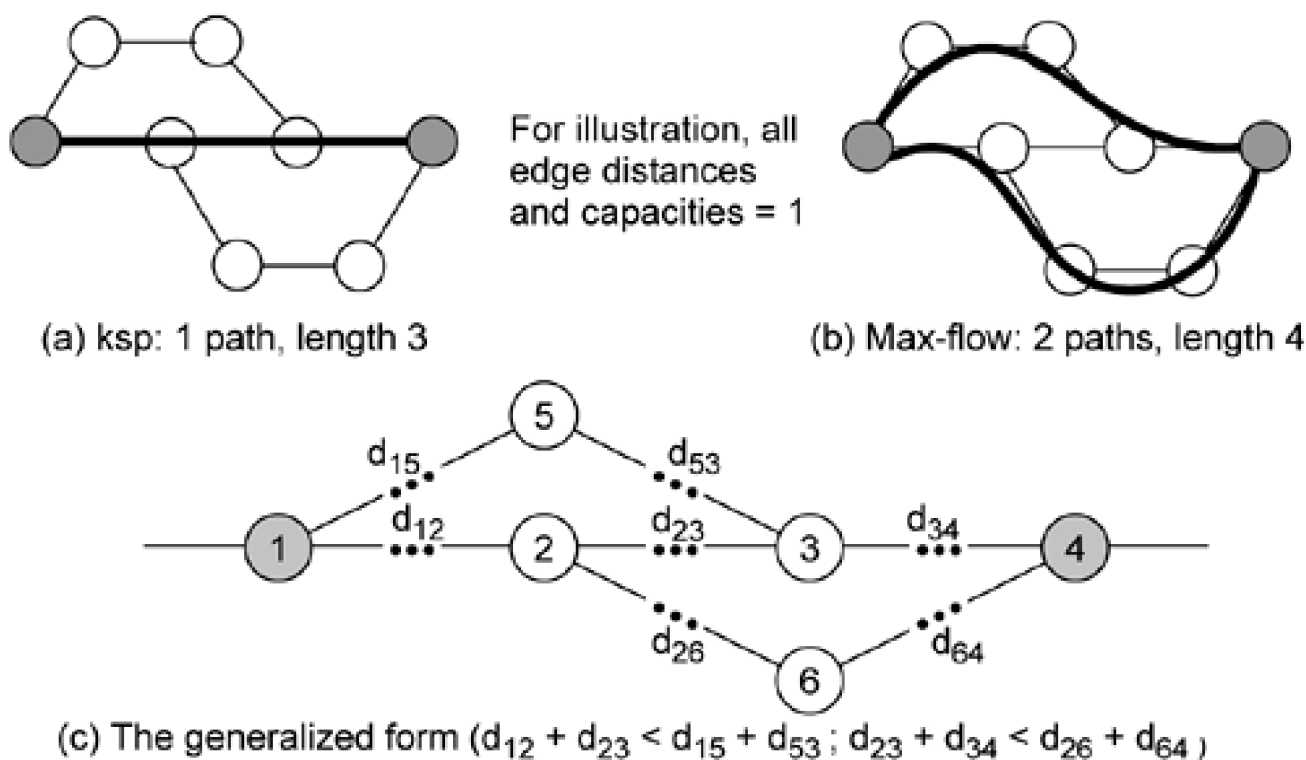
A way to understand the role of the reverse direction arcs is that they represent the willingness to "give back" some of the initially committed capacity on an edge, if it supports a later flow augmentation. To see this in the example we can look at the "naive" shortest path flow and the path-set that actually corresponds to the max-flow value found, which are shown in [Figure 4-20\(a\)](#) and [\(c\)](#), respectively. The comparison makes it clear how the initial routing is revised by effectively cancelling or rescinding the decision to route flow over the diagonal edge (b), which is the key to realizing the max-flow value in this case. It also makes the point that the augmenting flows are only intermediate vehicles to discover the max-flow quantity. They do not serve directly as the routes over which a feasible set of paths equal to the max-flow can be constructed. The point of the algorithm is to find the max-flow *value*, not the max-flow path-set. The set of possibly infeasible routes that nonetheless identify the max-flow quantity can, however, guide construction of the actual max-flow path-set by excluding all edges that have no flow on them in either direction as well as any edge that has been split and (at the end of the procedure) has forward and reverse arcs of equal (i.e., cancelling) flow. A set of actual paths that follows routing on this template set of edges and conforms to the capacity of those edges is then a feasible max-flow path-set construction.

4.6.3 The Trap Topology

In the example just shown, we can see that ksp (with $k=2$) and max-flow are not equivalent. ksp will, by "greedily" first taking the shortest route for 6 paths, make it impossible for itself to discover any further paths. In contrast, the max-flow solution requires using two routes, neither of which is the shortest route. In other words, in the topology of [Figure 4-18](#) ksp is trapped by its "greedy" nature.^[1] In fact, in all cases where max-flow and ksp differ, it is always due in one form or another to a manifestation of a *trap topology*, which is a canonical form of subnetwork structure in which taking a path on the shortest route(s) reduces the total number of paths that can be found. [Figure 4-21](#) illustrates the trap topology in its minimal and higher order general form. There are at most two edge-disjoint paths between nodes 1 and 4. One path is 1-5-3-4, and the other is 1-2-6-4. However, if we choose the path 1-2-3-4, which may be shorter and hence chosen by ksp, then only one edge-disjoint path from 1 to 4 can be created. This construction serves as an existence proof of the difference between successive shortest path buildup of a total flow and true max-flow. It also gives insight as to what happens: in the trap topology, taking the capacity on the shortest route first, as k -shortest paths is defined, foregoes the opportunity to create more total flow on two longer routes. This is the sense in which ksp is a greedy algorithm. Note that as expected the capacity of the min-cut in [Figure 4-21\(a\)](#) or [\(b\)](#) is two, equal to the maximum flow capacity. The question of practical interest is how often does a ksp path-set support less than the maximum flow?

^[1] In this context, "greedy" is a technical term in computing science. It refers to an algorithm that operates by maximizing some goal on a particular subproblem. A complete solution is not approached globally, rather it is developed as the accumulation of the greedily- (or myopically-) solved subproblems.

Figure 4-21. Canonical form of the trap topology—where ksp and max-flow may differ.



4.6.4 k -Shortest Paths as an Approximation to Maximum Flow

This question just posed was given theoretical and experimental treatment in [\[DuGr94\]](#). The study compared max-flow and k -shortest paths restoration quantities in over 300 randomly generated 50-node transport network models. The test networks varied systematically in average nodal degree (network connectivity) and in average span locality. Locality is a measure defined in [\[DuGr94\]](#) to reflect the degree

of non-planarity of the network graph. These parametric variations ensure that the sample space of test networks generously covered the range of topological characteristics observed in real transport networks. Broadly, real networks tend to have $2 < \bar{d} < 5$ and $1 < L < 1.5$ where \bar{d} is the average number of spans per node and L is a propensity for out-of-the-plane spans (i.e., non-planar edges). $L = \infty$ means any node is as likely to have a span to another node at any distance away and $L = 1$ implies spans exist only between nodes that are geographically adjacent in the grid space on which the trial networks were synthesized.

The most important finding was that in trials of over 2,600 span cuts in 300 different networks, the total number of paths found by max-flow was 14197 and by ksp, 14193, for a difference of 4 paths in > 14,000 possibilities. In other words, ksp found over 99.97% of the ideal max-flow restoration capacities. This difference is trivial in practice in terms of the restoration ratios that would be realized in real networks.

[DuGr94] also gives some insight and theoretical substantiation for why the difference is so surprisingly small, given the ease of constructing the basic trap topology (Figure 4-21). Inspection of cases where max-flow and ksp differed in the 300 networks did show that it was indeed always due to some manifestation of the basic topology but that the trap requires such a precise relationship between seven different spans, while not also being "short-circuited" by surrounding path options which defeat the trap. In effect the trap subgraph has to occur almost in isolation between nodes 1, 4 in Figure 4-21 while nonetheless being part of a larger essentially randomly connected network. In addition, even when an isolated trap topology subnetwork is present without such short-circuiting effects, the trap can also fail to have its deleterious effect for all paths due to random differences in available capacity on the spans of the path. This is an effect which is not appreciated when we only consider the trap as a simple graph. Briefly, the simplest example of this effect is to consider the path segment d_{23} in Figure 4-21(c). Even if part of the shortest path, if it contains a span with less capacity than the other path segments not on the shortest path in the trap, a certain amount of the total potential flow will be rejected from the shortest path, thus unwittingly avoiding the trap. [DuGr94] discusses these points further and includes some analytical models that also confirm that at least in graphs that are at all similar to transport networks, it is reasonable to expect that ksp is an extremely good approximation to max-flow. One practical advantage of this is that restoration preplans can realistically be re-computed continually with a ksp algorithm which can be $O(n \log n)$ rather than with a max-flow algorithm which is at least $O(n^2 \log n)$. It also gives a reassurance that any distributed adaptive algorithm for span-restoration that is equivalent to ksp in its performance will make essentially perfect use of any available restoration capacity within a network.

[[Team LiB](#)]

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

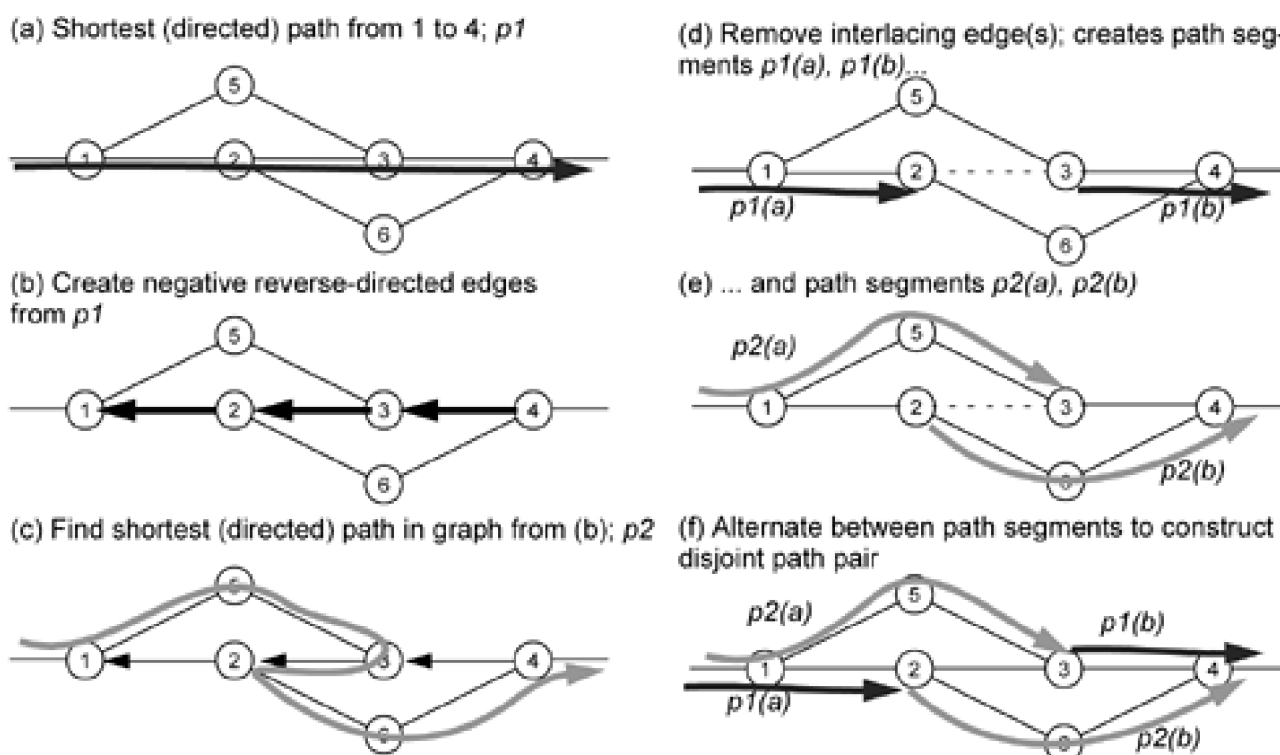
4.7 Shortest Disjoint Path Pair

A slight variation on the max-flow procedure can find the min-cost or shortest pair of edge-disjoint paths between nodes. Such path pairs are important in some schemes for protection where a working or primary path and a disjoint backup path are planned. The application to 1+1 APS DP is direct. The algorithm can also be applied to SBPP arrangements, or extended in various ways to recognize that with sharing of spare capacity on the backup route, the costs of using those spans for the backup may be discounted relative to the primary path. In later sections on optimum network flow problems with Integer Programming, we will see that the min-cost disjoint path pair can also be solved as a particular form of min-cost network flow problem.

A simple heuristic approach to find disjoint path pairs is to take the shortest path followed by the second shortest path not using any edges of the first—in other words, the k -shortest disjoint routes with $k=2$. In practice this often does yield a min-cost disjoint path pair, but in general optimality, even feasibility, is not guaranteed. However, the following algorithm attributable to Bhandari [Bhan99] is not much more complex and is guaranteed to find the min-cost (equally weighted) disjoint path pair. In practice, anything to address this problem is often generically referred to as *Suurballe's algorithm*, after Suurballe who first worked on the problem [Surb74], [SuTa84]. But often this is only a reference to the minimum cost disjoint path pair problem, not to Suurballe's specific solution—which involves a rather complex network transformation to avoid the reliance on negative edge costs. In contrast, Bhandari's algorithm is quite a bit simpler than the actual Suurballe algorithm, through use of his modified Dijkstra algorithm (Section 4.4) to handle negative edge weights. Thus, in fairness to Bhandari we should more precisely speak of *Bhandari's algorithm* being used for the *Suurballe problem*.

To illustrate the shortest disjoint path pair algorithm, it makes sense to use the trap topology directly, as we know this provides us with a case where the min-cost disjoint path pair differs from ksp with $k=2$. Bhandari's algorithm is illustrated for that case in Figure 4-22.

Figure 4-22. Bhandari's algorithm for finding the shortest disjoint path pair.



With reference to Figure 4-22, the steps of the algorithm are:

1. Take the shortest path between the source, s and target, t . Denote this $p1$. This is in Figure 4-22(a).
2. Define the direction of each edge traversed in $p1$ from s toward t as positive.

3. Remove all directed edges on the shortest path p_1 and replace them with reverse direction edges with minus one times the original edge cost—[Figure 4-22\(b\)](#).
4. Find the least-cost path from s to t in the graph of step 3 using the modified Dijkstra algorithm. Denote this path p_2 —[Figure 4-22\(c\)](#).
5. Remove any edge of the original graph traversed by both p_1 and p_2 . These are called *interlacing* edges. Identify all path segments created the edge removal from paths p_1 and p_2 .—[Figure 4-22\(d\)](#).

Bhandari's description in [\[Bhan99\]](#), (p. 65) states at this point that the remaining segments of p_1 and p_2 can be regrouped to obtain the shortest disjoint path pair but details are omitted. A general rule to assemble a minimum-cost disjoint path pair from the fragments of p_1 and p_2 is therefore offered as follows:

6. Follow p_1 , in the direction from source to destination, labeling each path segment terminated by an interleaving edge as $p_1(a)$, $p_1(b)$...etc., and do the same for p_2 , giving path segments $p_2(a)$, $p_2(b)$...and so on. This is shown in [Figure 4-22\(d\)](#) and [\(e\)](#).
7. In the example, one path of the shortest disjoint path pair is then path: $\{p_1(a), p_2(b)\}$ and the other is $\{p_2(a), p_1(b)\}$ shown in [Figure 4-22\(f\)](#).

In general, if there is more than one interlaced edge (such as 2-3 in this example), then each such edge creates another set of path segments which are stitched together in the same alternating manner of association of the path segments, to form the final two disjoint paths. Traveling from s toward t , the node on each proximate side of interlacing edges, becomes a switchover point to go from a fragment of p_1 (or p_2) over to a segment of p_2 (or p_1), respectively. In other words, one follows a segment until it ends, then switches to the segment from the other initial path leaving the same node. This is repeated until t is reached on one path, then on the other, to construct a minimum-cost disjoint path pair.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

4.8 Finding Biconnected Components of a Graph

In [Section 4.1.2](#) and [Chapter 3](#) we appreciated that a survivable network must be based on a two-connected or preferably a biconnected graph. A biconnected graph has no cut-nodes (or single-edge cuts) and by implication has a min-cut between any O-D node pair that contains two or more edges. While graphs with these properties are easily recognized by human inspection, algorithms for topology planning (such as those in [Chapter 9](#)) require mechanized processes for enumerating graph options and disqualifying graphs that are not suitable from a biconnectedness standpoint. Typical uses for the biconnectivity algorithm that follows are:

1. A random graph, or a set of alterations to an existing graph is automatically generated. Is the resulting graph biconnected?
2. If various nodes or edges are removed from a network within some larger search strategy, is the remaining graph still a single biconnected component? If not, what subgraphs or biconnected components remain?

The following procedure is based on a a depth-first traversal of the graph, G , followed by a backtracking phase [\[BaGe00\]](#) [\[ThSw92\]](#) (p.354). It is during backtracking that the process can infer—from information laid down during forward traversal—that there is no "exit" from a subgraph, G' , other than through the same node z , through which the subgraph was entered in the DFS. In this case z is identified as a cut-node and G' as a biconnected component of G . The algorithm itself finds the maximal biconnected components of a graph. Maximal means that each such *bicomponent* found is not contained within any larger biconnected subgraph. The same algorithm can be used as a test of overall biconnectivity by checking if a single bicomponent is returned which contains all nodes of the original graph. Tests for biconnectivity and two-connectivity use the same principle. The difference is that the biconnectivity test checks for a cut-node, while the two-connectivity test checks for a cut edge. Let us give an overview, then an example.

Overview of Algorithm

In a DFS traversal in general, each edge from a node is explored and marked when first traversed. Each time a new node is arrived at which has unmarked edges, the traversal continues to explore downwards or in "depth first" over an unmarked edge. When all edges from a node have been explored the process backs up to a node it has been at before and branches out on a new edge from that node. Some definitions: the *root* node, r , is the starting node for the DFS search tree. A node v is an *ancestor* of a node w in the search tree if $v \neq w$ and v is on the path from the root to w . The closest ancestor of a node w is the *parent* of w . If node v is an ancestor of node w , then edge (v, w) is also called a *back edge*.

In the DFS traversal for this algorithm, variables *depth* and *back* are operated on for each node encountered. $Depth(v)$ records the basic iteration number of the DFS search at which node v is first encountered. It serves as an arbitrary but unique number that becomes a tag each node in the order they are *first* visited in the traversal. Through the operations on $back(v)$, the *back* value in effect records the lowest yet-known number of hops to return *from the biconnected component currently being explored* (not from the node itself) to the root node. When the algorithm completes, the nodes of each distinct biconnected component all share a common back value which identifies that entire bicomponent.

When nodes are first encountered, they are assigned $back(v) = depth(v)$, since the only known route back to r follows the initial DFS sequence of exploration to get to this node. Later in the traversal of unvisited edges, the same node may be arrived at again, through a *back* edge. If node w has already been visited and is revisited on a traversal of an edge from node v , then if w is an ancestor of v (i.e., it is already found in the sequence of forward edges traversed by the DFS), then we say that a *back edge* (v, w) has been discovered. The significance of a back edge is that it represents discovery of an additional edge back to one or more of the prior nodes already traversed. This implies a certain local extent of biconnectivity between the current node and other nodes in the DFS sequence, at least back to the node w . Thus, when we step to a node v and discover a back edge (v, w) incident on the node, we set $back(v)$ to $\min\{depth(w), back(v)\}$. The back value of a node is "pulled up the tree" like this *only* when a back edge is discovered on a scan from that node (or later in backtracking) but not when the node is visited on a normal forward edge. The result is that later, when backtracking from v to w if a node is encountered at which $back(v) \geq depth(w)$, and if w is not the root node, then we have just discovered that a bicomponent exists which is only accessible through node w in this network. In other words, w is a cut-node.

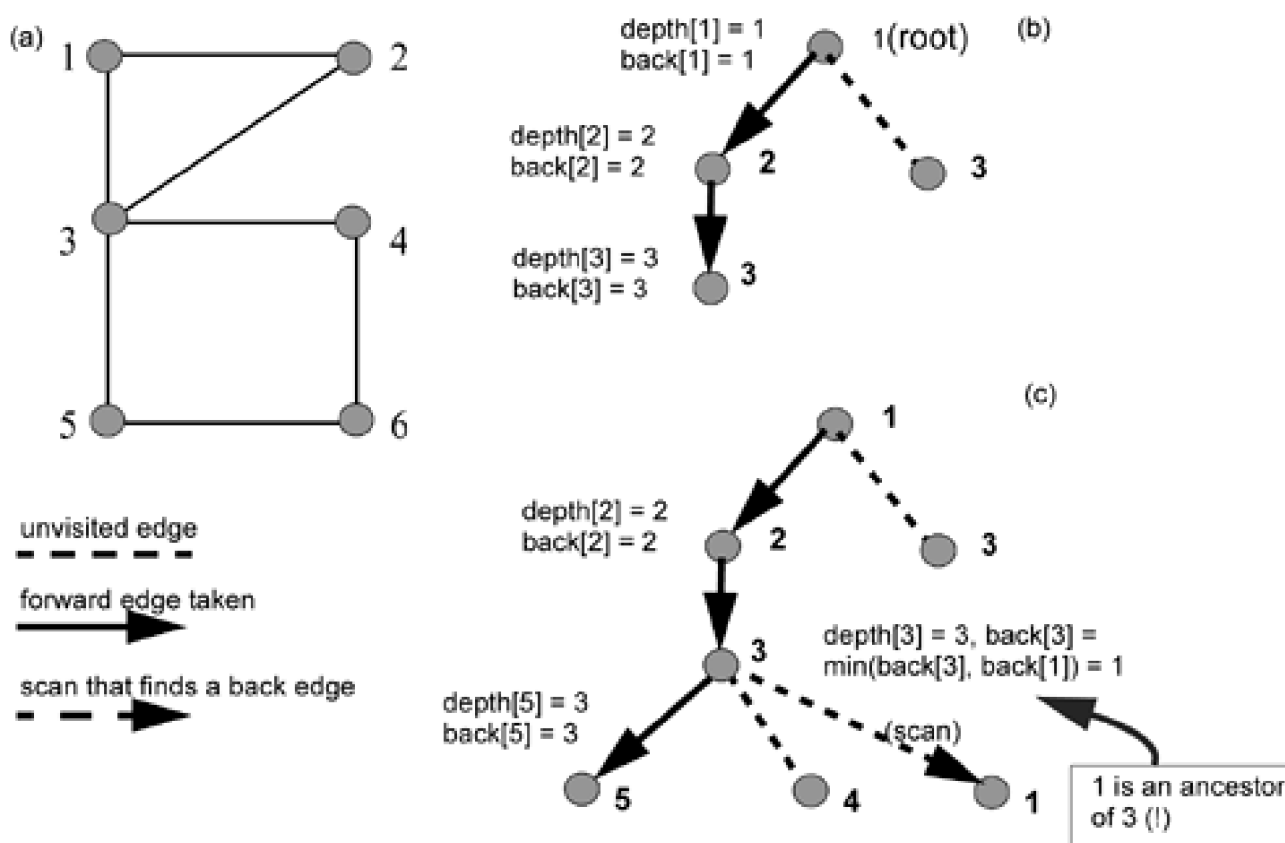
If we assume that a cut-node does exist, then as the traversal continues to explore all edges in what is currently (although unknowingly) a biconnected subgraph, the effect is that all nodes in the bicomponent wind up being assigned the same $back()$ value as the depth value of

the cut-node through which the subnetwork was entered. If the node in question was not a cut-node, the same traversal and backtrack process would have "pulled up" its $back()$ value too, by discovery of other back edges of the biconnected component that contains it.

Example

Let us now illustrate the algorithm with the network shown in [Figure 4-23](#) which has a single cut-node, node 3 and two obvious biconnected components. Let node 1 be the root node. We assign $depth[1] = 1$, $back[1] = 1$. Its incident edges are (1,2) and (1,3) and neither has been traversed. Let us set out on (1,2). For node 2, we assign $depth[2] = 2$, $back[2] = 2$. Now edge (1, 2) has been traversed and since the basic procedure is a depth first traversal, we go on to node 3 over edge (2,3) (rather than following edge (1,2) back to node 1 at this stage). Node 3 gets $depth[3] = 3$, $back[3] = 3$. At this stage the search tree is as shown in [Figure 4-23\(a\)](#).

Figure 4-23. Example network (a) and first two DFS tree states (b), (c) illustrating the algorithm to test for biconnectivity. (Example adapted from [\[Ezem03\]](#).)



Now the incident edges to node 3 are (1,3), (2, 3), (3, 5) and (3, 4). A scan from node 3 indicates (2,3) has been traversed, (3,5) and (3,4) are untraversed or forward edges and edge (1,3) has not been traversed but that it leads back to node 1, which is an ancestor of node 3. Therefore edge (1,3) is a *back edge* for node 3. We therefore "pull up" the $back()$ value of node 3 to the $depth()$ value of node 1, i.e.,

$back[3] = \min(depth[1], back[3]) = 1$. (It is not relevant to this step that node 1 happens also to be the root node.^[2] In general, when a node is visited on a forward edge, it is *scanned* to see if any adjacent nodes are its ancestors in the DFS sequence. If there is such a node, the related edge is a back edge. We do not traverse back edges, but we adjust the current node's $back$ value is to $back[node] = \min(depth[ancestor], back[node])$ and mark the back edge as visited. We then depart the node on any unvisited edge that is not another back edge (called a forward edge). If none, we backtrack—below. The tree is now in the state of [Figure 4-23\(b\)](#) showing how the scan of edge (1,3) from node 3 causes us to realize (1,3) is a back edge (i.e., to a node we've already been at) and hence we pull up the $back(3)$ to $depth(1)$.

^[2] Note that when we say it "pulls up" $back()$ value, the $back()$ value itself is numerically reduced, but the node is

now associated directly with a higher node in the DFS tree and it is in the latter sense that it is "pulled up."

Now edge (2, 3) has been traversed, and (1,3) is a back edge, so the next step from node 3 can be either to nodes 4 or 5. Assuming forward edge (3, 5) is chosen next, node 5 gets $depth[5] = 4$, $back[5] = 4$. From here the only forward edge is (5, 6) and it is traversed next. Node 6 then gets $depth[6] = 5$, $back[6] = 5$. Node 4 is then visited traversing edge (4, 6). At node 4, we initially assign $depth[4]=6$, $back[4]=6$, but the scan of other edges at the node indicates that (3,4) is a back edge with node 3 as the ancestor. Therefore $back[4]$ is adjusted: $back[4] = \min(back[4], depth[3]) = 3$ is assigned. Note that this is equivalent to scanning immediately on arrival for back edges and directly assigning $depth[4]=depth[6]+1$ and $back[4]=\min(depth[6]+1, depth[3])$, where node 3 is an ancestor.

Now from node 4 there are no forward edges to follow, so we start backtracking from node 4. The DFS sequence records that we came to node 4 from node 6, so we backtrack to node 6 and compare $back[4]$ with $depth[6]$. If $back[4] \geq depth[6]$ it means that node 4 has no other connection with any other node except through node 6 and therefore, node 6 would be a cut-node. If $back[4] < depth[6]$ it means that node 4 has a connection with another node which is a parent of node 6 (i.e., in this case it is node 3, through back edge (3,4) and node 4). In our case, $back[4] = 3$ and $depth[6] = 5$, so node 6 is not a cut-node. Node 6 then inherits the back value of its neighbor node 4, if lower. That is, in backtracking from node v to w , if $back(v) < depth(w)$ then we are still backtracking through the same bicomponent and we assert $back(w) = \min(back(w), back(v))$ —in effect labeling node w as part of this component and propagating this back value to see where it originated (equality in $back(v)$ with $depth(w)$ identifies the cut-node).

At node 6 we backtrack to node 5 and compare $back[6]$ to $depth[5]$. Since $back[6] = 3 < depth[5] = 4$, we realize node 5 is also not a cut-node, and we update $back[5] = \min(back[5], back[6]) = 3$. Finally we backtrack from node 5 to node 3 and, comparing $back[5]$ with $depth[3]$, we find that the values are equal (both =3). This identifies node 3 as a cut-node for a biconnected subgraph that is accessible only through node 3. In effect the $depth()$ value of node 3 propagated into becoming the back value for all nodes in the subgraph rooted on it. This equality is what identifies node 3 as the cut-node. $back(w) > depth(v)$ would identify additional series cut-nodes. Again, if this were not the case, the propagation of a lower back value through discovery of any other back edge out of the subnetwork would have pulled up the back value of node 3 so that $back(3) < depth(5)$ would have been the case. At this stage the bicomponent rooted at node 3 is identified and removed, and backtracking continues. The edges forming the component rooted on node 3 are easily identified because they all have $back()=depth(3)$.

In the example, if another subnetwork were present connected to node 2, a new forward edge traversal would then delve down into it following the next step which is backtracking from node 3 to 2. The algorithm continues in a similar manner until the root node for the tree is reached and all cut-nodes identified. As it is, here, backtracking will continue until node 1 is revisited. Being the root node, we terminate and the list of remaining edges not removed when cut-node 3 was discovered (i.e., (1,2) (2,3) (1,3)) forms the second bicomponent of the example network. A pseudo-code summary follows:

procedure Biconnected(graph G, root node r) {

$i := 0, v=r, depth[r]=1, back[r]=1$

Repeat

Case of forward edge available from current node v:

true: (with next node = w){

if node w previously unvisited { $depth[w]:= back[w]:=i$ }

scan node w for any or all back edges (to other node z):

for each back_edge that exists:

{ $back[w]:= \min(back[w], depth[z])$ }

false: (with next node = w the DFS parent of v){

if $back[v] \geq depth[w]$ then cut-node w is discovered

{report w = cut-node;

remove all edges in biconnected component}

else $back[w] := \min(back[w], back[v])$

$i:=i+1$

until (current node v is again the root node r)}

The complexity is assessed as follows: Each edge is traversed once. Each node is visited once and from there, calls are made to each incident edge on the node to perform the depth first search tree algorithm. Therefore, the complexity is linear of order $O(n + m)$, where $n =$ number of nodes and $m =$ number of edges.

For the two-connected test, we recognize that the only condition for which a network is not two-connected is when it contains a span that joins two biconnected components. As such, if a bicomponent is identified by the presence of a cut-node v , and the parent of v is also another cut-node, then the edge (v, w) where w is the parent of v , is a cut-edge, indicating a graph that is not two-connected. The complexity is the same in both cases.

[\[Team LiB \]](#)

◀ PREVIOUS

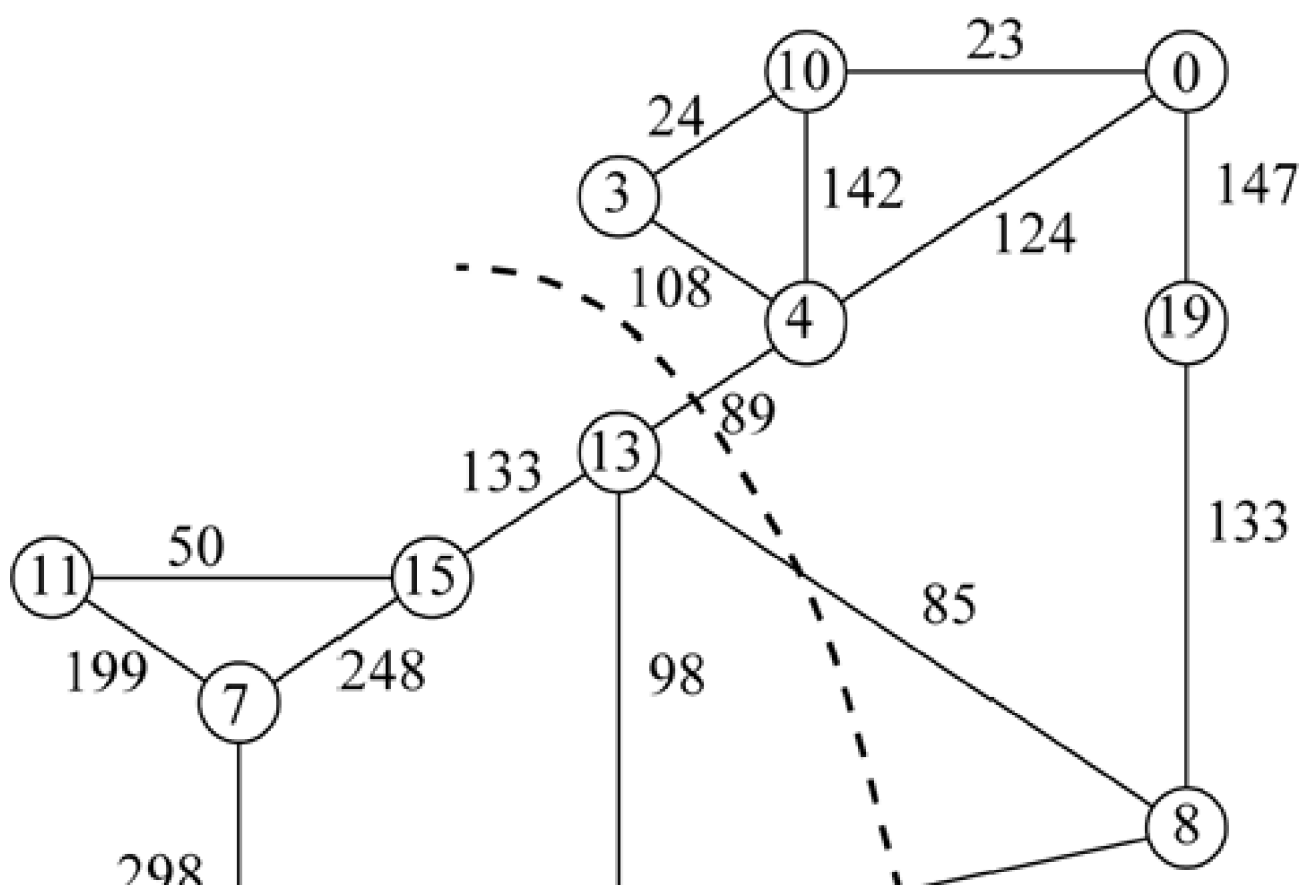
NEXT ▶

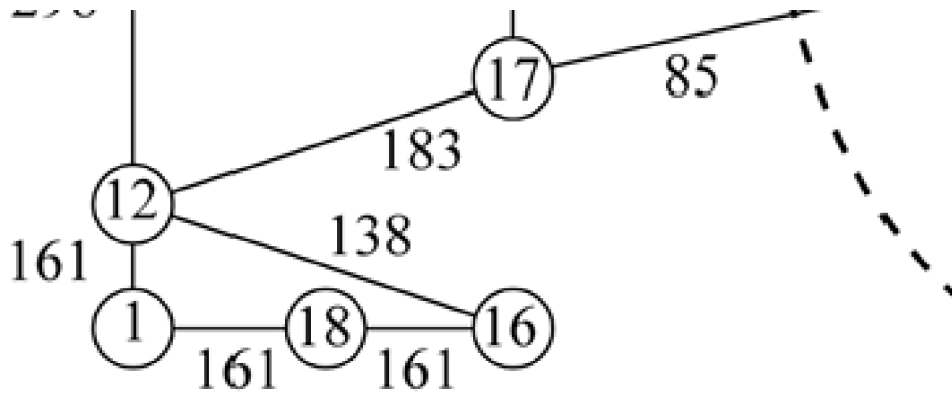
4.9 The Cut-Tree

A graph-theoretic construct called a Gomory-Hu cut-tree [Hu69] portrays the structure of a capacitated network in terms of the maximum amount of flow which can be supported between nodes. The cut-tree was considered for application to network recovery from node loss in [MaGr95]—from which we borrow the example below. The cut-tree of a network is a tree which is equivalent to the original network in the sense that it has the same nodes and the same maximum flow between each pair of nodes as the original network. But the cut-tree is, as the name suggests, a tree which provides only one path between any two nodes. The minimum capacity edge of the cut tree traversed on the path between any two nodes in the cut tree gives the maximum feasible flow (the minimum capacity cut) between those two nodes in the original graph. The cut-tree is relatively quick to construct, requiring only $(n-1)$ calls to a maximum flow algorithm, where n is the number of nodes in the network, rather than the $n(n-1)$ calls which would be required by calculating the maximum flow between each node-pair separately.

As an example, consider the network shown in Figure 4-24. The numbers beside each span represents the capacity on that span. Figure 4-24 and Figure 4-25(a) shows the beginning of the cut-tree construction. In the example we start (arbitrarily) with a maximum flow calculation between nodes 8 and 13, yielding a flow of 259, on the cut indicated. The nodes grouped with node 8 are on the same side of the min-cut as node 8; similarly, those nodes on the side of the min-cut with node 13 are collected with it. Note that the sequence of cuts chosen to develop the cut tree is arbitrary—(8,13) is only one starting point that could be chosen to develop the tree. Subsequent steps further divide the subsets of nodes into smaller groups, defining a new cut each time. The results of maximum flow calculations between nodes 8 and 4, and between nodes 13 and 12 are shown in Figure 4-25(b). Note that the capacity of the (8,13) cut arrived at in the first step does not change; each subsequent maximum flow calculation contributes one new edge to the cut tree, until we have $(n-1)$ edges joining n nodes in the cut tree. The results of additional maximum flow calculations are shown in Figure 4-25(c) and (d) and the final cut-tree is in Figure 4-25(e).

Figure 4-24. Example network and initial 8-13 cut.





With the cut-tree completed, the maximum flow between any pair of nodes can be obtained simply by finding the minimum-capacity edge between these nodes in the cut-tree. For example, the maximum flow between nodes 1 and 10 is 189, governed by the (4,10) cut. Because we are dealing with a tree, there is only one path between any pair of nodes, so there is only one choice of route through the post-failure cut-tree. The cut trees developed by choosing different sequences of node pairs for the maximum flow calculations may have different shapes, but they will all give the same value for the maximum flow between a specific node pair. The cut tree also conveniently identifies at least one minimum cut for each node pair: a minimum cut between two nodes in the original graph is given by removing the minimum-capacity edge of the cut tree between these nodes. The partitioning of nodes in the cut tree due to this removal also defines the minimum cut of the original graph in terms of the basic definition of a cut being a partitioning of nodes into disjoint sets each containing one of the two nodes in question.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.10 Finding All Cycles of a Graph

Finding the set of all distinct simple cycles on a graph can often be a preparatory phase problem for SBPP, p -cycle, or ring network planning. This section therefore looks at the problem of finding all cycles of a graph.

4.10.1 Fundamental Set of Cycles

An interesting property of an undirected graph is that the entire set of all cycles can be constructed from a *fundamental set of cycles* of the graph. The fundamental set of cycles completely determines the cycle structure of a graph because every cycle can be created from a combination of fundamental cycles. A set of fundamental cycles can be found from a spanning tree $\{V, T\}$ of a connected undirected graph $G = (V, E)$. As illustrated in [Figure 4-26](#), any edge in E but not in T will create exactly one cycle when added to T . Because every spanning tree of G has $|V|-1$ edges, the number of fundamental cycles is $|E|-|V|+1$. A fundamental set of cycles can also be defined by the set of bounded faces in a graph. Furthermore if $F = \{C_1, C_2, \dots, C_{|E|-|V|+1}\}$ is a fundamental cycle set of G , then any simple cycle in G can be written as $((C_{i_1} \oplus C_{i_2}) \oplus \dots) \oplus C_{i_n}$, where $\{i_1 \dots i_n\}$ is a selection of two or more fundamental cycles that generate the desired cycle.

When dealing with sets that are cycles, the previously defined symmetric difference operator \oplus can be thought of as a graphical XOR or mutual exclusion function. A simple but brute force algorithm to generate all cycles of a graph is therefore to first generate a fundamental cycle set from a spanning tree as above, and then run a binary counter to enumerate each combination of fundamental cycles, evaluating the symmetric difference of each to discover new cycles. This is, however, obviously $O(2^{|E|-|V|+1})$ in complexity and cannot selectively enumerate only the shortest cycles or specifically search out cycles of other specific properties when the set of all cycles is too large to work with completely in a network planning problem. To enumerate cycles in a more algorithmic way, which gives us some control over which cycles are discovered when the entire set of all cycles is too large to handle, we can instead use the following algorithm. In other contexts, however, knowing the way fundamental cycles can be combined to yield other cycles can be a useful tactic within certain improvement-search heuristics for ring or p -cycle-based design problems.

Figure 4-26. An example of a fundamental set of cycles. (a) Graph and a spanning tree T , (b) fundamental cycle set from T , (c) generating another cycle from two fundamental cycles.

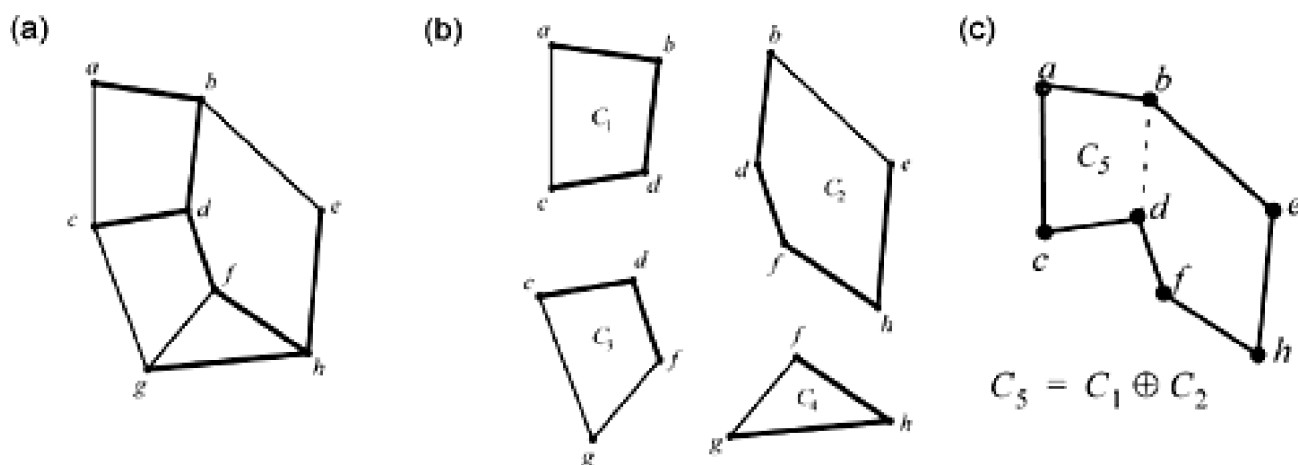
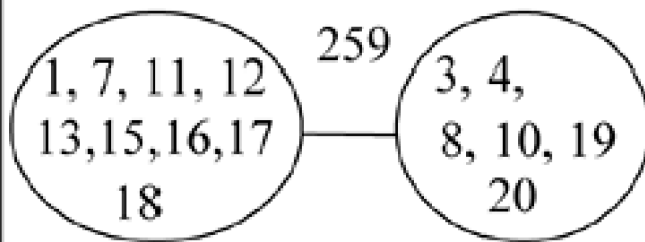
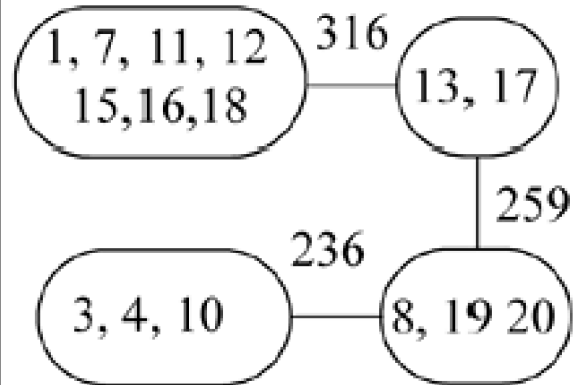


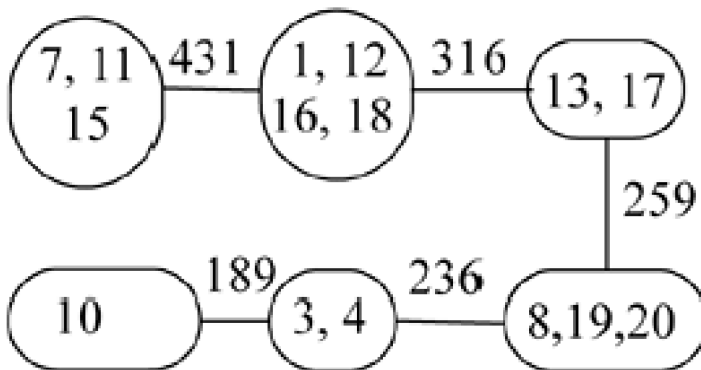
Figure 4-25. Construction of a cut-tree for the example network (adapted from [MaGr95](#)).



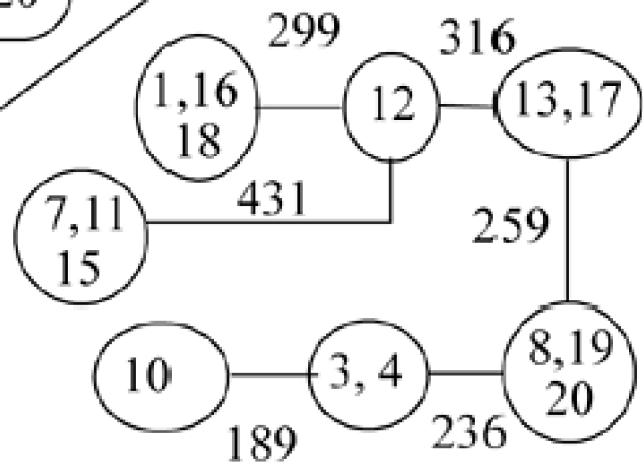
a) (8,13) cut



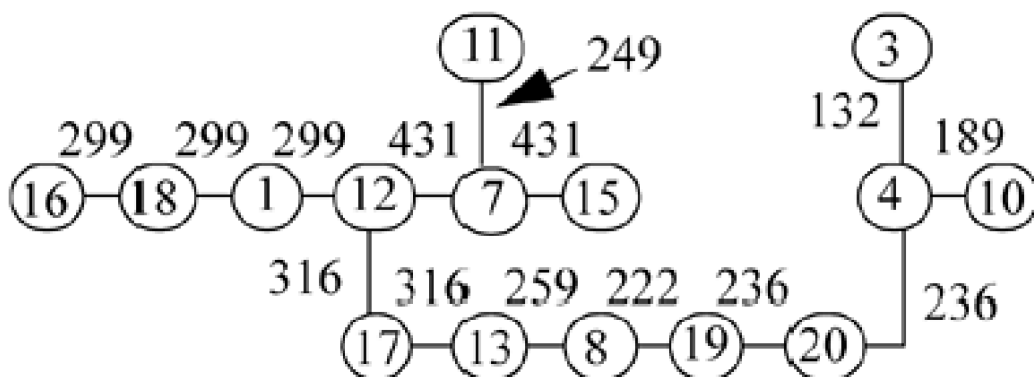
b) (4,8) and (12,13) cuts



c) (4,10) and (12,15) cuts



d) (1,12) cut



e) Final cut-tree

4.10.2 Depth-First Search for Cycle Enumeration

This section describes an *all-cycles finding algorithm* that has general uses in network planning. In [Chapter 10](#), on p -cycles, a variant on this algorithm is used to selectively populate the sets of candidate cycles to be considered for placement as p -cycles. But finding all cycles can also be a preparatory step in designing DWDM or MPLS SBPP networks where (by finding the shortest cycle or a set of candidate cycles that include the end nodes) we are assured that all primary and backup routes are disjoint. The algorithm uses a depth-first search to enumerate all cycles in an undirected graph. It is based on a similar algorithm by Johnson for directed graphs [\[John75\]](#), with some enhancements by Morley [\[Morl01\]](#) to improve its efficiency on undirected (i.e., bidirectional) graphs. For reference see also [\[YaDe76\]](#). In this algorithm cycles are generated by adding edges to a path until either a cycle is found or the cycle circumference limits (in hops or distance) are exceeded. To avoid exploring more paths than necessary, careful pruning is done to ensure that every cycle is generated only once. The description here is adapted from [\[Morl01\]](#).

The search begins at an arbitrary node s and a path $(s, v_1, v_2, \dots, v_k)$ is built using a depth-first search. To ensure that each cycle is unique and not simply a "rotated" specification of a previously found cycle, we only consider cycles rooted at s . As vertices are added to the path, they are marked to indicate that they are unavailable for extending the current path. The search continues adding edges to the path until one of three conditions is met: (1) the path is blocked by a marked node $v_{k+1} \neq s$, (2) length limits of a potential cycle are exceeded, or (3) a cycle is found (i.e., $v_{k+1}=s$).

In the general search, after all edges from a node v_k have been explored, we back up to the previous node. In doing so, we *unmark* v_k only if (i) a cycle was found or (ii) the cycle circumference constraints were exceeded during the current depth search. The reason for *unmarking* is that neither of these conditions rule out the possibility that further exploration, arriving from another incident edge, may still lead to (i) a different distinct cycle or (ii) a cycle of feasible length. Otherwise, if all paths extending out of v_k are blocked, the node remains *marked* as we back up past it. An edge from one node is said to be blocked if it leads to a marked neighbor node. This prevents the algorithm from searching parts of the graph that have been fully explored from the current node on the current in-depth probe. For each node w , a record is also kept of the adjacent vertices that are not in the current path but remain blocked by. This set $B(w)$ of adjacent vertices is referred to as the *predecessors* of w . If w is unmarked when backing up to the previous node, then all of its predecessors are also unmarked using a recursive call. The predecessors in turn unmark all of their predecessors and so on. This makes w and its predecessors available again for extending the path.

The above process continues until we back up to the root node s . At this point, all of the cycles traversing the first edge (s, v_1) have been found, so that edge is removed from the graph. This prevents duplicate cycles from being generated in the opposite direction. Once all edges incident on the current root node, s , have been explored, every cycle containing s will have been found. Note that it is not necessary to explore the last incident edge because all other edges from s will have been removed and so no further cycles can be formed. In other words, all cycles containing the last edge will have already been found. Pseudo-code for the sub-problem of cycle-finding given a specific root node, s , and one specific edge from s , (s, v) is:

```
procedure boolean Cycle(s,v) {
  flag := false;
  avail(v) := false;
  for each node w adjacent to v {
    if w = s {
      output "cycle found" cycle := path + w
      flag := true;}
    else if path + w exceeds circumference constraints {
      flag := true ;}
    else if avail(w) = true {
      path := path + w
      flag := flag OR Cycle(s,w) }
  }
  if (flag = true) {Unmark(v)}
  else {for each node w adjacent to v {B(w) := B(w) + v }}
  path := path - v
  return flag}

```

```
procedure Unmark(u) {  
  avail(u) := true;  
  for each node t in B(u){Unmark(t) }
```

Note that the procedure *cycle()* itself is a boolean function that returns only the "flag" true or false. It outputs discovered cycles as it executes, but as a boolean argument itself, it facilitates recursion to effect the DFS exploration down to leaf nodes of the DFS tree and back up each path. To see this, note that the overall code block under the first "for" statement explores ever downward, adding node after node to the current "path" until either a cycle has been found, the length limit has been exceeded, or the next node is already in the current path or otherwise blocked. The cycles themselves are output as they are found. As each new node is added to path, *cycle()* itself is called again from that standpoint. So the recursion drives the depth-probing aspect. This is countered by exhaustion of cycles in the current subregion (i.e., exploring off the end of the current path). What happens then is that all of the edges adjacent to the node are blocked and so the next instance of *cycle()*, which starts with *itsflag := false*, never sets the flag to true. Upon return to the calling instance of *cycle()*, this leaves the node *v* (just returned from) as *marked* and removes it from the current *path*. At this point all neighbors of *v* are re-opened for depth exploration because the end-point of the current path is backing up one. More specifically, if node *v* is marked (i.e., *flag:= false*), then upon return from the calling instance of *cycle()*, its neighbors are unmarked if and only if another node adjacent to its predecessor leads to a cycle or exceeds the circumference limit. In other words, if a cycle cannot be found (and the length limit is not exceeded) via any of the possible paths through a node, then that node and all its successor nodes remain marked. This prevents the search from re-exploring regions where there is no available path back to the root node, given the current path. Basically, the only way nodes are unmarked is when a cycle is found (or the length limit is exceeded). This means that the current path has changed sufficiently that an available path now exists back to the root node and the recursive call to unmark adjacent nodes allows other paths to explore it.

A visual picture is that path is like some kind of variable-length vacuum hose dropped into the DFS tree. It quickly falls right down to a leaf node to start with. Like a vacuum hose it suctions up all the cycles possible down at the bottom. It is then pulled up one node and squiggles around from there siphoning up all cycles with the same common set of nodes down to that point. The hose is then pulled up again, similarly slides down and extends into each other nearby region, vacuums up all the cycles there, and so on until the vacuum hose itself is eventually pulled right back up to comprise just (*s,v*) itself.

After all cycles containing *s*, starting in the direction of neighbor node *v* at *s*, have been found, each other neighbor node of *s* takes on the role of node *v* (except the last which is necessary as explained). When all cycles from *s* have thus been found, *s* itself is removed from the graph and the process begins again from the next node. (Note the accelerating effect this will have on the next master iteration). To complete the algorithm we observe that every cycle must lie within a biconnected component of the graph. Therefore, after the root node *s* is removed from the graph, we find the biconnected components of the subgraph ([Section 4.8](#)) and call the cycle finding procedure for each component. This is done recursively until all vertices have been explored. The pseudo-code for the overall main program is thus:

```
procedure FindCycles(s) {  
  for each node v adjacent to s in G {  
    Cycle(s,v)  
    G := G - (s,v);  
    G := G - v  
    H := Biconnected(G)  
    for each h in H {  
      FindCycles(h)}}
```

[[Team LiB](#)]

◀ PREVIOUS NEXT ▶

4.11 Optimization Methods for Network Design

Many important problems in network design belong to a class of problems known as *combinatorial optimization problems*. Combinatorial problems involve selecting among combinations of discrete alternatives, i.e., where the solution is a set or a sequence of integers or other discrete objects [\[Know89\]](#). Two properties of combinatorial optimization problems are:

- a. The number of feasible solutions increases rapidly as the size of the input increases.
- b. It is usually easy to construct a *feasible* solution. The hard part is to find an optimal solution.

Point b distinguishes this class of optimization problems from other problem domains where it is the construction of a single feasible solution (i.e., one that meets all requirements) that is the hard part. A classic combinatorial optimization problem is the *Traveling Salesman Problem* (TSP). This involves finding a tour that visits n cities where the distance between each of the pairs of cities is known and the objective is to minimize the total length of the tour. Like many combinatorial optimization problems, TSP is easy to state but difficult to solve because there are so many alternatives to consider. A naive approach to solving an instance of this problem is to list all possible solutions, evaluate their objective functions and pick the best solution. But evaluating all possible solutions quickly becomes impractical because the number of possible tours is $(n-1)!$. Thus, TSP with only 20 cities has more than 10^{17} possible tours! Even if one could evaluate a million per second it would take over 3.8 millennia to enumerate all possible tours. This behavior, called *combinatorial explosion*, is common in combinatorial optimization problems. Nonetheless, powerful methods exist that can help us find good, even provably optimal, solutions to many quite useful real-world instances of such problems.

4.12 Linear and Integer Programming

Linear and integer programming methods arise from the field of *mathematical programming* methods in the Operations Research (O.R.) community.

A Mathematical Programming Model is a mathematical decision model for planning (programming) decisions that optimize an objective function and satisfy limitations imposed by mathematical constraints. [\[Know89\]](#)

The first thing to clarify is that these methods have little to do with "programming" as it is usually meant today. Rather, Casti explains the origin in the 1930s of the somewhat misleading terminology still used today:

O.R. is concerned with creating minimal cost (or maximal benefit) schedules or plans. In the early days of O.R. such plans were labeled "programs". This led to the terminology that persists to this day by which we call finding optimal schedules or plans a "programming" problem. ...a programming problem has nothing to do with computer programs, per se. Rather it refers to determining a plan of action that is in some sense optimal. [Casti p. 185]

There are four main types of mathematical programming problems. In *Linear Programs* (LP) the objective and constraints are all linear functions and the decision variables may take on continuous real values. The process of solving an LP is called *linear programming*. In *Integer (Linear) Programs* (ILP) one or more of the decision variables are restricted to integer values only. All of the objective and constraint functions remain linear. In a pure ILP all variables are strictly integer. In a *Mixed Integer Program* (MIP) some variables are continuous and others are restricted to whole numbers or integers. A more specific form of MIP is called a *1/0 MIP* problem in which some of the integer variables are further restricted to take on only binary {1,0} values, typically representing yes/no decision outcomes. In *Nonlinear Programs* one or more of the functions (objective or constraints) is not linear. We do not consider nonlinear programs further here.

ILP, MIP and 1/0 MIP approaches to network planning and even on-line operations are significant and practical modern tools for both production and research use. The solution engines available today can handle enormously large optimization problems of these types, giving optimal or high quality solutions in reasonable amounts of time. On many problems where a solution with a formally optimal termination would take too long, a time-limited run nonetheless produces a feasible solution that is as good or better than highly tuned heuristics for the same problem running for the same time. With various "relaxations" the solvers can also provide lower bounds against which to gauge the effectiveness or performance of other more ad hoc or heuristic design procedures. LP/ILP methods are especially relevant to this book because:

1. The syntax and structure of mathematical programming provides a precise and compact language for developing, precisely specifying, and then understanding the underlying nature or structure of the planning problems of interest.
2. The same formalized problem statements can be almost directly converted into executable problem tableaus to obtain actual planning problem solutions or, in a research context, to search for the existence or feasibility of certain desirable combined network properties.

4.12.1 The Role and Limitations for Mathematical Programming

In the author's research group, and this book in general, there is an emphasis on formulating and testing ideas for new network architectures and network design methods with LP/ILP methods. MIP or 1/0 MIP models are often involved. The MIP formulations play an important role in research because they allow us to study, test, or compare alternatives on the most rigorously objective basis. By comparing methods or designs on test cases that are solved to optimality we are certain we are looking at the truth of the best that each alternative can achieve; that the comparison is not affected by side-effects of differing heuristics that might have been used to obtain the designs. This is the sense in which we sometimes refer to ILP methods as "the network researchers *microscope*"—ILP solutions let us see

what is *true* and what is *possible* about various architectures and ideas tested by ILP studies and experiments. In our view, the practical use of such ILP/MIP tools for research, planning, and even on-line operations, are often too quickly dismissed with assertions such as "The ILP approach is *NP*-hard and so doesn't scale up^[3]—with the apparent implication that *only* heuristics are justified. But this is too simplified an argument to forgo the use of one of the most powerful tools a network researcher or planner could possibly have. In practice there are enormously many uses for such techniques and we predict that on-line real-time use of O.R. techniques will arise in future transport networks. Near real-time use of ILP/MIP models is already the norm in the highly competitive airline industry where it is often attributed to making the difference between profit or loss on overall operations. Notwithstanding that there will always some problem size that is beyond our ability to solve to optimality, MIP models can serve at least the following roles in network planning and research:

^[3] Even worse, basic network architectural principles or options, such as SBPP or *p*-cycles as examples, are sometimes also dismissed based on the same association with ILP methods used *to study them*. The misconception is that a basic architectural concept such as SBPP or *p*-cycles is somehow intrinsically dependent on ILP. But a correct understanding separates basic network architectures from the methods used to design them, which can be *either* ILP-based or heuristic.

1. MIP models serve as a technical language precisely defining the design intent.
2. A MIP model can often reveal basic structural aspects of the problem that can inspire decomposition or heuristic insights.
3. The MIP model may be solvable with reduced variable space, such as for example with a reduced number of eligible working routes, a restricted *p*-cycle candidate set, or with a grouping of flows into a fewer number of wavebands, etc. In such cases, the full MIP model solved with a reduced decision space is effectively a heuristic approach to the problem. Often such heuristics can be quite effective and require far less development effort than a completely custom heuristic program.
4. Solutions to the MIP model at small sizes, even if they take a long time to obtain, serve as valuable quality references against which any heuristic for the problem can and should be tested. This gives a basis for confidence when a heuristic is then applied to larger problems where it will never be known exactly how close to optimal its solutions are.
5. By setting reasonable MIP gaps and/or run-time limits, quite good, even optimal solutions to some problems can be obtained even without a full termination. An important aspect of certain MIP problems is that the solver may actually find a high quality solution, indeed sometimes even optimal, within the first minute or so, but could then spend the life of the universe effectively just to *prove* the optimality of the early feasible solution. Viewed in this light, every difficult MIP problem (that is at least feasible within a reasonable time) always has a readily available heuristic for its solution: If *problem X* takes too long to solve optimally, then a typically still very good heuristic (available with no further computer programming) is "*problem X given an hour of run time*." Especially for comparative studies results obtained on this basis could often be even more reliable and repeatable than attempting to reach comparative conclusions or general insights based on completely heuristic design methods for each alternative.
6. Finally, as mentioned, there is a sense in which an exactly-solved MIP model is, in studies of a new idea or approach to networking, analogous to a laboratory microscope for looking into the problem and seeing what is true without any obscuring effects from heuristic approximations or biases arising from the heuristic solution strategy. For example, it is only with the exact MIP model can one truly ask a research question such as "in such and such a network context, how far will the working routes go beyond their shortest path distances?" Any attempt to answer such a question with a solution obtained from a heuristic design can never be sure of the truth of its findings. Thus, MIP solvers assist us in truly doing network science in that we can ask fundamental questions about the nature of various networking phenomena by formulating the correct models, solving them to optimality, and then studying the properties of the resultant routings, topology, protection capacity distribution, cycles employed, etc.

So, for many purposes in research and development especially, MIP models have an important role. Their main drawback is when the solution times (and sometimes memory requirements) are unacceptable for production use in network planning tools. This is a context where fast, even if approximate, heuristics may be preferable or necessary. But ideas leading to good heuristics almost always benefit from studies of the basic architecture and its behavior with optimal models. And heuristics always need the optimal models as a benchmark for performance assessment at smaller problem sizes before we can rely on them at large problem sizes where we will never actually know how close to optimal the designs are.

4.12.2 General Symbolic Form of an LP or ILP

An LP or ILP consists of four main components: a set of decision variables, an objective function, a set of constraints, and a set of bounds. A *decision variable* is the amount or level of each resource to be determined. The *objective function* is a function of the decision variables (e.g., cost, profit) that we want to either minimize or maximize. The *constraints* and *bounds* are sets of inequalities (or equalities) that restrict the decision variables to values that would be considered an acceptable or feasible solution to the problem. Any solution that satisfies all the constraints and bounds is said to be a *feasible solution*. Such problems have a canonical algebraic form as shown in [Figure 4-27](#).

Figure 4-27. General form of a Linear or Integer Program.

$$\begin{array}{ll}
 \text{Maximize:} & c_1x_1 + c_2x_2 + \dots + c_nx_n \quad \left. \vphantom{c_1x_1 + c_2x_2 + \dots + c_nx_n} \right\} \text{Objective} \\
 \\
 \text{Subject to:} & \left. \begin{array}{l}
 a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \quad \{\leq, \geq, =\} b_1 \\
 a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \quad \{\leq, \geq, =\} b_2 \\
 \vdots \\
 a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \quad \{\leq, \geq, =\} b_m
 \end{array} \right\} \text{Constraints} \\
 \\
 & 0 \leq x_j, \quad j = 1, \dots, n \quad \left. \vphantom{0 \leq x_j, \quad j = 1, \dots, n} \right\} \text{Bounds}
 \end{array}$$

In the standard form x is a vector of decision variables $\{x_1, x_2, \dots, x_n\}$. All decision variables must be non-negative. Constant terms cannot appear on the left hand side (LHS) of a constraint. No variable can appear on the right hand side (RHS) of a constraint and no variable can appear more than once in a function, i.e., objective or constraint. The *sense of optimization* is either maximization or minimization. The coefficients on the left hand side of the system of constraints collectively form the *coefficient matrix*.

4.12.3 Example Development of an Optimization Model

The following example is excerpted from [\[FoGa93\]](#) to create familiarization with the process of developing an LP and then an ILP model.^[4] The problem is first posed verbally as:

^[4] [Figure 4-27](#) through [Figure 4-31](#) and related discussion are adapted with permission from [Mor101](#).

"A steel company must decide how to allocate production time on a rolling mill. The mill takes unfinished slabs of steel as input and can produce either of two products: bands and coils. The products come off the mill at different rates and also have different profitabilities. The weekly production that can be justified based on current and forecast orders are: Bands: 6,000 Coils: 4,000. The question facing the company is: If 40 hours of production time are available, how many tons of bands and coils should be produced for the greatest total profit?" [\[FoGa93\]](#)

Figure 4-31. ILP branch and bound tree for the Bands and Coils example. A new LP is solved at each node in the tree.

Bounds

Solution

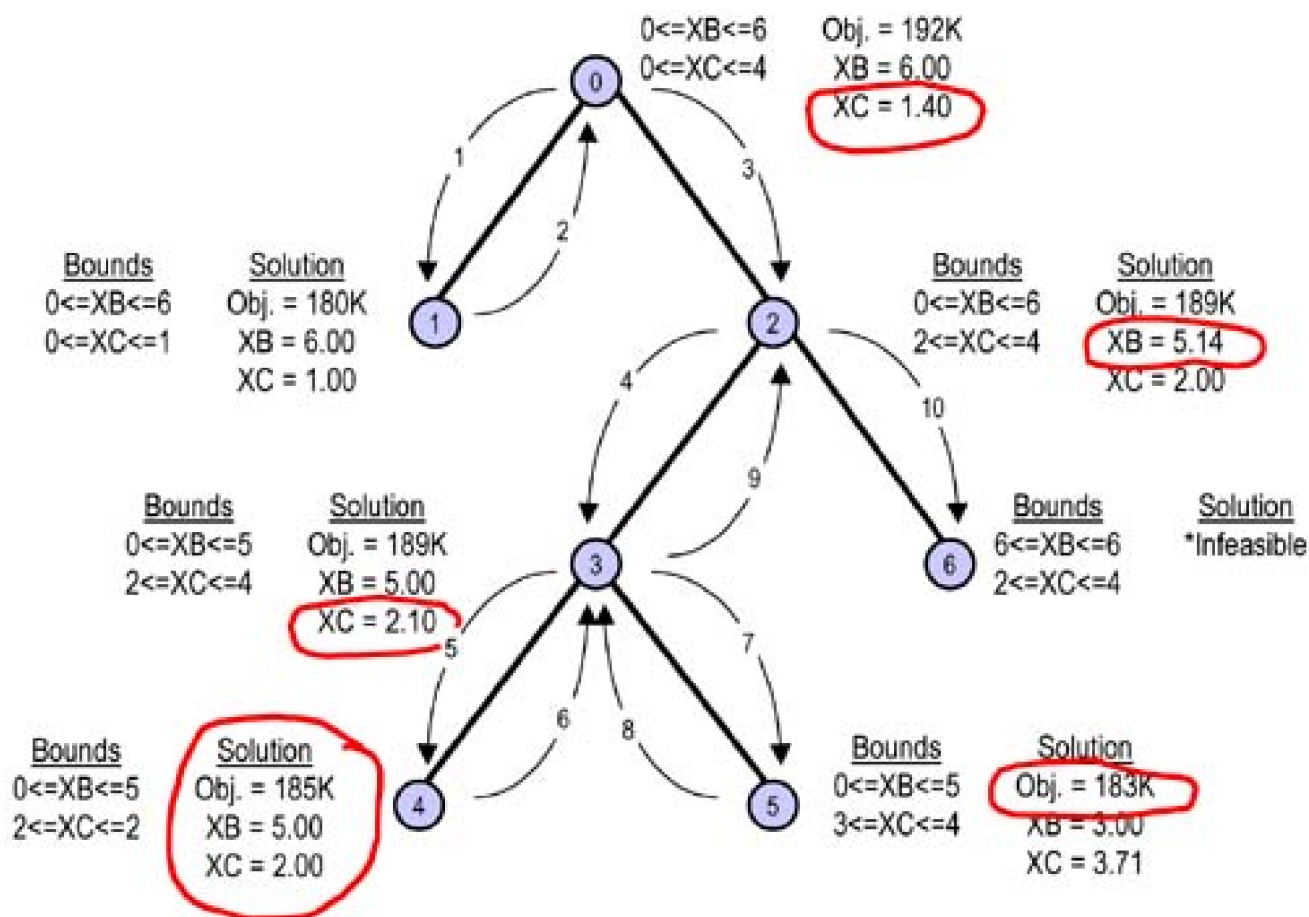


Table 4-1 shows the profitability and production rate data for the problem.

Table 4-1. Data for Bands and Coils Problem

	Tons / hour	Profit / ton
Bands	200	\$25
Coils	140	\$30

Step one is to construct the "verbal model", putting the objective and constraints into words. The result in this case might be:

Maximize: total profit

subject to: total number of production hours ≤ 40

tons of bands produced $\leq 6,000$

tons of coils produced $\leq 4,000$

and the resulting symbolic model would be:

Maximize:

Equation 4.9

$$25 \cdot X_B + 30 \cdot X_C$$

Subject to:

Equation 4.10

$$(1/200) \cdot X_B + (1/140) \cdot X_C \leq 40$$

Equation 4.11

$$0 \leq X_B \leq 6000$$

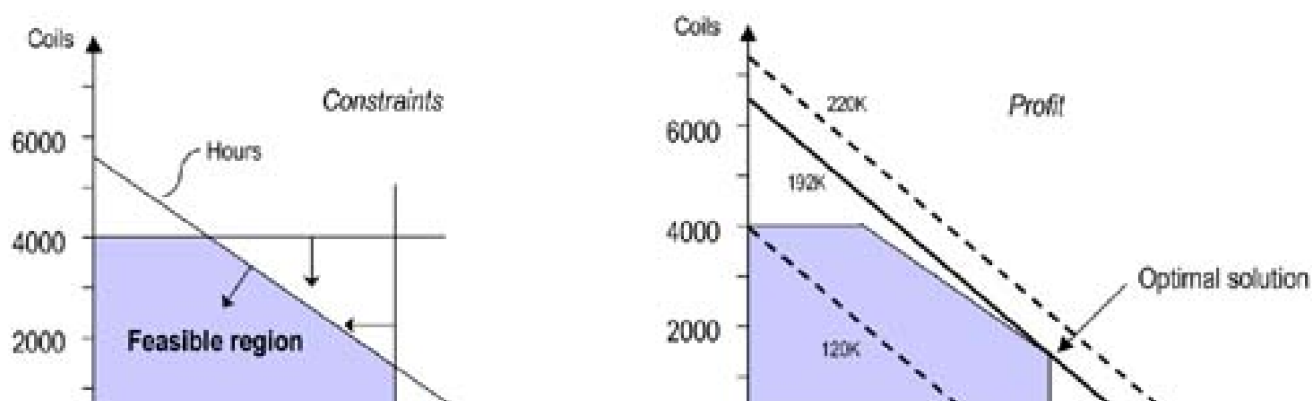
Equation 4.12

$$0 \leq X_C \leq 4000$$

where X_B and X_C are the number of tons of bands and coils to be produced, respectively.

Let us now look at the mathematical structure of the system of objective and constraints. Because it involves only two variables, this problem can be represented and solved graphically as shown in [Figure 4-28](#). In general, the multi-faceted shape defined by all the constraint planes (called the *polytope*) actually exists in an n -dimensional space of all the variables. The left panel in [Figure 4-28](#) portrays just the constraint lines. Every (X_B, X_C) point that satisfies all constraints is said to represent a feasible solution. The right hand panel shows lines of constant value in the objective function, e.g., lines where $X_C = (C - 25 X_B)/30$ where C is some arbitrary constant representing the total profit. By inspection of the objective function at every vertex of the feasible region one finds that profit is maximized at the vertex shown. This shows an important fundamental property of LPs; the set of feasible variable values that maximizes the objective function is always found at a vertex of the polytope formed from intersection of all the constraint planes. More generally, the space of all feasible solutions defined by intersecting constraint planes in an LP (the polytope) is a *convex* set. This means that if any two points are inside the polytope, then so is every point on the line between them. The corner points where constraint planes intersect are therefore of special interest because the optimum (maximum or minimum) is always found at some "corner point" of the polytope. In fact this is why LP problems can be solved so efficiently: the solution methods proceed directly from one corner to another until the optimum is discovered. They need not search the entire polytope volume or even its surface area.

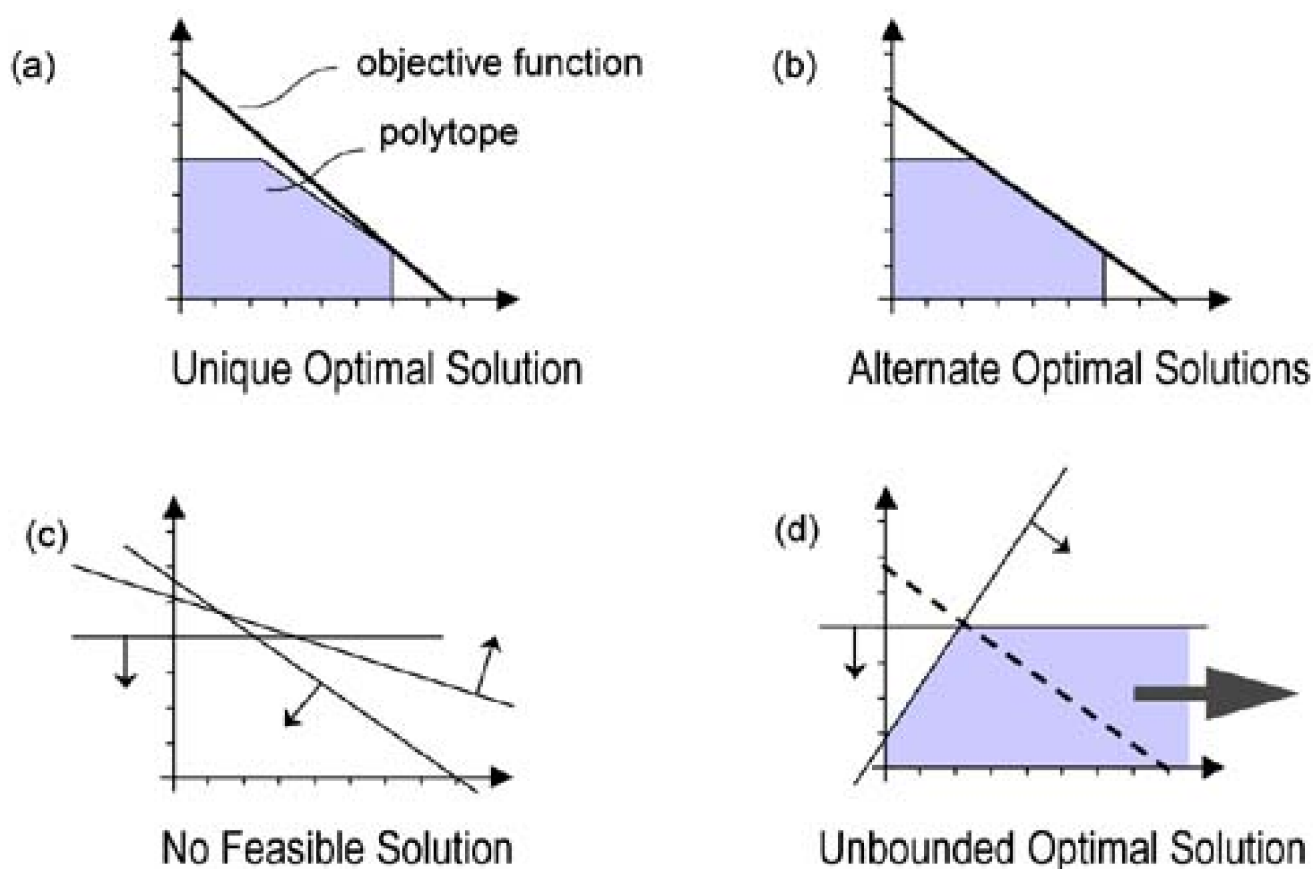
Figure 4-28. Graphical representation of the bands and coils LP.





There are four possible outcomes to an LP solution attempt, illustrated generically in terms of the constraint space and the objective function in [Figure 4-29](#). In (a) the outcome is a unique optimal solution. In (b) the objective function has a constant and maximal value at all points on one of the constraint planes. This means that a certain range of solutions are all equally good policies and that some further preferences (outside the current problem formulation) might be used to decide, or additional constraints added to the problem. In (c) the problem is not feasible. This happens if the set of constraints cannot be simultaneously satisfied by any arrangement of variable values. Conversely, in (d), the problem is unconstrained: the constraints leave the objective function unbounded in some vector direction of variables in the n -space. Both of the latter situations (infeasible or unconstrained) indicate an error in the model or the basic problem understandings from which the model was developed.

Figure 4-29. Four possible outcomes from an LP.



4.12.4 Making the Problem an Integer Linear Program

Let us further develop the example to illustrate the much more demanding nature of an ILP as opposed to an LP problem and to make the related point that one cannot just "round up" LP solutions to get the integer solution. The true ILP solution can be quite different. We can make the bands and coils example into an ILP problem by stipulating that orders for bands and coils can only be placed and filled in 1,000 ton lots. In that case we have to restate the problem to solve for the slightly different variable X_B' and X_C' which will be the number of 1000-ton *lots* of bands and coils to be produced, respectively.

Maximize:

Equation 4.13

$$25000 \cdot X_B' + 30000 \cdot X_C'$$

Subject to:

Equation 4.14

$$(1000/200) \cdot X_B' + (1000/140) \cdot X_C' \leq 40$$

Equation 4.15

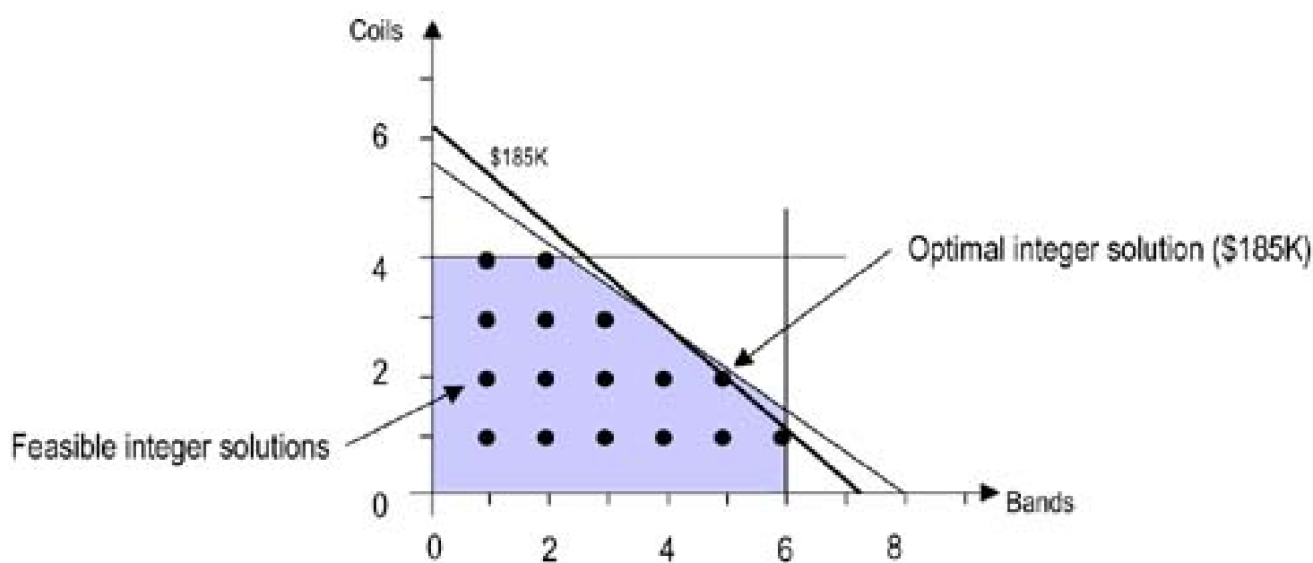
$$0 \leq X_B' \leq 6 \quad 0 \leq X_C' \leq 4$$

Equation 4.16

$$X_B', X_C' \text{ integer}$$

Now, if we revisit the sketch of the feasible region of [Figure 4-28](#), we have an overlay of discrete points which are the actual feasible solution space. This is shown in [Figure 4-30](#).

Figure 4-30. Feasible solution space in the ILP version of the bands and coils problem.



The problem is more difficult computationally because now all discrete integer solutions near any vertex of the prior LP (or "relaxed") version of the problem have to be checked in detail. One way that ILP solvers approach the problem is through a *branch-and-bound* search tree process. The root node of the tree is the *LP relaxation* of the complete problem, i.e., all integer variables are allowed to be fractional if the optimal solution calls for it. If by chance the LP solution of the relaxed problem produces all-integer variable values, then this is optimal for the ILP problem as well. Such circumstances are sometimes called "lucky problems." More formally they involve the

concept of total unimodularity, which follows.

More generally, however, if the solution of the LP relaxation has fractional values for integer variables, one of those variables is selected for branching and two new subproblems are generated, each with an additional bound on the branching variable. In one subproblem, an upper bound is added on the chosen variable at the *floor()* of its real value, and in the other, a lower bound at the *ceiling()* of its fractional value, i.e., the next higher integer value. The subproblems can result in an all-integer solution, an infeasible problem, or another fractional solution. If the relaxed solution of a subproblem has fractional values for integer variables, the process is repeated. Branches are "fathomed" if the subproblem is infeasible, if the objective value is worse than the current best integer solution, or if the subproblem gives an integer solution. Fathoming is one aspect of branch-and-bound wherein whole branches of the search tree are disqualified from further consideration. The other is the addition of new bounds added at each branch taken in the tree. The process continues until all branches have been fathomed.

The ILP search space consists of a tree of LP subproblems, in which each successively lower vertex in the tree has more and more restrictive bounds placed on its integer variables. For each vertex in the search tree, there are a number of possible LP solutions to be tested corresponding to each combination of branching and bounding of the fractional LP variables up or down. So in a problem with n variables there may be up to 2^n combinations per vertex to test against the constraints (for feasibility) and for which to evaluate the objective function (if feasible). Practical implementations do better than this by cutting off the search tree in certain directions through the branch-and-bound procedure.

The branch-and-bound process is illustrated in [Figure 4-31](#). The root node starts with fully relaxed LP solution and initial bounds from the problem statement. Since the solution results in a fractional value of X_C , that variable is chosen to perform the branch-and-bound on. A new (relaxed) subproblem where X_C is bounded to 1.0 or below is the next LP to be solved (at node 1 in the tree). The outcome shows an objective function less than the initial, but all integer variables do take on integer values, so this solution is recorded as our best-to-date integer solution. Node 2 is then visited where bounds force variable X_C to 2 or higher. Now X_B becomes fractional, so at nodes 3 and 6 respectively, additional bounds are used to force solutions where X_B is bounded to 6 or above (node 6) or 5 and below (node 3) while the bounds for X_C remain unchanged from node 2. Node 6 cannot provide a feasible solution (i.e., it is not possible to find a solution to the LP where all constraints and bounds are met), and at node 3 X_C turns fractional again ($X_C=2.1$), so at nodes 4 and 5 bounds are used to inspect LP solutions where X_C is first forced to 2 or below or up to at least 3, respectively. At node 5 the best objective function with X_C forced to 3 or more is seen to be lower than the best integral solution already found at node 4 so no further branching in that direction is considered. Similarly because node 4 (and node 1) became fully integral no further branching goes on from either of those locations and so the process is complete and the best integral solution found is $X_C=2$ thousand ton lots, $X_B=5$ thousand ton lots, and the objective value is \$185,000.

Contrast this result to the notion of just rounding up the LP result. The latter would prescribe $X_C=2$ (thousand ton lots), $X_B=6$, which is not even feasible. What if we round X_C down instead of up then? Then we'd have $X_C=1$, $X_B=6$ which is feasible, but for which the objective function is only \$180,000. This illustrates that the distinction between LP and ILP is a real and practical one that really does matter, not just a mathematical nicety. Optimal ILP solutions can be significantly different from any integral point in the solution space that lies adjacent to the pure LP solution. Here we saw that the integral points neighboring the LP result would offer us only choices of X_C , $X_B= (2, 6)$ or $(1, 6)$ whereas the true ILP solution lies over at $(2, 5)$ and is worth \$5,000 more in the objective function.

One reason we stress this point is that ILP comes with a large computational penalty (over LP), so it is tempting to hope the LP result might be close enough. But in network design problems in particular, which often have 1/0 ILP aspects, the LP solutions often won't even describe any feasible construction at all. For instance, rounding up does not even have meaning if a decision variable (to choose a certain backup route, say) has a value of 0.7.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.13 Duality

An interesting and useful property of LPs (or ILPs) is *duality*. Every LP problem defined in its standard *primal* form (all of the above are primal forms) has a corresponding *dual* formulation which has the same objective function value at optimality but has a transformed structure of objective function, variables, and constraints. Every constraint in the primal corresponds to a variable in the dual. With a primal problem in *standard form* construction of the dual problem is facilitated by rules which apply in all cases [Taha87](#), (p.116-170). Duality is an extensively studied topic in the O.R. literature. Here we only introduce the basic concepts. The main interests in duality are for sensitivity analysis and for computational advantage on certain problems. Our main interest is in the latter.

The computational advantage of solving the dual, rather than the primal problem, is greatest when there is a large number of constraints. If the primal problem has n variables and m constraints, the dual will have m variables and n constraints. Since the execution time of an LP is dependent on the number of constraints, it can be more effective to solve the dual rather than the primal formulation when $m \gg n$.

The main formality of writing the primal problem in "standard form" is the use of *slack (or surplus) variables* to convert inequality constraints to equalities. Also, any unrestricted variables (variables that may be either negative, zero, or positive) are replaced by dummy variable pairs, each of which is individually restricted to being non-negative. The standard primal form is thus written as:

Maximize (or min):

Equation 4.17

$$z = \sum_{j=1}^n c_j \cdot x_j + \sum_{i=1}^m 0 \cdot u_i$$

Subject to:

Equation 4.18

$$\sum_{j=1}^n a_{ij} \cdot x_j + u_i = b_i \quad i = 1, 2, \dots, m$$

Equation 4.19

$$x_j \geq 0 \ ; \ u_i \geq 0 \quad j = 1, 2, \dots, n \quad i = 1, 2, \dots, m$$

In standard form, the objective function and constraints are supplemented by non-negative slack variables u_i which convert an inequality into an equality. Say, for example, a constraint in some primal LP is of the form $5 \cdot x_1 + 3 \cdot x_2 + x_4 \leq b_5$. This becomes $5 \cdot x_1 + 3 \cdot x_2 +$

$x_4 + u_5 = b_5$ and u_5 is called a *slack* variable. If the inequality was of the opposite sense; $5x_1 + 3x_2 + x_4 \geq b_5$ then the standard form includes the constraint $5x_1 + 3x_2 + x_4 - u_5 = b_5$ and u_5 is referred to as a *surplus* variable.

The slack/surplus variables often have physical meanings and provide added insights of a practical nature. For instance, surplus variables in the formulation for mesh spare capacity design can indicate the extent to which certain spans could have their working capacity increased *without* adding any more spare capacity for 100% restorability. Because slack variables have no costs associated with them they do not change the objective function. The standard form recognizes them as being logically present, however, in the objective function. This facilitates the application of defined rules for generating the dual problem. Note also that the coefficient of any slack variable in the constraint matrix is only ever +/- 1. To put the primal in standard form each constraint in the primal is also multiplied by -1 if needed (and sense of inequality flipped) so that every right-hand parameter b_j is non-negative in the standard form. Finally, if any of the true (not slack) variables x_j is unrestricted in the original problem, it has to be replaced by a pair of non-negative dummy variables in the standard form. That is:

Equation 4.20

$$x_j \text{ unrestricted} \rightarrow x_j = x_j' - x_j''; \quad x_j', x_j'' \geq 0$$

Having put the original problem in this standard primal form, the dual problem is constructed by:

1. Reversing the sense of optimization, e.g., minimization in the primal implies maximization in the dual.
2. The objective function of the dual is comprised of m variables, y_i , with coefficients which are the corresponding right-hand-side parameters of the primal (the b_i 's).
3. Each coefficient of the standard-form objective function (e.g., n c_j values and up to m zeros corresponding to the slack variables) becomes the right-hand-side parameter for a constraint in the dual. The y_i variables in each constraint have coefficients which come from the corresponding column of a_{ij} coefficients in the primal.
4. If the primal is a maximization problem, the y_i variables in the dual are:
 - a. unrestricted if they correspond to an equality constraint in the original primal (e.g., no slack/surplus variable needed in the standard form).
 - b. strictly non-negative if the corresponding constraint of the original problem was a \leq inequality (when b_j is adjusted to be positive or zero.) i.e., a slack variable is used.
 - c. strictly non-positive if the corresponding constraint of the original problem was a \geq inequality (when b_j is adjusted to be positive or zero.) i.e., a surplus variable is used.
5. The converse of 4. applies if the primal is a minimization problem.

[Figure 4-32](#) gives an example of a primal form, the corresponding standard primal form, and the corresponding dual problem (adapted from [Taha87], p. 118). [Figure 4-33](#) further details the transformation relationship between standard-form primal and dual problems. Note

that it is the "slack" variables added to build out the standard primal form that lead to the last two constraints in the dual: $y_1, y_2 \leq 0$. In [Figure 4-33\(a\)](#), x are primal variables, c are the coefficients in the primal objective, and u are slack variables. In (b), y are the dual variables and b are their coefficients. The region to the right of the dashed vertical line in the primal constraint matrix is all zeros except in slack variable positions arising from an inequality constraint in the original problem. The corresponding constraints generated in the dual arise below the horizontal dashed line are only single-variable range restrictions on variables in the dual, as in the example above. Note consequently that each true variable in the primal generates one non-trivial constraint in the dual and the dual has one variable for each constraint in the primal. Thus, primal problems with few variables but many constraints may be more easily solved in the dual form.

Figure 4-32. Example of Primal, Standard Primal, and Dual LP Forms.

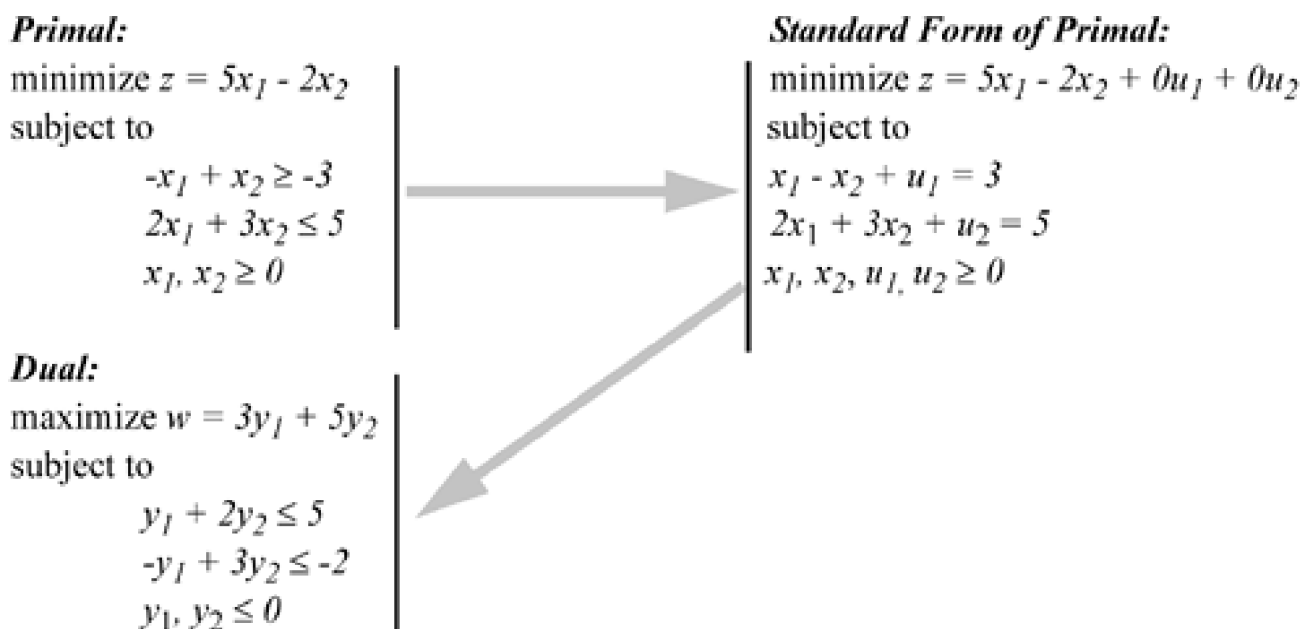
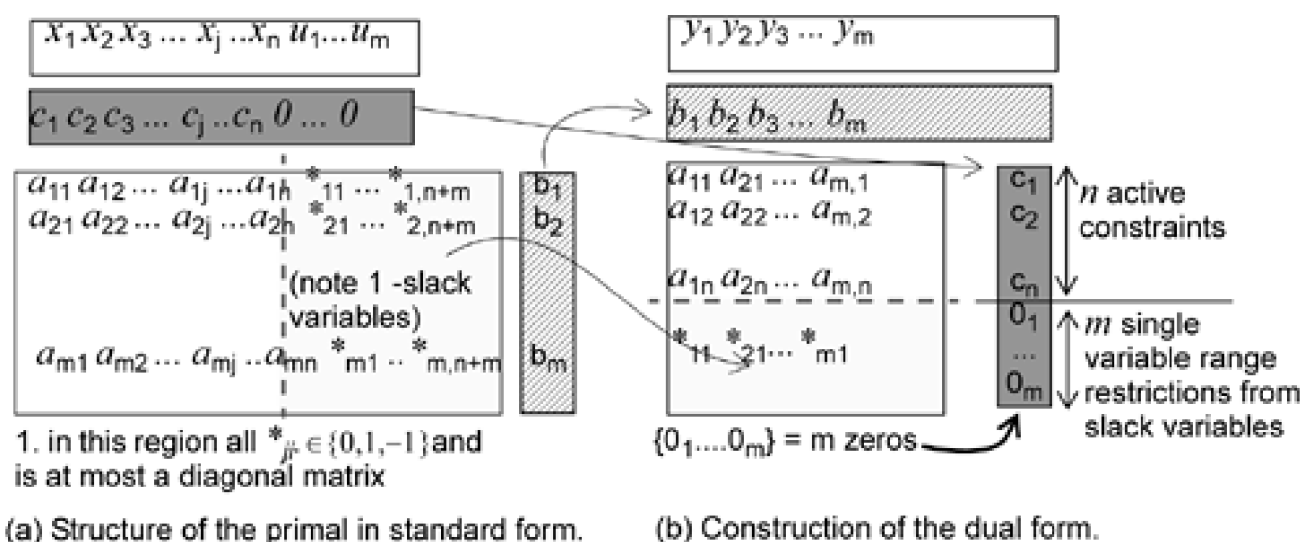


Figure 4-33. Constructing the dual problem from the primal LP in standard form.



The theory and uses of the primal-dual relationship are developed further in many O.R. texts. A property with several uses is that when, in the process of solving an LP, the primal is feasible but non-optimal, the corresponding solution in the dual will be apparently better than optimal, but also infeasible. The converse also applies: when the current primal solution is infeasible and (better than) optimal, the dual will be feasible and sub-optimal. This can be appreciated from the opposing nature of their optimization senses. For a primal minimization problem, the maximum (feasible) objective function value of the dual is identically the minimum feasible value of the primal solution. Thus, *at optimality primal and dual meet in terms of objective value*, but at all other times the dual objective value is below the feasible minimum and the primal is above the same point (for a primal minimization problem). Therefore on difficult LP problems, a pair of incomplete primal and dual solutions can still provide a two-sided bound on the optimum value. For a primal minimization problem, the incomplete dual solution is a better-than-optimal lower bound and the incomplete primal solution is a feasible upper bound. The use of a primal and dual problem pair to bracket the optimal solution of an LP in this way should not be confused with the later use of an LP relaxation as a lower bound for an ILP solution. In the latter case an LP is used to lower-bound an ILP, but both remain primal problems. The former concepts pertain to estimating the optimal solution of a pure LP where it is difficult to solve the LP to optimality.

Another application of duality principles is the *dual-simplex method* for LP solving (which CPLEX implements as its TRANOPT command). The dual simplex method should always be tried on a difficult LP problem because it approaches the solution from a dual-like direction of better-than-optimal infeasible states in the solution space and works to clear the infeasibility in the primal. The regular simplex method works from the direction of improving always-feasible sub-optimal solutions until optimality is reached solely from the primal side of the solution space. Dual simplex operates on the primal problem, but it starts from an infeasible better-than-optimal solution from the dual and makes reference to the constraint relationships of the dual to determine which new simplex pivots to follow to work toward feasibility. When

the constraints of the dual are all satisfied the solution just reaches feasibility, at the optimum value.

Some other properties of interest:

- a. The dual of the dual is the primal. This can be used to check the correctness of a dual construction.
- b. If the primal problem has an unbounded optimum, the dual will always be infeasible.
- c. If an optimal solution to a minimization problem is known and changes to the right-hand-side (b_i) parameters are proposed, a quick way to test if the changes remain feasible (for the primal) is simply to re-evaluate the objective function of the dual corresponding to the original optimal solution. If the dual objective goes below the optimum, one or more constraints of the primal is being violated.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.14 Unimodularity and Special Structures

Notwithstanding the general difficulty of ILP, some problems have special structure that allow an ILP to be solved exactly under its relaxation as an LP. This is significant because LP problems can be solved much more quickly than ILP problems. One way of thinking about such problems is that the constraint matrix \mathbf{A} has a structure that in effect traps the optimal solution onto discrete values if the RHS constants are themselves integer. The relevant mathematical property is called *Total Unimodularity* (TU). Hoffman and Kruskal [HoKr56](#) showed that a matrix \mathbf{A} is TU if and only if:

- For all integral vectors b , the polytope $A \cdot x \leq b$ has integer vertices.
- The problem $\max\{c \cdot x \mid Ax \leq b, x \geq 0\}$, or its dual $\min\{Yb \mid YA \geq C, Y \geq 0\}$, are achieved by integral solution vectors x^*, y^* .

To exploit this property we need to be able to recognize when the constraint matrix is totally unimodular. One definition of TU is:

"A $(n \times m)$ matrix is totally unimodular if the determinant of each of its square submatrices is 1, -1, or 0." [ThSw92](#) (p. 145).

This is not, however, an easily implemented general test criterion. Dubhashi states the closely related but more prescriptive proposition that:

"A matrix is TU iff each collection of its rows can be split into two parts such that the sum of the rows in one part minus the sum of the rows in the second part is a vector with entries -1, 0, or +1." [Dubh96](#)

An even more prescriptive, but only sufficient (not always necessary) for TU is stated by Wolsey [Wols98](#). A matrix \mathbf{A} is unimodular if:

- a_{ij} is an element of $\{-1, 0, 1\}$ for all (i, j) .
- each column contains at most 2 non-zero coefficients.
- there exists a partition (M_1, M_2) of the set M rows of \mathbf{A} such that each column j containing 2 non-zero coefficients satisfies:

Equation 4.21

$$\sum_{\forall i \in M_1} a_{ij} - \sum_{\forall i \in M_2} a_{ij} = 0 \quad \forall j$$

From a practical standpoint the important thing is to know that the TU property exists, not necessarily to implement formal tests for TU before deciding to run a problem as an ILP or LP. Neither the definition of TU, nor Dubhashi's proposition yield an efficient (i.e., polynomial time) test for the TU property, since both themselves involve combinatoric enumeration of all various subsets of the constraint matrix. Moreover, the outcome of such a test can be data dependent, not just structure dependent, so one does not obtain any general assurance that an entire class of problems will always be TU even if a full-blown TU test is implemented on one instance of \mathbf{A} for that problem. Wolsey's criteria is essentially as difficult to test and yet further limits the conditions to 1/0 coefficients. In practice, the simplest way to test for TU is just run the ILP problem as an LP. If an integer solution arises, one has in effect proven the TU of matrix \mathbf{A} , not to mention benefiting from that property at the same time.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

4.15 Network Flow Problems

4.15.1 The Transportation Problem

Some problems, however, satisfy Wolsey's criterion for TU in their canonical structure because they inherently have only 1/0/-1 coefficients and other structural properties. One of the most important problems known to have the TU property in any of its instantiations (i.e., not depending on the data values) is known as the *transportation problem*. In the transportation problem a number of nodes are *net supply points* of a *commodity*, other nodes are *net sinks* or *consumers*, while remaining nodes are *pure trans-shipment points*, and the problem is to assign flows to each arc so that all sinks are satisfied from the available sources with a minimum of total cost for flow across edges. More generally, the transportation problem and its many variations are all special cases of the minimum-cost network flow (MCNF) problem. The special significance of any problem that can be cast in an MCNF framework is that it *is* TU and can be solved as an LP with the *network simplex* method in which all simplex pivot operations involve only additions and subtractions. (See [Wins94](#), p. 446 for the solution method itself.)

Because we will later want to consider single source-destination problems, and the general transportation problem involves multiple source-sinks, we will introduce the general transportation problem here as *multi-source/sink MCNF*:

Multi-source/sink MCNF

Minimize:

Equation 4.22

$$\sum_{\forall (i,j) \in E} x_{ij} \cdot c_{ij}$$

Subject to:

Equation 4.23

$$\sum_{\forall j \neq i | (i,j) \in E} x_{ji} - \sum_{\forall k \neq i | (k,i) \in E} x_{ik} = b_i \quad \forall i \in N$$

Equation 4.24

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in E$$

$$L_{ij} \leq x_{ij} \leq U_{ij} \quad \forall (i,j) \in E$$

where:

N, E are the sets of nodes and edges, respectively.

x_{ij} is the total flow over edge (i,j) .

b_i is the total (net) outflow or supply of the commodity sourced at node i .

c_{ij} is the cost of transporting one unit of flow over edge (i,j) .

L_{ij}, U_{ij} are lower and upper bounds (in the general form) on the flow through any edge.

First some notes on the notation: All edges are *directed* in this type of model. The edge (i,j) has the opposite direction to (j,i) . At node i , the edge (j,i) can only bear incoming flow to node i . Conversely, (i,j) is an outgoing edge at node i directed toward node j . If both (i,j) and (j,i) are present in the graph we may interpret the overall situation as equivalent to bidirectional edges, or in effect undirected edges. So why bother to represent simple bidirectional edges as explicit pairs of opposing unidirectional edges? The answer is that by doing so we allow the expression of *flow conservation* in a whole variety of problem models. Incoming flow and outgoing flow effectively become signed quantities on which we can make important statements such as total incoming flow = total outgoing flow. The practical importance of this is that an entire class of network problems called *arc-flow* solution models are enabled in which routes need not be explicitly represented. Rather, flow variables on each edge are solved for directly, with the routes being only implied. Secondly, because the coefficients of directed flow at each node can only be 1, -1 or 0, it turns out that all network problems with the basic structure of MCNF have total unimodularity!

[Equation 4.23](#) uses explicit indexing of arcs so the reader can appreciate what is implied in subsequent formulations, and in other literature

where this is considerably contracted. The notation $\forall_j \neq i \exists (i,j) \in E$ says "for every node not equal to the current node such that the arc (i,j) exists in the network." In other words, we just mean that the corresponding sum runs over all the edges that exist at each node. But it is understood that edges are still directed. The edge (i,j) supports only outgoing flow at node i . Edge (j,i) , which is separately enumerated in the summation, supports only incoming flow at node i . Once this is understood it is more common to denote this whole indexing specification simply to " j " to specify the members of the summation that is intended. Also note in this and the following formulations that it is nodes that are indexed i or j and arcs are designated by the (i,j) pair of nodes at their ends and all flow quantities on an arc x_{ij} are inherently directional. Due to flow directionality in arc-flow models in general, indexing is on the nodes. In applying this type of model to bidirectional communications contexts, we will need to remember that the directionality was really only a technical requirement of the particular optimization model. In effect we are solving one-way min-cost flow or one-way max-flow etc., then interpreting our one-way solution result as needed in the case of a communications network which usually has symmetric bidirectional communication links and flows. In addition, many models of a different type, called arc-path models, which are nondirectional, will naturally employ indexing on the edges, not on the nodes, so it is important in looking at a model to understand what indexing convention and type of model is intended.

If we look at the MCNF model again, we see the objective is to minimize the total costs for routing flow over edges. Next the constraints provide one instance of [Equation 4.23](#) for each node. It allows any node to be predefined as a producer of the commodity, in which case it has a $b_i > 0$ equal to the amount of the commodity it can produce. Conversely, if the node is a consumer or sink node it has a negative valued b_i indicating its total requirement for the commodity. Any node with $b_i = 0$ is a pure relay point (in this context a distribution warehouse) that neither produces or consumes the commodity in question, but by its existence it adds to the routing options for its distribution. [Equation 4.24](#) merely sets (optional) limits on the total flow over each edge. The solution variables are the amount of flow to route over each (directed) edge.

Beyond its use for the distribution problem it describes in the real world, this problem structure is a canonical form known as the transportation problem and is TU. If the b_i quantities (and edge capacity limits) are integer, the solution variables will be integer as well when solved as an LP. A whole class of useful problems ultimately have the same basic form of **A** matrix and are said to have *network structure* and benefit from solution techniques that take advantage of this.

4.15.2 Two-Terminal MCNF (or Minimum Cost Routing)

As mentioned the pure transportation problem is a form of multi-source/sink MCNF where every node is either a net consumer or producer

of a *single* commodity. The multi-source, multi-sink aspect is, however, not a direct matching to our notions about communication networks: the message that Frank sources to Mary cannot be just as well supplied from an alternate source. The more usual communications context is single-pair communication. Let us therefore look at a single-pair version of the MCNF problem. With a single source and sink, MCNF maps into the problem of minimum-cost routing between nodes:

Two-terminal MCNF

Minimize:

Equation 4.25

$$\sum_{\forall (i,j) \in E} x_{ij} \cdot c_{ij}$$

Subject to:

Equation 4.26

$$\sum_{\forall j \in N} x_{sj} = d_{st}; \quad \sum_{\forall j \in N} x_{jt} = -d_{st}$$

Equation 4.27

$$\sum_{\forall j \in N} x_{ij} = 0 \quad \forall i \in N - \{s, t\}$$

Equation 4.28

$$0 \leq x_{ij} \leq C_{ij} \quad \forall (i,j) \in E$$

where:

N, E, x_{ij}, c_{ij} are as before,

d_{st} is the total demand to be routed between source and destination nodes, and s and t are

C_{ij} is the capacity of arc (i,j) .

The main difference that emerges more clearly than in general MCNF is the aspect of pure trans-shipment at nodes other than s or t . At all nodes other than s , and t there is no net sourcing or sinking of the commodity, which is the total communications demand between nodes s and t alone, so all other nodes act purely as trans-shipment points or relays of the signal flow. Note, however, that [Equation 4.26](#) allows

node s to source its total demand out over all its outgoing arcs. Similarly the destination may absorb incoming flow over any or all of its incoming arcs. The sign reversal for the net demand at the destination creates net flow balance in the formulation. [Equation 4.27](#) is the assertion of pure trans-shipment of flow at all other nodes, also called *flow conservation*. As written it adopts the convention that incoming flow is negative, outgoing flow is positive. Regarding notation, note that a fuller but equivalent statement for the trans-shipment constraints ([Equation 4.27](#)) is:

Equation 4.29

$$\sum_{\forall k \neq i | (k,i) \in E} x_{ik} - \sum_{\forall j \neq i | (i,j) \in E} x_{ji} = 0 \quad \forall i \in N - \{s, t\}$$

[Equation 4.29](#) first sums flow on all outgoing edges at node i , then subtracts the sum of all incoming flows. All flow variables thus stay nonnegative, but obtain a "sign" for the summation of flows in such a constraint by virtue of the direction of the edge they are associated with at the node. This is, again, the key to why we need to go to the directed edge orientation in arc-flow models: it allows us to express flow conservation, and, by doing so, allows us to exploit the special structure of MCNF. The routing solution and flows on each arc obtained can certainly still be interpreted as two-way signal flows such as we would expect in a transport network. Alternately we can observe that if flows are considered directed, then there is always a mirror image LP that will produce the same logical solution with $t \rightarrow s$ flows instead of $s \rightarrow t$ flows as above. The pair together return us to the bidirectional transmission paradigm of transport networks.

4.15.3 Maximum Flow as an MCNF

Maximum flow provides an example of how problems can be changed from their original form to take on an MCNF-like structure. Say we have a network of given edge capacities and we want to send the maximum feasible flow between nodes s and t . Then a straightforward expression of maximum flow through a capacitated network would be:

Max-flow

Maximize:

Equation 4.30

$$z = \sum_{\forall j \in N} x_{sj}$$

Subject to:

Equation 4.31

$$\sum_{\forall j \in N} x_{jt} = -z$$

Equation 4.32

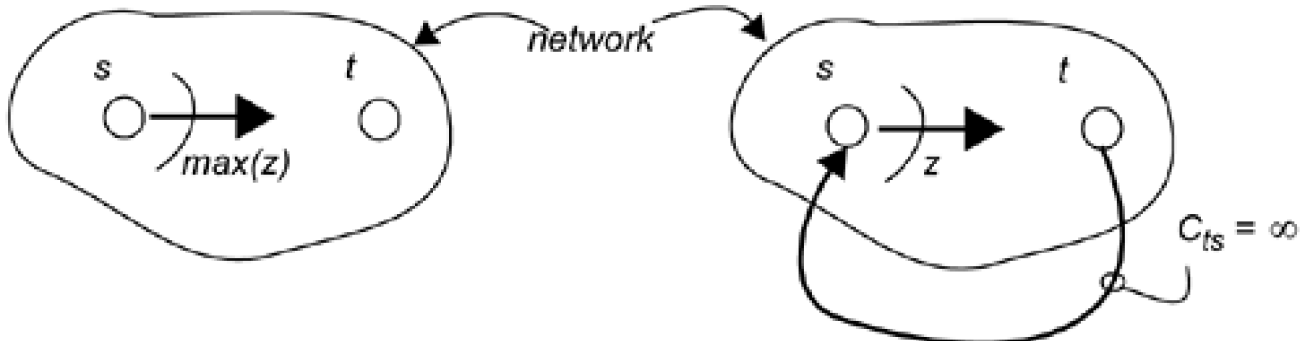
$$\sum_{j \in N} x_{ij} = 0 \quad \forall i \in N - \{j, s, t\}$$

Equation 4.33

$$0 \leq x_{ij} \leq C_{ij} \quad \forall (i, j) \in E$$

In the objective the net flow out of the source node, z , is maximized. The destination node has a corresponding flow-matching constraint (Equation 4.31) and all other nodes have trans-shipment constraints (Equation 4.32). This expression of the problem is not, however, in standard MCNF form. If it was important to solve large integer instances of the problem it might be advantageous to convert the problem into an MCNF form so that TU can be exploited. This can be done for max-flow by addition of an artificial link, illustrated in Figure 4-34, with no capacity limit and unit cost, which supports a direct (but artificial) flow *back* from the sink to source. The maximum flow formulation can then be written as:

Figure 4-34. Converting max-flow to a min-cost network flow problem structure.



Max-flow as MCNF

Minimize:

Equation 4.34

$$z = -x_{st} \cdot 1$$

Subject to:

Equation 4.35

$$\sum_{\forall(i,j) \in E} x_{ij} = 0 \quad \forall i \in N$$

Equation 4.36

$$0 \leq x_{ij} \leq C_{ij} \quad \forall (ij) \in E - \{t, s\}$$

The objective is now in a cost-minimization form on a single flow variable and all other constraints are either pure flow conservation or arc-capacity bounds, all of which adhere to the MCNF canonical structure in [Equation 4.22](#) to [Equation 4.24](#). Note that [Equation 4.36](#) provides capacity constraints for all arcs of the original network but omits any constraint for the artificial return flow arc, thus letting $C_{ts} = \infty$ so that it is indeed the finite arcs of the real network that determine the maximum feasible flow.

If the constraint system above is instantiated for any network it will have the following important properties which make it amenable for solution with a network simplex solver. First every x_{ij} variable in the flow conservation constraint for node i will have a +1 coefficient, a coefficient of -1 in the node j constraint and 0 in all of the other flow conservation constraints—a condition which satisfies the TU criterion. This is an example of how a problem may sometimes be transformed to take advantage of fast solvers for problems with transportation-like structure.

4.15.4 Multi-Commodity Maximum Flow (MCMF)

In the discussions of maximum flow to this point there has been one source node and one target node or only one origin-destination (O-D) pair. The flow associated with one O-D pair is sometimes called a *commodity*. A *multi-commodity* flow problem is one where flows of different kinds co-exist. They may be different types of goods or materials, as the term commodity originally implied, or, in communication networks, the commodities may be signal flows of various status, grades-of-service, or most often simply the flows on different O-D pairs.

In an MCMF type of problem it is some measure of the total flow over all commodities that we are trying to maximize. No commodity can transform into another, however, so we can expect that each commodity-flow may have its *own* conservation equations at nodes other than its own source-sink (or O-D) nodes. However, all the flows compete for use of the edge capacities on the network. The decision about how much of each flow to route over a given edge is coupled with that for every other commodity by the edge capacity constraints they must collectively respect. In other words, they share the capacity of edges mutually, giving rise to the concept of *mutual capacity constraints*. It is this aspect of entanglement between the routing decisions for individual flows that makes the solution of an MCMF routing problem much more difficult than single-commodity max-flow and ksp-type problems treated previously with polynomial-time algorithms.

Broadly, an MCMF problem has structure that is like a number of individual max-flow instances, but which are all cross-coupled through the mutual-capacity constraints. If the capacity of one edge is allocated to the max-flow solution of one commodity it changes the apparent network for solution of the other max-flow subproblems. The general problem is an NP-hard combinatorial allocation problem and requires LP or ILP (or equivalent methods) for its exact solution.

The concept of mutual capacity constraints is illustrated in [Figure 4-35](#) and it is key to understanding MCMF-type problems. It is also a central issue arising later in the problem of path-restorable networks and in distributed restoration protocols that aim to approximate MCMF solutions. It is also the central issue that is overlooked in the proposal of pair-wise independent GMPLS auto reprovisioning as a restoration mechanism (the discussion of [Section 3.5.8](#)). The simple example in [Figure 4-35](#) is devised to show how the outer perimeter capacity is effectively shared in a zero-sum way between the two flow commodities (C-D) and (A-F). Any unit of capacity on edge B-C (or certain others) allocated to the A-F flow is directly subtracted from the total feasible flow for C-D. In contrast we know from [Figure 4-20](#) that the max-flow realization for A-F cannot use capacity on C-D at all, so there is no contention on this edge.

Figure 4-35. Concept of mutual capacity constraints in multi-commodity routing.

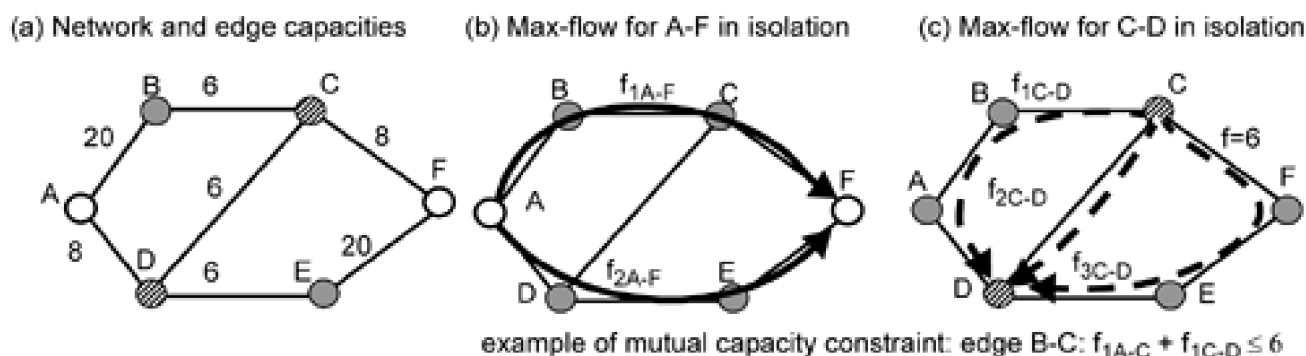


Figure 4-20. The routes corresponding to the max-flow compared to the shortest path flow.

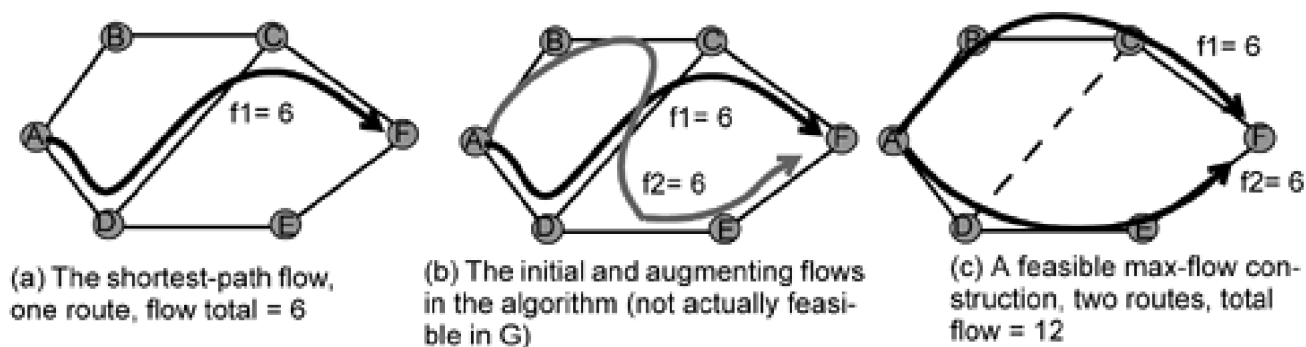


Figure 4-35 also makes it apparent that there can be more than one idea of what it means to maximize flow in a multi-commodity context. If the aim is simply to maximize the total of all commodity flows, then the following solutions would be equally acceptable, both having 18 units of flow:

- a. $f_{1C-D} + f_{2C-D} + f_{3C-D} = 18$, $f_{1A-F} + f_{2A-F} = 0$, or
- b. $f_{1C-D} + f_{2C-D} + f_{3C-D} = 6$, $f_{1A-F} + f_{2A-F} = 12$.

We may have obvious reasons for preferring (b) over (a) (even if the total flow was less) so that no O-D pair is left disconnected. Also, consider some limited aspects, not of MCMF itself, but of the network capacity design problem to support such multi-commodity routing. In Figure 4-35, for instance, it can be seen that if two units of capacity are added to edge C-D, the flow for O-D pair C-D would rise by the same amount. The same is true for C-D flow if the investment of two extra capacity units was made on edge B-C, but in the latter case the flexibility would exist to give the added flow to A-F, or, if A-F restoration arises non-simultaneously with that for C-D, then adding capacity to edge B-C benefits both cases, whereas on edge C-D it benefits only one O-D pair. These are important considerations in restorable network capacity design and distributed restoration protocols. In preparation for these later treatments we now want to define some basic variants of multi-commodity flow problems.

4.15.5 Maximum Sum Multi-Commodity Maximum Flow (MS-MCMF)

MS-MCMF is the most direct generalization of single-commodity max-flow to a multi-commodity framework. The objective is the simple maximization of the total flow over all commodities. The prior designation of a single pair of nodes (s,t) to designate the flow source and sink is now generalized into a set of node pairs each sourcing and sinking a unique commodity flow. We define set K to contain all the node name pairs that identify each flow. In other words, this is our set of commodities. $h \in K$ denotes one such pair, $s(h)$ and $t(h)$ are the

individual source and target nodes for the h^{th} flow. x_{ij}^h denotes the (directed) flow on edge (i,j) for commodity h . Note that no $s(h) = t(h)$. Then:

MS-MCMF

Maximize:

Equation 4.37

$$z = \sum_{\forall h \in K} \sum_{\forall (s(h),j) \in E} x_{s(h),j}^h$$

Subject to:

Equation 4.38

$$\sum_{\forall j \in N} x_{t(h),j}^h = \sum_{\forall j \in N} x_{s(h),j}^h \quad \forall h \in K$$

Equation 4.39

$$\sum_{\forall j \in N} x_{ij}^h = 0 \quad \forall h, \forall i \in N - \{j, s(h), t(h)\}$$

Equation 4.40

$$\sum_{\forall h \in K} x_{ij}^h \leq C_{ij} \quad \forall (ij) \in E$$

Equation 4.41

$$\sum_{\forall (s(h),j) \in E} x_{s(h),j}^h \leq F_h \quad \forall h \in K$$

where:

N, E , are as before,

C_{ij} is the finite capacity of arc i, j .

F_h is a maximum total amount of flow required for commodity h .

The objective maximizes the sum of all flow over all commodity types. It characterizes this through summation over each commodity of the outflow total over edges at the node that is the source for that commodity. Note that if commodities earned different rates of revenue, then corresponding coefficients would weight each source-node flow summation and the result would be a maximum revenue model. The constraints of [Equation 4.38–Equation 4.41](#) are analogous to those of the simple max-flow model above, but now they apply simultaneously for every commodity. The first ([Equation 4.38](#)) is an assertion that total flow sourced = total flow sunk, for each commodity. [Equation 4.39](#) expresses flow conservation at each other node, individually for each commodity, and [Equation 4.40](#) gives edge capacity constraints which now apply to the sum of all commodity flows over each edge. These are the "mutual capacity" constraints that couple each flow to the others by sharing the capacity on each edge so that the total of all flows crossing each edge respects its capacity. In single-commodity max-flow, there is no explicit limit on the amount of flow (and if one is added it is trivial in its effect). In MCMF, however, we may have specific limited requirements or targets for the flow on each commodity. [Equation 4.41](#) expresses these limits on the maximum flow required for each commodity. In the MCMF context such pair-wise commodity limits stop the solution from reaching its maximum by producing a large saturating amount of flow on only one or a few node pairs. If desired, another copy of [Equation 4.41](#) can be added with right-hand side changed to $\geq Gh$ to assert minimum requirements (Gh) on each commodity flow as well.

Note that without the edge capacity constraints ([Equation 4.40](#)), the whole structure would be separable into $|K|$ independent single-commodity max-flow problems and there would be no cross-coupling between instances due to mutual capacity effects. Moreover, if there were also no limits for individual commodity flows, MS-MCMF will "push" the greatest flow between O-D pairs which can most exploit the edge capacities to produce end-to-end flow. Also note that if $K = \{\text{every } i, j \text{ pair}\}$ then with the edge capacity constraints but no commodity flow limits, MS-MCMF is able to escape all mutual capacity coupling issues but still maximize its objective simply by the

solution $x_{s(h),j}^h = C_{s(h),j} \quad \forall h | \{s(h), t(h)\} \in E$ and zero otherwise. That is to say, that if all O-D pairs are admissible commodities and there are no pair-wise flow maximums, then the maximum total flow will correspond to a pattern of perfectly adjacent-node flows that fully saturate the direct edge capacities, and provide zero flow on all O-D pairs that do not correspond directly to an edge in the graph. Of course this is a degenerate solution to the real problem, but it provides us with an upper bound for any type of multi-commodity flow problem:

Equation 4.42

$$\sum_{\text{all commodities } h} \text{flow}_h \leq \sum_{\text{all edges } (i,j) \text{ in } E} \text{capacity}_{i,j}$$

because equality in this expression would only be reached in the conditions just outlined. All other carried flow totals must be less than the sum of all edge capacities present. Of course if commodity limits such as [Equation 4.41](#) are given, then the sum of all such commodity limits is another independent bound on the MS-MCMF maximum value.

4.15.6 Maximum Concurrent MCMF (MConMF)

A form of multi-commodity flow problem that is more like that arising in network restoration is MConMF. Here the interest is in achieving the highest common fractional flow allocation, l , for each commodity, relative to an initial demand quantity on each O-D pair d/h .

MConMF

Maximize: l

Subject to:

Equation 4.43

$$\sum_{\forall j \in N} x_{j, t(h)}^h = \sum_{\forall j \in N} x_{s(h), j}^h = \lambda \cdot d_h \quad \forall h \in K$$

Equation 4.44

$$\sum_{\forall j \in N} x_{ij}^h = 0 \quad \forall h \in K, \forall i \in N - \{j, s(h), t(h)\}$$

Equation 4.45

$$\sum_{\forall h \in K} x_{ij}^h \leq C_{ij} \quad \forall (ij) \in E$$

This formulation does not maximize total flow per se, but maximizes the fraction of flow relative to total requirements d_h on each

commodity. In that sense l can be thought of either as a throughput measure (when $l \leq 1$) or a bandwidth allocation factor ($l \geq 1$) for performance or growth, relative to some required minimum level for each commodity pair. MConMF with $l = 1$ has similarity to the restoration routing problem because at $l = 1$ we are seeking complete flow replacement. The MConMF aspect of maximizing a common relative performance measure at $l < 1$ (in effect achieving the maximum minimum performance for all) is more relevant to circumstances of multiple failures or node loss in a transport network wherein full restorability of all affected O-D pairs may not be possible, but one has an interest in producing the highest overall recovery level with pro-rating of each restoration level to its prefailure level.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

4.16 Techniques for Formulating LP/ILP Problems

One of the practical keys to using LP/ILP solvers is to be able to express the design intent in terms of valid LP or ILP statements. Capturing the problem understanding in this form is not always easy because we cannot use iteration or mix variables in a nonlinear way in expressions, as in computer programming languages. High-level interface tools such as AMPL [FoGa93] give considerable additional flexibility in expressing problem details, however, and are highly recommended. AMPL provides convenient high-level ways of implementing set operations in particular, and implements a certain number of capabilities that bridge between procedural programming languages and the logic of a purely symbolic LP/ILP formulation. There are however some basic aspects of formulating and solving LP/ILP problems that are useful to know.

4.16.1 Mutual Exclusion

When it is desired to express that only one of a group of choices can be made, one can use the following type of constraint structure:

Equation 4.46

$$\delta_1 + \delta_2 + \dots + \delta_n = 1$$

or

$$\delta_1 + \delta_2 + \dots + \delta_n \leq 1$$

where all the δ variables are $\{0,1\}$. In the case of equality to 1, exactly one of the choices will be forced. An example might be when different routes are eligible for routing a given demand flow over a network topology but it is desired to have all such demand between a given node pair flow over a single route. This type of construct is sometimes called a "radio button" constraint because if you "push" a one on one variable, you automatically deselect all others. When the sum must be less than or equal to one, it means that at most one, but possibly none, of the n choices will be made.

4.16.2 Switching

Another form of expressing or forcing mutual exclusion is by creating a false cost coefficient or apparent capacity for one parameter in dependence on the state of another parameter or decision variable. For example, say a network is to be as restorable as possible against any two simultaneous span failures, although perhaps only capacitated so as to assure *full* restorability against any one at a time. Then in the case of any single failure, all surviving spans are available for restoration routing. But if a double span failure arises on spans i and k , say, then one needs some way to express that the restoration paths for span i cannot use span k (since in this scenario span k is also severed), and vice-versa. Here we could use a construct such as:

Equation 4.47

$$f_i^p \leq C_{\infty} \cdot (1 - \delta_{jk}^p)$$

where δ_{jk}^p is an input parameter equal to one if the p^{th} eligible route for restoration span j uses span k , and zero otherwise, and C is an arbitrarily high "capacity constant." The way this works is to say that the flow assignment to p^{th} eligible route for restoration of span i , f_i^p , is effectively unlimited (it will be limited by other constraints) except for flow assignment to routes that cross span k , which are limited to zero.

Another goal might be to ensure that the individual paths of a diversely routed 1+1 APS setup share no spans in common. These requirements can be expressed as:

Equation 4.48

$$f_{i,A} \leq s_i \cdot (1 - \delta_{i,B})$$

where s_i is an amount of available capacity on span i and $f_{i,A}$ is a variable for flow assignment of type A to span i . Normally the effect of the constraint is only to ensure the flow remains below the capacity on span i . But if the wider solution decides to route any B type flow over span i , $\delta_{i,B}=1$ and then the effective capacity for A type flow on span i is thereby zero.

4.16.3 Peak Minimizing

Sometimes the aim of an optimization is to minimize the largest single value of some class of variables within the problem. For instance, in [Chapter 7](#), we will be concerned with minimizing the largest-case oversubscription factor on any span over all failure scenarios. In this kind of problem, the difficulty is that if the objective function is written directly to express the intent we get:

Minimize:

Equation 4.49

$$(\max(z_{i,j}))$$

where $z_{i,j}$ is the measure of concern. The issue is that the max operation is not a linear function as required for an LP objective function. This particular form can, however, be handled by realizing the "maximum" operator in a family of constraints instead of being in the objective function, as follows:

Minimize:

Equation 4.50

$$Z$$

Equation 4.51

$$Z - z_{i,j} \geq 0 \quad \forall (i,j) \in E | i \neq j$$

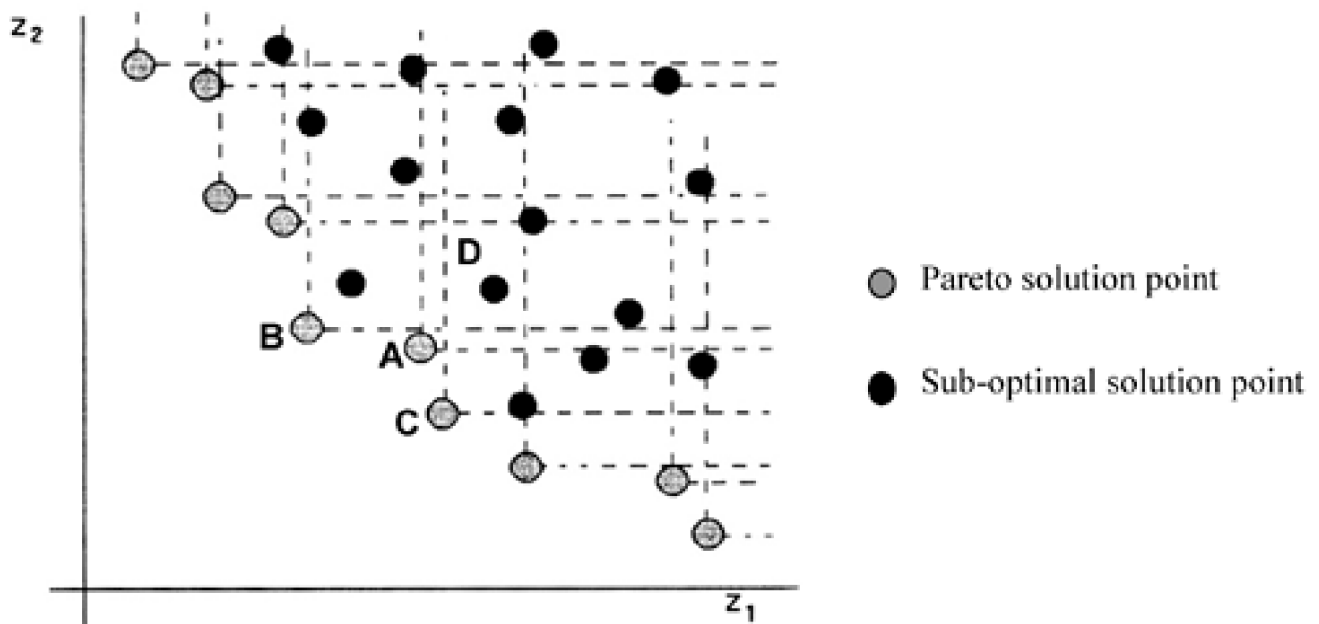
The way this works is that the scalar variable Z now being minimized is like a sheet of plywood laying over all the $i \times j$ space of individual $z_{i,j}$ values. The "sheet" Z is held up by the highest $z_{i,j}$ peak on the $i \times j$ plane. But since Z is being minimized by the objective function an overall solution is found where the highest of the $z_{i,j}$ are pushed down as much as possible by the sheet Z . Thus, one succeeds in minimizing the largest value of a set of variables. The price of changing the formulation (in the example) is a set of $i \times j$ added simple constraints.

4.16.4 Bicriteria Studies and Pareto Optimal Solutions

The wider the scope or realism of a planning problem, the more likely it is that there are actually several somewhat conflicting goals to be optimized, not a single objective. To understand multiple objective optimization, we start by looking at the nature of the solution space. When dealing with bi-objective problems it is accepted that because the two constituent objectives are of different natures or types (for instance cost and availability), they cannot be simply added to combine them into a single objective function without introducing some arbitrary constant to convert one to the other, or both into a common measure of utility. If this can be done, the problem is effectively restored to a single-objective problem. More generally, however, no single conversion constant can be agreed upon. The situation is truly a bi-objective one. In this case the goal becomes discovery of the loci of extreme feasible solutions which can be thought of as a curve revealing the trade-off between one objective and the other.

For bi-objective optimization, there are two objective functions, Z_1 and Z_2 . A solution is said to be "efficient" if any further improvement in one objective necessarily results in a degradation of the other objective. For example in [Figure 4-36](#) if we are minimizing both functions Z_1 and Z_2 then each grey point describes a *Pareto-optimal* solution, and all the grey points together comprise the *Pareto solution set*. As drawn, solution points are discrete but the notion is the same for continuous variables. For the Pareto solution point A, if we want to further minimize on function Z_1 , then we have to use solution point B, in which we will have a lower Z_1 value but a higher Z_2 value. Conversely to further minimize Z_2 from A we have to jump to point C which has a higher Z_1 value. Thus, the Pareto solution is the set of solutions in which it is not possible to improve on Z_1 without worsening Z_2 (and vice versa). Solution point D is not a Pareto (or efficient) solution because we can further minimize either Z_1 or Z_2 without any penalty.

Figure 4-36. A discrete bi-objective solution space and Pareto Solution set.



Methods for solving a bi-objective optimization problem, i.e., finding the Pareto-optimal solution set include:

- a. Defining a blended objective function such as $\{a Z_1 + (1-a) Z_2\}$ and solving one instance of the single blended objective function problem as a is swept from 0 to 1.

- b. Leaving one (only) of either Z_1 or Z_2 as a single objective function and constraining the other by bounds in the constraint system, similar to the manner in which an ILP is solved by bounding the LP solutions, but using the bounds here to permit a systematic revelation of Pareto-optimal points by optimizing one objective while stepping along the bound of the other.
- c. Especially in discrete problem contexts, there may be problem-specific principles which suggest techniques to enumerate Pareto solutions. For instance, in min-cost max-availability routing, a preprocessing step may identify all distinct path options of successively increasing route length. Stepping through these and solving a single objective max-availability problem for each would then be an example of a problem-specific means to reveal the Pareto solution set.

4.16.5 Representing Routes and Graph Topology

Often in using LP/ILP techniques in network design problems, one needs to represent possible routes, paths and demand flows and to be able to express relationships between these. And yet, a specific irregular network topology usually underlies and determines these relationships. The general way to represent such topological structure in the symbolic formulation is through 1/0 indicator parameters. One can think of these as answering an appropriately phrased question: Is there an edge of the graph between nodes i and j ? Does the provisioning route between nodes k and m cross span j ? Does the failure of span i affect the working flow between nodes x and y ? If the answer is yes, the indicator is 1, zero otherwise. This way all required relationships between failures, routes, flows and the graph topology can be abstracted for representation of the problem in a way that is suitable for presentation to a solver such as CPLEX.

Such indicator *parameters* should not be confused with *1/0 variables*. An example of a 1/0 variable might be: Is wavelength k on span j used in the solution? Whereas the number of 1/0 variables in a problem strongly affects its complexity, large arrays of indicator parameters can be part of a formulation without effect on speed because the indicator parameters themselves disappear before run-time. They only define the detailed structure of the problem for that run. If an indicator is 1, then some corresponding variable will be involved in one of the constraints. If the indicator is zero, it is simply absent. So indicator parameters are our main way of expressing network structure and relationships between routes, flows, failures and the graph in which they occur. But they do not actually enter the solver itself. They are like a scaffold or a template that helps to assemble the detailed run-time model. In later chapters we will see numerous examples of the use of indicator parameters to represent network structure and interrelationships that are only determined by the particular network graph that is present.

4.16.6 Modularity of Capacity

Although many planning problems involve integer or even continuous capacity variables, we explained in [Section 1.5.1](#) that the capacity provided by real transmission systems is *modular*. A primary example is the SONET hierarchy in which actual system products are only available at OC-3, 12, 48 and 192. In addition, successive module sizes may exhibit economy-of-scale effects in their cost, i.e., a 128-wavelength system may cost only two to three times that of a 32-channel system (not four times). The overall capacity versus cost function is therefore nonlinear. Because of this it is sometimes thought that these cost-capacity effects cannot be represented in an LP "because the objective function has to be linear." We can, however, represent these effects exactly as follows.

One approach to asserting modularity in a capacity design problem is through constraints that would force the total design capacity on every span to have only one of the allowed capacities, possibly using the radio button approach above. A limitation of this is, however, that the total required capacity on a span, and the true optimum design, may require stacked modules, say two OC-48 systems, instead of a single OC-192. The radio button constraint has a problem in this case because it is limited to saying pick only the one best choice. It cannot admit a choice of stacked modules and mixtures.

A way to deal with this is to recast the capacity design problem from one of "min-cost of capacity subject to capacity modularity constraints" to one of "min-cost of modules placed." In the latter the decision variables become the integer number of modules of each type to place on each span. Prior constraints relating the required span capacity to the flows crossing the span change to involve weighted sums of the capacity each module size presents. The following illustrates the basic logical form:

Minimize:

Equation 4.52

$$\sum_{\text{all spans, } i} \left(\sum_{\text{all module sizes, } k} c_{k,i} \cdot n_{k,i} \right)$$

Subject to:

Equation 4.53

$$\sum_{\forall k} n_{k,i} \cdot m_k \geq w_i + s_i \quad \forall i \in S$$

In the objective function, $n_{k,i}$ is a whole number of modules of the k^{th} size or type placed on span i . $c_{k,i}$ is the cost of a module of type k on span i . Below, some constraint that is concerned with capacitation of the design to meet its flows for both working (w_i) and restoration (s_i) purposes then relates the capacity sum of modules placed on each span i to the requirement through m_k parameters which give the number of capacity units associated with each of the module types indexed by k . Note that [Equation 4.53](#) is in addition to other constraints that may be in the initial model.

4.16.7 Implicit Representation of Functionality

Sometimes the expression of our functional understanding about a certain system manifests itself in an ILP as the simple *absence* of certain constraining relationships, or a lower dimensionality of the variables than would otherwise be needed. An example is representation of the function of wavelength conversion (WC) at nodes of an optical network. The WC capability requires an additional cost relative to the non-WC case in the real equipment. If WC is present in the real equipment, however, it *removes* any constraints in a corresponding ILP model that would otherwise have to deal with assuring wavelength coherence through switching nodes. In the specific example we obtain a lower-dimensional "capacity-only" model because of the capability to switch any wavelength to any other as needed. This is in contrast to a corresponding ILP model where the equipment cannot perform WC: then we need explicit additional variables and constraints to assert end-to-end wavelength consistency. Thus while WC takes equipment in the real world, it is actually the *absence* of any constraints or variables pertaining to wavelength assignment that implies WC in a network optimization problem! In general it is good to be aware and watch for other contexts where absence of explicit considerations in the ILP problem actually implies elaborate special considerations in the real-world, and vice-versa.

4.16.8 Tips for Getting the Solutions

Once all stakeholders agree that a certain LP/ILP model meets the design intent, problem instances have to be solved with real data sets. For ILPs in general, including MIP and 1/0 MIPs, there are a number of tactics that can also be used at the solution phase. The user manuals for a solver such as CPLEX will give more detailed information on these but we can summarize some of the principles to be aware of here.

Bounds and Added Valid Constraints

First, on difficult ILP problems, consider if there are any upper or lower bounds on the objective function that might be derived from first principles or obtained separately from a heuristic or simplified problem model. Such bounds can be added as explicit additional constraints in the model and may help the solution time by reducing the branch and bound search space. Similarly, are there additional constraining relationships between variables or bounds on subsets of variables that are not a required part of the model, but represent valid extra knowledge the user has of the problem? In some cases "added valid knowledge" constraints can also reduce the ILP search space. In general, however, added valid knowledge constraints need to be binding in the solution to be of much help. Otherwise the added constraints can increase the basic LP node solution time more than the benefit received from reducing the number of ILP nodes to be searched.

Relaxations and Unimodularity

Often there are certain variables that may be relaxed to help speed up solution. By relaxed we mean they are allowed to be considered as real valued in the solver, although they are strictly integer or 1/0. Well chosen relaxations can sometimes be shown to have no net effect on the objective value and/or may be easily "repairable" if fractional in the solution. A good example is the practice of relaxing working and/or restoration flows within an environment of strictly integer demands and capacities in a number of design problems that follow. The integer nature of the demand and capacities tend to constrain the real-valued flows to coming out as integer values in the solution and if they do come out fractionally, a simple repair process can restore us to an equivalent fully integer solution. (This specific example is discussed further in [Chapter 5](#).) For some purposes, the detailed integer construction may not even be necessary, just the objective value, so it is important to keep this in mind as well, especially if a given relaxation affects only the feasibility of constructing the solution but not its objective value. Generally, however, relaxation of pure 1/0 variables is not effective as it greatly affects the solution value and leaves an "unrepairable" relaxed solution.

The general point is to look for and test various relaxations that may be effective in reducing run times while not highly distorting to the solution. Of course total unimodularity may permit complete relaxation so it is helpful to know of problems with such special structure, and consider if the problem at hand can be transformed to match such structure. In general, however, a prior analytical test for the TU property is not worthwhile. If one has lucked into TU, simply running the problem as an LP will show it. Otherwise not much time is lost by trying the whole problem as an LP first. Commercial branch-and-bound ILP solvers typically do this as a first step in any case.

Tolerances and Priorities

Another important thing to know about solving ILPs is that reaching a full and formal termination of the problem with a solver such as CPLEX actually constitutes a mathematical proof of the optimality of the corresponding solution. For many purposes this will not actually be required, however. This is why solvers such as CPLEX provide for a user-defined MIP gap and run-time limits. The MIP gap tells the solver when it can stop in terms of solution quality (within a certain percentage of the best LP relaxation solution value yet seen in the branch and bound search). If a problem terminates with a 5% gap it means that solution is at most 5% from optimal. With run-time limits explicit CPU time limits can also be set after which the best-feasible solution is reported as the result. The latter can be especially useful for iterating ILP subproblems within a larger overall heuristic framework.

Heuristics Using LP/ILP

A final point about using LP/ILP methods to solve network design problems in general is not to get hung up on the optimality issue. An ILP solver can well be used just as a general purpose engine to produce good designs under a stipulated time limit. Viewed this way a whole range of relatively overlooked LP-related heuristic strategies and ideas are enabled that can often outperform pure heuristics for the same problem, especially if code development time and effort are considered. Three broad classes of such heuristics are ones that:

- a. Use an optimal solution model but compromise on the completeness of the data sets provided. *Example:* capacity design with limited numbers of preselected eligible route candidates instead of all-distinct routes. This can be especially effective where some separate principle or knowledge can be brought to bear on how to eliminate the least "important" data from the problem.

- b. Use repeated solution of an optimal model but compromise on the run time allowed for each iteration and/or use relaxations, within some wider heuristic search strategy. *Example:* In topology optimization one might solve a time-limited subproblem to approximately evaluate the routing and capacity cost required as the graph topology options are being searched.
- c. Break the complete design or optimization problem into a part (or parts) that are well-handled by an ILP solver and other parts that are best handled by a procedural algorithm or ad hoc heuristic. *Example:* In planning for uncertain future demands, an ILP might handle design for assumed-known scenarios and a higher level search heuristic generates test-case demand patterns to observe evolution of the overall capacity envelope required to future-proof the design.

The importance of these approaches is that they represent a middle ground in approaching network planning problems in which we neither rule out LP/ILP methods nor do we use them only with an expectation of optimal problem solutions. This middle ground is one where LP/ILP models are themselves only building blocks of larger overall heuristic approaches. Sometimes such heuristics can be put together and tested in a day for testing ideas in research, and inherently offer an easy speed-quality trade-off whereas pure heuristics may take months of development and show uncertain accuracy.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.17 Lagrangean Techniques

4.17.1 Vector-Matrix Notation for an LP Problem

Often when we are developing an LP or ILP model for some problem we use indexed summations of expressions defined on the problem variables for the objective functions and constraints. Usually there are important ideas involved regarding which specific variable indices are included in each expression, so summation operators with explicit indexing are usually the most understandable form to use. In the end, however, when all the details are pinned down, the result is expressible in a compact vector-matrix form. For example, a generic minimization problem is:

Equation 4.54

$$\min\{c \cdot x \mid A \cdot x \leq b, x \geq 0\}$$

This compact form makes use of matrix-vector multiplication operations to imply the whole problem structure. The compact vector-matrix form is useful for generic arguments or reasoning about LPs in general, in contexts such as duality and Lagrangean relaxation for instance, which follow. The solution vector, x , is understood to be a column vector of length n variables and c a row vector of cost coefficients implicitly also of length n . A is understood to be an $n \times m$ matrix, the *constraint matrix*. The number of constraint relationships, m , may generally be much larger than the number of variables n . The Ax product b is a vector of constants (*parameters* of the problem) also of length m . The inequality is understood to apply individually at every row level. Vector multiplication can also displace use of a summation sign. In some literature for instance $\min\{\hat{1} \cdot x\}$ is written where $\hat{1}$ is defined as the "all ones vector" $=\{1, 1, \dots, (n) \dots 1, 1\}$. Then of course:

Equation 4.55

$$\{\hat{1} \cdot x\} = [1 \ \dots \ 1] \cdot \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} = \sum_{i=1 \dots n} x_i$$

4.17.2 Lagrange Multipliers

In a problem with one or more *equality* constraints it is possible to remove them by moving them into the objective function and defining an additional variable, called a *Lagrange Multiplier*, associated with the removed constraint. The key concept is that conformance with an equality constraint can be asserted through an added term in the objective function which acts like a penalty term for not satisfying equality in the constraint. To illustrate the basic principle consider an objective function and one (of possibly several) equality type

constraints we would like to remove:

Minimize:

Equation 4.56

$$z = f(x_1, x_2, \dots, x_n)$$

Subject to:

Equation 4.57

$$g_i(x_1, x_2, \dots, x_n) = b_i \quad i = 1, 2, \dots, m$$

Then the basic concept is only to observe that, at optimality, the following objective function alone will have the same full effect as above with the separate constraint:

Minimize:

Equation 4.58

$$z = f(\hat{x}) + (g(\hat{x}) - b)$$

where \hat{x} is the solution vector. Functional inspection shows how this works. If trying to minimize z , choice of an \hat{x} that gives $g(\hat{x}) = b$ (if feasible) is unavoidable because when $g(\hat{x}) = b$ the second term contributes zero extra value to the objective value z . So simply minimization pressure alone will force compliance with the prior explicit constraint. On the other hand if \hat{x} gets into a region where $g(\hat{x}) < b$, then there is an issue because this will contribute an artificial negative amount to the objective function, which could run away on us under minimization unless appropriately constrained.

Further, because inequality constraints can be converted to an equality constraint with the addition of a slack or surplus variable, there is considerable range for trading constraints for added variables. Ultimately any LP with n variables and m constraints can therefore be converted by this method into a completely unconstrained global optimization problem with (at most) $n+2m$ variables. In practice, the much higher dimensionality of the completely unconstrained problem version may overcome the computational advantage of reducing the number of constraints so judgement is required in selecting which if any constraint systems may be advantageously converted by such use of Lagrangean multipliers.

Because all the constraints are equalities (recall we may have added slack variables to convert from inequalities), it follows that at the optimum solution vector x , every $g_i(x) - b_i = 0$. Therefore, notionally, (under minimization), instead of having the explicit constraint, we have a kind of penalty function added to the objective function to bias it toward compliance with the constraint. Any time the solution was away from the condition of the constraint, the absolute value $|g_i(x) - b_i|$, which is a measure of the corresponding extent of constraint *unsatisfaction*, would contribute weight to the objective function. This would not upset the location of the optimum solution in the original problem because at optimality each such penalty function would be contributing zero penalty value to the objective. The absolute value of the $g_i(x) - b_i$ difference is involved, however, to avoid spuriously lowering the objective function below optimum when the solution is feasible and the constraint is not binding. In practice, we cannot use an absolute value operator so we use Lagrange multipliers which themselves become variables to be solved for, but as variables also cannot become negative. More formally, this notion of converting an equality constraint to a penalty function added to the objective is known as forming "the Lagrangean":

Equation 4.59

$$L(x_1, x_2, \dots, x_n, \lambda_1, \lambda_2, \dots, \lambda_m) = f(x_1, x_2, \dots, x_n) - \sum_{i=1}^m \lambda_i \cdot (g_i(x_1, x_2, \dots, x_n) - b_i)$$

In the completely general case we obtain a nonlinear unconstrained global optimization problem in $n+m$ variables. It is nonlinear because, even if $f(\mathbf{x})$ and $g(\mathbf{x})$ are linear functions, terms of $O(\|\mathbf{x}\|^2)$ arise in the right-most term, where λ and \mathbf{x} are both vectors of variables. The Lagrangean objective function is therefore quadratic in nature, suitable for solution by a steepest ascent or descent type of gradient maximization or minimization. If, however, for a minimization problem, $f(\mathbf{x})$ is any convex^[5] function and each $g(\mathbf{x})$ is a linear function, then any solution $\{\mathbf{x}, \lambda\}$ that minimizes $L(\{\mathbf{x}, \lambda\})$ ^[6] is also an optimum solution for the original constrained problem formulation. The corresponding statement for maximization requires only that $f(\mathbf{x})$ be a concave function.

^[5] In a *convex set* S , all points on a line joining any two points $\{x, y\} \in S$ are also contained in S . $f()$ is a *convex function* if all points on a line joining coordinate points $\{x, f(x)\}, \{y, f(y)\}$ on the function fall above the function value at all intermediate points, i.e., $f(ax + (1-a)y) \leq af(x) + (1-a)f(y)$ for all $0 \leq a \leq 1$. For a *concave function* all points on a line between points on the function are below the function value. Confusion can easily arise because both convex and concave *functions* are defined on *convex sets*.

^[6] More formally, all partial derivatives of $L()$ with respect to each individual x , and λ variable must all be simultaneously equal to zero.

Thus, we have an additional technique to be considered for addressing large or difficult LP problems (or for solving LP subproblems in ILP). Because the constraint functions $g(\mathbf{x})$ are linear they cannot cause any local minima to arise when added to an already convex $f(\mathbf{x})$. Thus, there remains a unique minimum and, for the set of signs on the λ vector that correspond to the optimum, its position is unchanged. For any given λ vector only the shape of the polytope has been distorted by the linear penalty function planes. Note that if one has a constrained continuous optimization problem where *all* constraints are equalities, then it is possible to solve completely for the optimal solution by solving the system of $n+m$ linear equations in $n+m$ unknowns arising from the conditions of optimality:

Equation 4.60

$$\frac{\partial}{\partial x_i} L(\mathbf{x}, \lambda) = 0 \quad \forall i \in 1 \dots n$$

Equation 4.61

$$\frac{\partial}{\partial \lambda_i} L(\mathbf{x}, \lambda) = 0 \quad \forall i \in 1 \dots m$$

This inherently requires a solution for all the Lagrange multipliers as well as the desired variables of the original problem. The numerical values of the λ_i that are obtained have various meanings in terms of sensitivity analysis. At the optimum itself their values are (from one point of view) rather unimportant as all of them are then multiplying a zero quantity, adding nothing to the objective function. In this view they are just a set of artificial or internal variables used as a device to guide the unconstrained optimum search to a minimum in the penalty-function-laden space that corresponds to the constrained optimum. Lagrange multipliers therefore also suggest a way to structure the search. For instance if one had the problem:

Minimize:

Equation 4.62

$$z = f(x_1, x_2, x_3)$$

Subject to:

Equation 4.63

$$g_1(x_1, x_2, x_3) = b_1$$

Equation 4.64

$$g_2(x_1, x_2, x_3) \geq b_2$$

it could be solved by running a sequence of LPs of the form:

Minimize:

Equation 4.65

$$z(\lambda_1) = f(x_1, x_2, x_3) + \lambda_1 \cdot (b_1 - g_1(x_1, x_2, x_3))$$

Subject to:

Equation 4.66

$$g_2(x_1, x_2, x_3) \geq b_2$$

where λ_1 is swept over a search range and selecting the superior solution which is feasible under [Equation 4.66](#).

4.17.3 Extension to Inequalities

The usefulness of the Lagrangean method is obviously higher if it can apply to inequality constraints as well as equalities. One way of extending the method to deal with inequalities is to recognize that at the constrained optimum solution some number k of the m inequality constraints present will be *active*. An inequality constraint is active in the solution if, at optimality, it is satisfied in a numerical state of equality. In others, only some k out of m constraints will be binding on the solution. For the given parameters, the others will be redundant.

This suggests a way to structure the search of the solution space, particularly for cases where a good, but possibly not optimal, solution is required. First, one solves the unconstrained problem \min (or \max) $z = f(x)$. If this satisfies all constraints we are finished. Otherwise

we set $k = 1$ and *activate* each inequality constraint individually by converting the corresponding $g_i(x) \leq b_i$ statement to $g_i(x) = b_i$ and dropping the other constraints entirely. This produces m instances of a single-multiplier Lagrangean function for *unconstrained* optimization. Any feasible solutions found among the m problems are local optima. Next $k = 2$ is set and all pairs of inequality constraints

are activated by asserting equality and dropping the remaining constraints. This produces $\binom{m}{2}$ (" m choose 2") two-multiplier Lagrangean functions for *unconstrained* optimization, and so on. On a problem with a unique constrained optimum, the best of all feasible optima found in this way is a global optimum [Taha87] (pp.767).

Obviously, however, the number of subproblems generated by this method can become large with many inequality constraints. This method could be used however, in conjunction with a random sampling strategy that would try to saturate or broadly explore the space of all activated constraint combinations or with a genetic algorithm or "tabu" search (meta-heuristic methods that follow) that would try to develop a set of the most important constraint combinations to retain in an activated state.

4.17.4 Lagrangean Relaxation and the Lagrangean Dual

Duality theory and the concept of replacing an equality constraint (or an inequality plus a slack variable) with a Lagrange multiplier come together in the method of *Lagrangean Relaxation (LR)*. The procedure is based on recognition that many NP-hard problems (including ILP) have a structure that comprises a set of subproblems that are relatively easy to solve individually, but which are complicated by a set of "side constraints" that creates linkages between each of the subproblems. We have already seen a case in point: in MCMF there is a structure of individual max-flow subproblems for each O-D pair, which would be solvable in $O(n^3)$ time on their own. But the mutual capacity constraints link all the individual max-flow subproblems through the need that they collectively respect the capacity of each edge, making the problem of exponential complexity.

The idea in LR is to identify these complicating constraints and *dualize* them in the sense that each constraint of the primal generates a Lagrangean multiplier variable in the objective function. This produces a Lagrangean problem that is easier to solve to completion than the original problem or at least provides a bound (or even the optimum objective function value) for the original problem. To develop the method let us consider a generic minimization problem and, for compactness, adopt the vector-matrix notation for this discussion. The generic problem is:

Minimize:

Equation 4.67

$$v = c \cdot x$$

Subject to:

Equation 4.68

$$\begin{aligned} A \cdot x &\leq b \\ D \cdot x &\leq e \\ x &\geq 0 \end{aligned}$$

where x is a column vector of the solution variables and may be integer or $\{1,0\}$ in nature c is a row vector of cost coefficients. A and D are matrices, and b and e are column vectors of the same dimension, collectively generating the constraint system of inequalities. Suppose

that $A \cdot x \leq b$ are the complicating constraints. These are "relaxed" by referring them into the objective function through a vector of Lagrange multipliers, leading to the Lagrangean Relaxation of the problem:

Minimize:

Equation 4.69

$$v_\lambda = c \cdot x + \lambda \cdot (A \cdot x - b)$$

Subject to:

Equation 4.70

$$\begin{aligned} D \cdot x &\leq e \\ x &\geq 0 \end{aligned}$$

More than one subset of constraints can be correspondingly dualized by adding another family of Lagrange multipliers for each such set in forming the Lagrangean function that [Equation 4.69](#) represents. Note that because we have $A \cdot x \leq b$, all elements of λ must be ≥ 0 for v_λ to approach the optimum from a lower bounding sense, i.e., better than feasible, because the $A \cdot x - b$ quantities are then all negative until the dualized constraint term $\lambda \cdot (A \cdot x - b)$ reaches equality on the binding constraints (the non-binding constraints will have $= 0$ through the remaining considerations to follow). [Equation 4.69](#) is not the full dual of the problem as previously defined for the LP dual above, but it is a dual-like formulation in the sense that it approaches the optimum from better-than feasible objective function values and only reaches feasibility (and optimality) when all terms where $\lambda > 0$ satisfy feasibility *at equality*, i.e., have $A \cdot x = b$, and any other terms having $A \cdot x \leq b$ at optimality correspond to non-binding constraints and have $\lambda = 0$. Conversely if the complicating constraint system that is being dualized is of the form $A \cdot x \geq b$, then we have to have $\lambda \leq 0$ so the same $v_\lambda \leq v$ dual-domain approach to optimality pertains. Finally if the dualized constraints are equalities $A \cdot x = b$, then all λ are unrestricted in sign. Of course this all leads up to the key question of how to determine the vector λ so that problem $\min v_\lambda$ does indeed have the same optimum solution as the original problem $\min v$. Whereas [Equation 4.69](#) is called the *Lagrangean Relaxation*, the *Lagrangean Dual* now comes into play.

Following the interpretation that (with the correct λ vector), v_λ approaches the optimum "from below," (i.e., from better-than-optimal, infeasible states) the concept of duality tells us that when $\lambda \cdot (A \cdot x - b)$ is just at its *maximum*, it just meets the *minimal* (optimal) solution of the corresponding original primal problem statement. In fact the best choice for λ in the LR comes from solving the Lagrangean Dual with respect to the constraints $A \cdot x \leq b$. This is:

Equation 4.71

$$\max_{\lambda \geq 0} w$$

Subject to:

Equation 4.72

$$w \leq c \cdot x + \lambda \cdot (A \cdot x - b) \quad \forall x \in X = \{x | (D \cdot x \leq e)\}$$

Note that this dual is in the form of an LP in variables λ with a large number of constraints, one for each combination of variables from the primal that is feasible under the $D \times \sum e$ constraints. At first this may not look like an improvement in terms of problem complexity, if we assume normal LP solving methods. However, it follows from the form of the large constraint system [Equation 4.72](#) that w is the lower envelope of a finite family of linear functions in the λ -vector space. It has the properties of continuousness and concavity but is not differentiable as would be required to apply a normal gradient search method. The method of sub-gradient optimization is therefore used.

A vector g is called a *sub-gradient* of w at λ if: $w(\lambda) \leq w(\bar{\lambda}) + (\lambda - \bar{\lambda}) \cdot g \quad \forall \lambda$. In other words, it is any vector (magnitude and direction) in the λ -space which does not exceed $(w(\bar{\lambda}) - w(\lambda)) / (\lambda - \bar{\lambda})$ (a first order estimator of the local gradient) in any direction. In the subgradient method, an initial vector λ^0 starts the generation of a sequence $\bar{\lambda}^k$ where m is the length of the λ vector and each element of it is indexed by i . g^k is any sub-gradient selected at $\lambda = \lambda^k$ and $\alpha_k > 0$ is the step size. A commonly used step size is $\alpha_k = \alpha_k \cdot (\bar{w} - w(\lambda^k)) / \|g^k\|^2$ where $0 < \alpha_k < 2$ is a step-size parameter and \bar{w} is an upper bound usually obtained by applying some heuristic to provide a reasonably good feasible solution to [Equation 4.71](#) and [Equation 4.72](#) [\[Galv93\]](#).

4.17.5 Complexity of LP and ILP

Where do LP and ILP fit into this scheme of things complexity-wise? The historic simplex algorithm for LP (by G.B. Dantzig in 1947) is still the most widely used technique for solving LPs. Although exponential in its worst-case time-complexity ($O(2^n)$ [\[Wins94\]](#)) it is found in practice that the simplex algorithm typically finds the vertex with highest objective value in about as many pivot moves as there are constraints in the problem. LP simplex is thus a case where average execution times are dramatically better than the theoretical time complexity. In practice, LP code is also highly optimized, so typical problems solvable by LPs tend to exhibit low-order polynomial run time increases with problem size. The more recent method of Karmakar for LPs has been shown to be a true polynomial-time method [\[Wins94\]](#) (p.182). Integer programming, however, is NP-hard and this is manifestly evident in actual run-times. As can be appreciated from the branch and bound algorithm, all combinations of rounding each variable up and down may need to be checked—so worst-case ILP run times go as $O(2^n)$ where n is the number of variables.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

4.18 Other Combinatorial Optimization Methods: Meta-Heuristics

LP and ILP are powerful and often preferred tools for optimization in both production and research. But it is still relatively easy to pose a problem that would take more than the life of the universe to solve exactly with an ILP. Worse still are certain problems that may take a long time for the ILP to even reach feasibility, hampering the direct use of time-limited ILP runs as a ready form of heuristic.

Purpose-specific heuristic methods for all or part of the problem may then be warranted. Such heuristics may also embody a level of design detail that goes beyond that which is practical to represent in ILP models but is helpful in ordering, provisioning, and costing the design in detail. For instance, the exact location of every manhole from a geomatic database can easily be brought into the detailed costing calculation for a new cable run by determining the section lengths to order and the number of splicings required, but this level of detail may exceed the practical ability to represent it in an ILP. On any given problem, experts in the area may come up with problem-specific heuristics based on aspects of the *particular* problem only. This possibility should always be kept in mind. There are, however at least three well developed *meta-heuristic* approaches that can be considered for almost any problem: These are:

- Simulated annealing (SA)
- Genetic algorithms (GA)
- Tabu search (TS)

A *meta*-heuristic is a basic framework and a high level process or strategy for optimization that requires only some means of defining and evaluating an objective function from a specification of the current design state with no other dependence on particular problem specifics. In other words, meta-heuristics are optimization engines in the same sense that an LP/ILP solver is; if you can express your problem in their framework, they can be applied. *Meta*-heuristics can not, however, assure optimal results, but are often easily customized and can cope with highly detailed representations of the optimization problem. For more on meta-heuristic concepts and techniques in general, see [\[RiHa01\]](#).

4.18.1 Simulated Annealing

Simulated annealing (SA) [\[KiGe83\]](#) is inspired by the idea that strain minimization at the interfaces between crystal grains in the slow cooling of a metal is somehow akin to the nature of combinatorial optimization problems. SA involves making random changes to the design in which the extent of the random change is slowly reduced or "cooled." Improvements are kept but sometimes changes that worsen the design are also kept to escape local minima. Simulated annealing can also be thought of as a search diversification tactic that can be added to any existing improvement-seeking (hill climbing or pure descent) method that involves some aspect of trial and error or random search in generating its moves. The limitation of pure steepest-descent /ascent algorithms is that they get stuck in the first local minimum/maximum that they encounter. A common approach when faced with this is to generate numerous random starting points looking for the lowest of all local minima found from each restart. This works poorly, however, in the presence of fine-scale ruggedness in the objective function, as is characteristic of discrete combinatorial optimization problems, especially if the computation of a local gradient estimate is complex. SA often performs better than random-start sampling of the objective function by effectively searching or sampling the surface at a range of scales, starting very wide, then narrowing down to the fine scale only near the end of the "cooling schedule." The generic SA procedure (written for a maximization context) is:

```
Generic Simulated Annealing Algorithm {  
   $t := 0$   
  temperature  $T(0) := T_{max}$   
  define a cooling function or schedule  $T(t)$   
  initialize underlying search or move-generating procedure  
  while  $T(t) > T_{min}$  and not done{
```

```
for  $n = 1$  to  $N_C$  {
  make a move
  calculate  $DE$  (resultant change in obj. value)
  if  $DE > 0$  {accept the move
    else if  $\text{rand}() < P_{\text{accept}}[T(t), DE]$  {accept the move
    else reject the move }}
   $t := t + 1$  (or often:  $T(t) := a T(t)$ )
  if termination criteria met (time, obj. value, etc.) {
    done := true}}
```

The cooling schedule $T(t)$ is a decreasing function of t which controls the probability that a step backwards in the objective function is accepted as computation time progresses. The lower the a constant, the faster the cooling. At the start almost any move is accepted but with time only smaller backward steps are accepted, "cooling" the random jumping around of the process on the surface of the objective function in both frequency and magnitude of jumps. Eventually as $T(t)$ becomes small, only improvements are accepted. N_C is the number of moves or configuration changes to consider at each temperature step. Improvements are always accepted, but sometimes a move that worsens the objective function is also accepted. $\text{rand}()$ is a value from a uniform random number generator on $[0,1]$. A poor move is thus accepted with probability P_{accept} which depends on the magnitude of the objective change and the current temperature according to:

Equation 4.73

$$P_{\text{accept}}(t, \Delta E) = e^{-\Delta E/kT(t)}$$

where k is Boltzmann's constant (from thermodynamics). This is called the *Metropolis criterion* for deciding whether the move is accepted or not. If the time spent at each temperature is enough to allow a statistical mechanical equilibrium, (here, if N_C is high enough) the theory is that this results in a statistical distribution of solutions at each temperature that is distributed according to a Boltzmann probability function:

Equation 4.74

$$p(E_i) = e^{-E_i/kT} / \left(\sum_{\forall i} E_i \right)$$

where E_i is a discrete objective function value (or the energy of a system state in the thermal analogy).

Thus, early on when the temperature is high, the probability of accepting a large move in a bad direction is at its highest. The idea is that the objective function is thus being sampled widely, at a coarse scale, at the start. Because all improvements are kept, however, as the temperature drops the effect is to localize the process into the largest scale minima or basin in the landscape. The process still jumps around but is less likely to jump out of this basin entirely. Within that broadest basin, the process similarly gravitates to the next most dominant basin on the local scale, and so on. It is this cloud-like state of the process dwelling over and sampling the landscape, and its slow condensation into a progressively smaller sub-region, repeated at each scale as temperature drops, that lets SA find globally good local minima by the end of its cooling schedule. This can do better than random starts of a pure descent routine because the latter always searches at the finest scale of the landscape whereas SA is first sensing the large-scale structure, settling into the most globally dominant depression, sensing the structure at that scale, settling into the dominant basin at the next regional scale, and so on. Only at $T=0$ is SA strictly searching in a fully deterministic mode at the finest scale and is equivalent to a pure ascent/descent process.

In practice good performance with SA depends primarily on the choice of cooling schedule and the underlying move generation process. These two issues are linked in achieving a good overall search result in that the inherent randomness of the basic search method and the temperature must be calibrated together so as to produce a starting point with sufficiently wide search diversity, but not spend a lot of time stepping through temperatures where the search diversity is essentially saturated. One approach is to use a preliminary set of trials where $T = 1$ is initially assigned and then increased until an acceptance rate (of bad moves) of over 90% is achieved in $N_C = 100$ trial moves. In effect this provides a calibrated starting point defining an initial temperature $T(0) = T_{\text{max}}$ with respect to the corresponding move generating process (which is always problem specific) at which the search will be almost completely random over the objective surface. $T(t) = a @ T(t-1)$ is then used to generate a geometric progression for the cooling schedule. A typical value of a is 0.9.

4.18.2 Genetic Algorithms

Genetic algorithms (also called evolutionary computing or evolutionary algorithms) were initially developed by John Holland [Holl75] and are inspired by ideas about the processes found in evolution, especially the concept of natural selection of offspring. A genetic algorithm (GA) uses an encoding scheme (a "schemata") which represents a potential solution as a string formed from some symbol set and a *fitness function* by which it can assign a score to the solution that the string represents. The schemata is in effect a specification from which the individual can be fully constructed in all detail. Such reconstruction from the schemata may involve algorithms or procedures that are understood to apply, but which need not appear directly as part of the GA strategy itself. For instance, if the edges of a graph are specified, the performance or cost (i.e., the fitness in general) of this topology may depend on a defined routing algorithm that is understood to apply when using any specified set of edges to route a given demand matrix. The string representing a particular solution is also called the *genome* or the *chromosome* of that individual solution. As a simple example one can imagine a chromosome which specifies the amount of spare capacity on each span of a mesh network. The fitness function could be the total cost of such capacity or the restorability level that such a disposition of spare capacity provides, or a combination of both considerations. The GA generates an initial population of solutions from randomly generated strings and evaluates the fitness of each individual in the population. This initial population is operated upon by combining pairs of strings to a new population.

Pairs of strings are chosen probabilistically for this combining operation, with a probability that is proportionate to each string's fitness relative to the average fitness of the population. Typically pairings are generated until the population size would be doubled and then the least-fit individuals from both the prior generation and the current pairings are killed off, keeping the total population of constant size. This process of selection for combining, reproduction, and killing off excess low-fitness individuals is repeated to produce successive populations. The GA can terminate either when any single individual arises that meets a certain target fitness level, after a certain number of generations have passed, or when the population converges by some measure (where all individuals are the same or have the nearly the same fitness). The basic iterative procedure is as follows:

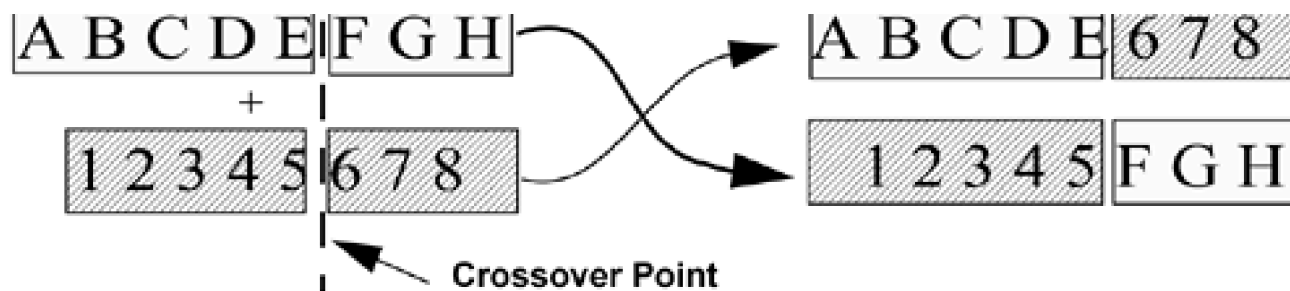
Generic Genetic Algorithm {

```
t := 0;
P(t) := a random population of individuals (with size N);
evaluate(P(t)); (evaluate fitness of initial population)
while not done {
  t := t + 1;
  P(t) := selectparents(P(t));(choose parent subpopulation)
  combine(P(t)); (combine genes of selected parents)
  mutate(P(t)); (perturb mated population stochastically)
  evaluate(P(t))' (evaluate fitness of new population)
  P(t) := survive(P(t),P(t)); (select survivors on fitness)
  if termination criteria met (time, fitness, etc.) {
    done := true}}}
```

Crossover is the most common method for the generic process called *combine* above. Crossover effectively mates individuals from a surviving population to produce two new individuals from the genomes of the original pair. Individuals are chosen randomly in proportion to their fitness (in *selectparents*) from the last complete generation to take part in crossover operations. A common procedure for selection is the roulette-wheel approach: a uniform random variate on {0...1} is generated. If the value is less than the (suitably normalized) fitness value of the individual being considered, it is selected for reproduction in a crossover operation. Crossover operates by cutting two strings in half at the same location and connecting the first half of the first cut string with the second half of the second cut string and vice versa to generate two new strings from the original two. [Figure 4-37](#) is an example. The two strings on the left are mixed using a crossover point between the fifth and sixth string location to produce two new strings shown on the right. The example is of a one-point crossover operator. It is possible to have two or more crossover points. The location of the crossover point is, generally, chosen at random.

Figure 4-37. Example of the crossover operation in a genetic algorithm.





The central idea of GA is that when the crossover operator acts on individuals with beneficial schemata (particular gene sequences) it can produce individuals with a higher concentration of good schemata. But crossover can also disrupt a good schemata if the crossover point occurs within its defining symbol positions. Importantly, crossover can also create or propose new schemata, which are not present at all in either of the original individuals. Most new schemata may be fatally destructive to the resulting individuals fitness, but when a beneficial new schemata is chanced upon, the further actions of fitness evaluation and selection for reproduction in the next generation mean that the new "idea" tends to be retained and multiplied as a characteristic in subsequent populations. Moreover, under GA action this new characteristic will be tried out in combination with other design ideas present in the genomes of surviving individuals.

A theoretical basis for GA effectiveness is given by schema theory. A *schemata* is a particular pattern of genes which can appear in an individual's defining genome. For example, the bit sequences 100100 and 101101 both contain the schemata defined by 10x10x, where x is a wild card bit location. Each schemata represents a hyperplane of the solution search space. A schemata can be beneficial or harmful to a pattern's overall fitness depending on what its effect is on the solution. The fitness of an individual is determined by the interaction of all of the schema present in its genome. If an individual contains a beneficial schemata, it will, on average, have a relatively high contribution to the formation of the next generation, and more copies of itself will result. On the other hand, if an individual contains a harmful schemata, it will, on average, have a relatively low contribution to the next generation. The net effect is that the proportion of the population that accumulates beneficial schema will increase with the number of generations, while the representation of the harmful or ineffective schema decreases.

A *mutation* operator is used in addition to crossover to continually enrich and diversify the population, making sure the population does not fall into using only a limited vocabulary of schemata. Mutation effectively says now and then, "Let's throw *this* idea into the mix," thus diversifying the search into parts of the solution space which may not be represented by the individuals in the population. A typical mutation operator works by flipping a gene in an individual's defining genome if it is binary or otherwise modifying the value if integer. The probability that a mutation will occur at a particular gene is set to some small value (typically 0.1% or less, per iteration or generation). An individual modified by a mutation operation would thrive if it contained a new schema which provided it with benefit, otherwise, the selection process will cull it out. After mutation in the generic procedure above, all new individuals have their fitness assessed (*evaluate* $P'(t)$) then the next generation (of a constant population size) is formed by keeping only the V most fit individuals from the union of the last generation and the current batch of offspring ($P = \text{survive}(P, P'(t))$).

In practice one of the key factors in deciding whether or not to try a GA approach on a given problem is the availability of an effective genome encoding scheme and a readily evaluated fitness function. The genome must contain all the information needed to fully specify a given design in a reasonably compact form. It is also essential that the problem and encoding scheme exhibit the property that the vast majority of crossover operations result in new individuals that remain *feasible* solutions. This is where use of GA on some problems can be awkward if an encoding scheme and crossover operator are not known that together tend to produce feasible resulting combinations. For instance, one might devise a genome to encode ring structures on a graph by listing the edges that form a ring as its genome. In this case, however, few crossover operations will result in a new specification of rings that is even functionally meaningful or qualified as a solution. Most crossovers in this scheme result in loss of ring structure in the combined schema. (i.e., a ring is crossed with a ring, but the result is an arbitrary open fragment or unconnected subgraph of edges). The general issue is that if one is trying to breed good widgets, say, one needs a schemata and combining operation that can produce either good or bad widgets, but does not often (or at all) produce outcomes that are not even widgets at all. A change in the encoding scheme can sometimes work. In the ring example for instance an alternate approach is to code each possible ring directly as a symbol. Combinations of schemata then always result in specifying a set of rings for further consideration, but at least does not produce meaningless specifications. This is the most important general issue in approaching problems with GA methods. Repair operators can be defined to work on a few cases of feasibility loss following a crossover but repair is not good to rely on for the bulk of most crossovers. Another way to think of it is that the solution domain must be mathematically group-like under the crossover operation: that is every operation on two members of the group of feasible solutions must be another member of the same group.

4.18.3 Tabu Search

Tabu search^[7] (TS) is not based on an appealing analogy such as in SA or GA. Instead, TS developed from an initially simple idea by Glover to improve the performance of a conventional descent (or hill-climbing) search by at least temporarily prohibiting the search from revisiting certain regions of the solution space (making them *tabu*), once they have already been visited. Such prohibitions force search diversification and lead to the breaking out of local minima or limit-cycles near a local minimum. The field has since developed greatly with a range of detailed tactics all under the broad method of Tabu Search. The book [GILa97] offers comprehensive coverage.

^[7] Tabu, spelled as such (i.e., not taboo), is the name given to this technique by F. Glover [GILa97].

Like SA a tabu search employs some problem-specific mechanism for generating "moves" that change the configuration of the current best solution. The generation of moves is the basic driver of the search. Qualification of allowed moves by tabu considerations is what improves the search effectiveness. A move will be adopted if it provides an improvement *and* it is not currently in a list of tabu moves. Tabu moves are based on short- and long-term memory of the prior sequence of moves taken or exact solution states visited. The simplest strategy would be to make the reverse of the last move taken tabu for the current move and/or to not let the last move taken be repeated for some minimum number of upcoming steps. These basic ideas have evolved into strategies involving recency and frequency memories and even consideration of the statistical distributions of the benefits associated with past instances of the various types of moves. A key concept is that from every current configuration s , a set of k possible moves that may be applied define a *neighborhood* of s , $N(s)$. $N(s)$ is the set of (feasible) configurations or design states resulting from application of each move (individually) on s . For instance if the current solution, s , is specified as a vector of span spare capacities, then two basic move operators could be to increase or decrease spare capacity on one span at a time by a unit amount.

The neighborhood is the set of all new solution states reachable by application of the move operators. Thus in this example, each solution state would have $k=2|S|$ neighbors to be evaluated. Not all of the k moves from s may result in feasible solutions and some of the moves may be tabu. Assuming a minimization problem, Tabu Search can be summarized as follows:

[\[View full width\]](#)

Generic Tabu Search Algorithm {

$i := 0$

generate an initial solution, s_0 and its obj. value $f(s_0)$.

repeat {

 generate neighbor set $N(s_i)$

 generate obj. values for each neighbor, $f(s_j) \forall j \in \{1 \dots |N(s_i)|\}$

 identify any neighbor which are tabu: tabu set $T(s_i) \subseteq N(s_i)$

 identify any aspirant subset of tabu neighbors, $A(s_i) \subseteq T(s_i)$

 choose $s_{i+1} \in (N(s_i) - T(s_i)) \cap A(s_i)$ for which $f(s_{i+1})$ is minimal

$i := i + 1$

update search memories()

 if termination criteria met (time, obj. value, etc.) {

 done := true}

 } until done}

procedure update search memories() {

 Update any recency or frequency memories and aspiration criteria according to the

 ➡ move taken and the problem-specific tabu strategy being used. }

Note that at the step where we choose s_{i+1} , the best qualified neighbor may have *higher* objective value than the present solution, forcing a step away from a local minimum. The concept of *aspiration* is simply that an exception to a tabu rule can be made if doing so would satisfy certain separate criteria. Such a neighbor is called an *aspirant*. Typically, the intent of the aspiration criteria are to admit the move if it would lead to a significant one-step jump in solution quality, or lead to the best objective function value yet seen, or, if without loss of $f(s)$ value it would provide some other desirable side objective that the planner may aspire to have in the result. In the general procedure this is why all neighbors of each state are evaluated before disqualifying tabu neighbors. Even though we might know the corresponding move is tabu at this stage (and hence tempted not to evaluate it) the move could still qualify under aspiration rules.

The criteria for determining what neighbors (or equivalently what moves) are tabu at any step is typically based on *recency* and *frequency* histories or simple absolute rules. A particular move may be tabu if, for instance, it leads to a solution which has already been considered in the last Q iterations, or has too high a number of building blocks in common with recently seen solutions. A move may similarly be tabu if it leads to a solution that has been revisited often before, but without a resulting solution enhancement following the move. Thus, success with the generic TS engine above, depends considerably on the problem-specific strategies developed for recency (short term) and/or frequency (long term) memories about past solutions and in the interaction of these tabu criteria with the aspiration criteria that can override them. This is an aspect of TS that makes it perhaps the most flexible and easily evolved in that elaborate intuitions or problem-specific insights can be captured in the tabu rules. A simple tabu rule might be: *if spare capacity was added to span i less than 10 moves ago, don't try doing so again just yet*. But at the other extreme, a highly evolved rule to guide the search might be developed such as: *if, over the last X times move Y was employed there was resulting increase (decrease) of $K\%$ over the following N iterations, then reduce (increase) the amount of time move Y is tabu following its subsequent uses*. This illustrates just how adaptive and sophisticated the rules based on recency and frequency histories can become with TS. For some problems of practical importance and heavy production use in industry, highly evolved TS strategies have been developed and perform extremely well.

Tabu search also involves strategies for explicit search *diversification* and *intensification*. If a number of iterations have passed without significant net progress, a diversification criterion may be triggered which typically acts like a random restart, applying some problem-specific method to explicitly jump into another whole region of the solution space and continue from there, possibly also resetting the recency and frequency memories.

4.18.4 Comparison of Optimization Techniques

To conclude this section on optimization methods, [Table 4-2](#) offers a comparative summary of strengths and weaknesses for the four main methods discussed: ILP, SA, GA, and TS. Most items in the table are self explanatory but a few comments are warranted. The comparison is meant only as a qualitative guide based on experience by the author and reports from others. By "GAP information" we mean the ability of the method to provide at least some indication of solution quality. Even when an ILP is not solved to optimality, it still provides GAP information indicating a bound on how far from optimal the result is. The "code development" column refers to the extensiveness of the coding effort to get going with some new problem using the methods given. For ILP it can be zero if using arc-flow models. For arc-path models code is usually required to enumerate eligible route sets of various types. SA can require little code development requirements. If a standard SA engine is adopted for the outer loop of the program then the problem-specific evaluation of the objective function is almost all that is needed. GA and TS both require much more coding in general to represent a problem to where a first run can even be attempted.

GA is so conceptually appealing that many of us are drawn, at least once or twice, to try to use its framework for that reason alone. But success with GA is highly variable and problem-specific in practice. Sometimes much effort can be expended and ultimately abandoned because the GA just cannot be made to work right for a particular problem. For the coding effort involved, a more certain bet on average is to develop a TS model. Far less framework is asserted onto the problem under TS compared to GA and once a TS model is running, it is amenable to considerable ongoing refinement which can be well worth the effort on an important production problem. By a production problem we mean one that is run repeatedly and at large scale for real operational uses, such as in airline operations or future transport network operations. The recent O.R. literature carries many reports of successful TS development efforts and even TS/ILP combined strategies wherein an LP/ILP subproblem is evaluated under a TS search framework.

Table 4-2. Comparative Features of Combinatorial Network Optimization Approaches

Method	Strengths	Limitations	Keys to effectiveness	Code dev.	Typ. Best Uses
ILP and ILP heuristics	Formal model, optimal solutions, fast studies, AMPL/CPLEX tools. GAP indicates solution quality.	Problem size limitations, linear constraints and objective relationships.	Suitable relaxations, variable priorities, problem decompositions, ILP-based heuristics.	Low	Research, comparative studies, benchmarking for heuristics.
SA	Simplicity and generality. Fewer "tunable parameters" than other meta-heuristics.	Can be slow. Can require much experimentation. No GAP info.	Cooling schedule and number of iterations at each $T(t)$.	Low	Fairly specific problems where SA works best.
GA	Faster convergence, highly detailed problem models, arbitrary functions and relationships.	Hard to fit certain problems, can require much experimentation and tuning. No GAP info.	Finding a suitable genetic schema and crossover operations. Behaves best when fitness is continuous not discrete.	Higher	Nonlinear or highly detailed problem frameworks.
TS	Universally applicable framework. High performance often documented, framework accommodates continual enhancements to TS strategy.	Coding intensive approach.	Move definitions, tabu strategy, use of recency and frequency data.	Highest	Heavy production use problems.

Further Resources

Above, we have tended to give the primary references for attribution of each method (SA, GA, TS) to its historical originators. While Glover's book [\[GILa97\]](#) still serves well for TS (also [\[GILa99\]](#)), the primary references for SA and GA may not be as accessible. More recent books are [\[RiHa01\]](#)[\[Rawl91\]](#). Numerous web-based guides [\[BeBu93\]](#) and sources of executable code such as [\[Heit93\]](#) or [\[Ingb03\]](#) are also available for SA and GA.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

Part 2: Studies

[Chapter 5. Span-Restorable and Span-Protected Mesh Networks](#)

[Chapter 6. Path Restoration and Shared Backup Path Protection](#)

[Chapter 7. Oversubscription-Based Design of Shared Backup Path Protection for MPLS or ATM](#)

[Chapter 8. Dual Failures, Nodal Bypass and Common Duct Effects on Design and Availability](#)

[Chapter 9. Mesh Network Topology Design and Evolution](#)

[Chapter 10. \$p\$ -Cycles](#)

[Chapter 11. Ring-Mesh Hybrids and Ring-to-Mesh Evolution](#)

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

Chapter 5. Span-Restorable and Span-Protected Mesh Networks

A span-restorable network protects services by protecting the working capacity on which they are borne. This is different from the path-oriented schemes to follow, which protect services with end-to-end replacement of the service paths themselves. Span restoration (SR) acts locally in the vicinity of a failure to deploy a set of detouring path segments between the end nodes of a failed span. This has certain advantages in terms of simplicity and speed because it is only the status of each working channel on a failed span that needs to be known, not the overall destination or routing of the affected service paths. In addition, with *distributed preplanning* (DPP), restoration paths can be continually discovered in advance of a failure, establishing a preplanned shared-capacity protection scheme where each replacement path is known ahead of a failure. Compared to path-oriented schemes path characteristics such as length, optical loss or noise can be more easily controlled under span restoration to ensure optical path quality. On the other hand, because it acts locally, a span-restorable network generally requires more spare capacity than a path-oriented approach. Span restoration also assumes a means of fault detection at the OXC nodes adjacent to a failed span so that the failure is isolated and the two local end-nodes are suitably notified to activate the protection path-set. This chapter is concerned with the logical capacity and routing design of this class of network.

Span-restorable networking can be implemented at the OC-n, wavelength, waveband, or at the whole fiber cross-connection level. When considered at the fiber-level it offers a networking option that is not applicable with path-oriented schemes. An OXC for whole-fiber switching is relatively simple and compact and a simple pilot tone can be monitored to detect fiber integrity at each node, even if the lightpaths are transparent through the node. In addition, all fibers physically arriving at a site would normally be terminated at such a fiber OXC, whereas at the wavelength level, much bypass may be present, complicating restoration. At the whole-fiber level for protection, churn and dynamic traffic effects are also not as prevalent as at the wavelength layer. Protection preplans can be continually regenerated to have the needed cross-connect switching arrangements in place in advance of a failure. And when whole fibers are switched there is no need to plan wavelength-specific protection paths or to require wavelength converters for access to protection paths. In addition, by using fiber-level cross-connects to terminate all fiber transiting each node, the logical dual-failure scenarios which complicate restoration at higher layers are contained to being single-failure events. The protection preplanning process can also be invisible to the service provisioning process, and not dependent on exchange and synchronization of network-wide topology and capacity databases. Thus, although it is an older and simpler approach than path-oriented schemes, there is still considerable motivation to study the design and properties of span-restorable mesh networks.

5.1 Updating the View of Span Restoration

Distributed automatic ("Self-Healing") span restoration was first proposed in 1987 [\[Gro87\]](#) [\[Gro89\]](#). Study of the corresponding spare capacity design problems began about 1990 [\[SaNi90\]](#) [\[GrBi91\]](#). As such, most prior work on span-restorable networks was conducted in the context of SONET technology. But it has direct logical applicability to DWDM "optical networking." Differences in terminology are superficial to what is a basic concept, applicable to any technology era. Some important paradigms have changed, notably the view of static forecasts versus dynamic demand, the adoption of IP-centric control protocols, and the greater transparency of lightpaths relative to SONET SPEs. Wavelength conversion is so-far also more technically difficult and costly than time slot interchange. But span-restorable networking itself is a generic logical approach to the restoration problem and is not tied to any one technology. The concept is just as relevant to DWDM-based networks as it was to SONET.

One of our aims in this chapter will thus be to update the thinking about span-restoration, lifting it from its apparent SONET-era stigma and showing in many ways how remarkably well suited and advantageous it can be for dynamic optical networks as well. This entire approach and each of the following concepts stands in sharp contrast to the much more software and database dependent IETF-driven view of end-to-end disjoint path protection, which has been almost the sole approach considered for optical networking. Thus, in the spirit of at least providing some fresh alternatives for the industry to consider, we will show in this chapter (and in part of [Chapter 8](#)):

1. How span restoration can be almost ideally suited to dynamic demand environments with the Protected Working Capacity Envelope concept.
2. How all of the channel-associated signaling needed for statelet-driven self-organized distributed restoration algorithms (DRAs), previously based on SONET overheads, are now also available in optical networking with advances such as Digital Wrapper or Optical Service Channels and the Link Management Protocol.
3. How the Distributed Preplanning (DPP) concept updates span restoration methods to provide preplanned protection, not just on-demand restoration, options.
4. How the *first-failure protection*, *second-failure restoration* concept makes possible priority services classes having "better than 1+1 availability" through dual-failure restorability using a DRA both for preplanning and adaptively on demand for dual failures.
5. How gracefully span-restoration lends itself to an operating of multiple Quality of Protection service classes.

5.2 Operational Concepts for Span Restoration

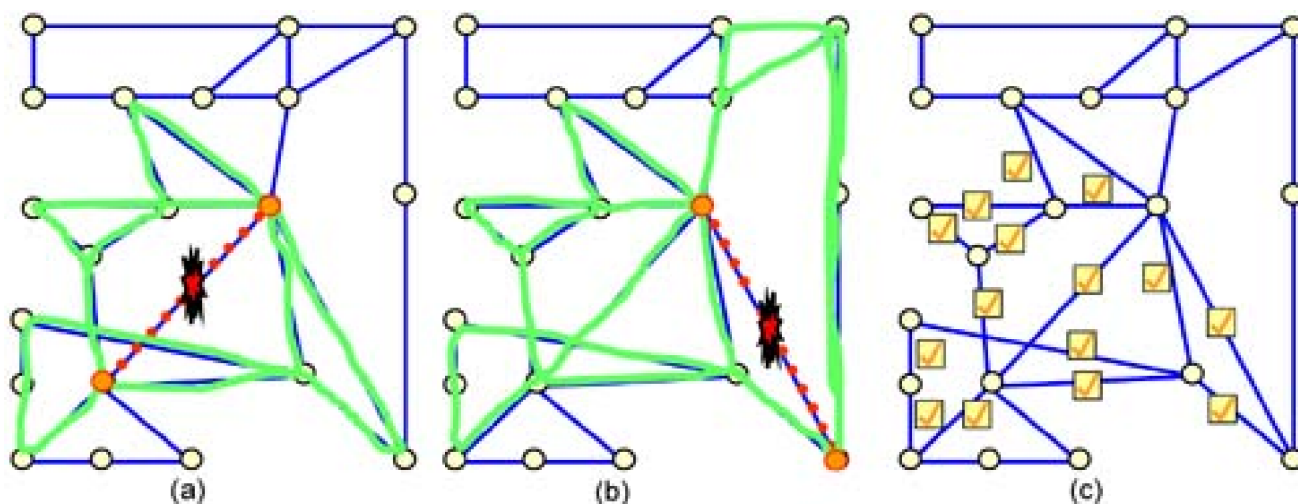
Before proceeding to the capacity design aspects of span-restorable networks, this section covers several of the important operational concepts surrounding span-restoration. This is required to ensure a correct basic understanding of the functional model that underlies the later capacity design methods. It also helps motivate why we should consider span restoration as an option, through the "updating" of relevance to optical networks. These preliminaries will show that span restoration offers some rather overlooked but advantageous properties and options for a network operator.

5.2.1 Details of the Span Restoration (SR) Concept

Let us start by simply looking at an example of span restoration as shown in [Figure 5-1](#). In (a) and (b) the figure portrays two different span failure scenarios and shows examples of the restoration routesets that might be deployed to recover from these failures. In [Figure 5-1\(c\)](#) a check mark identifies each span on which *spare capacity sharing* occurred over the two failures in the example. With reference to [Figure 5-1](#), the following points help further convey the key ideas of span restoration:

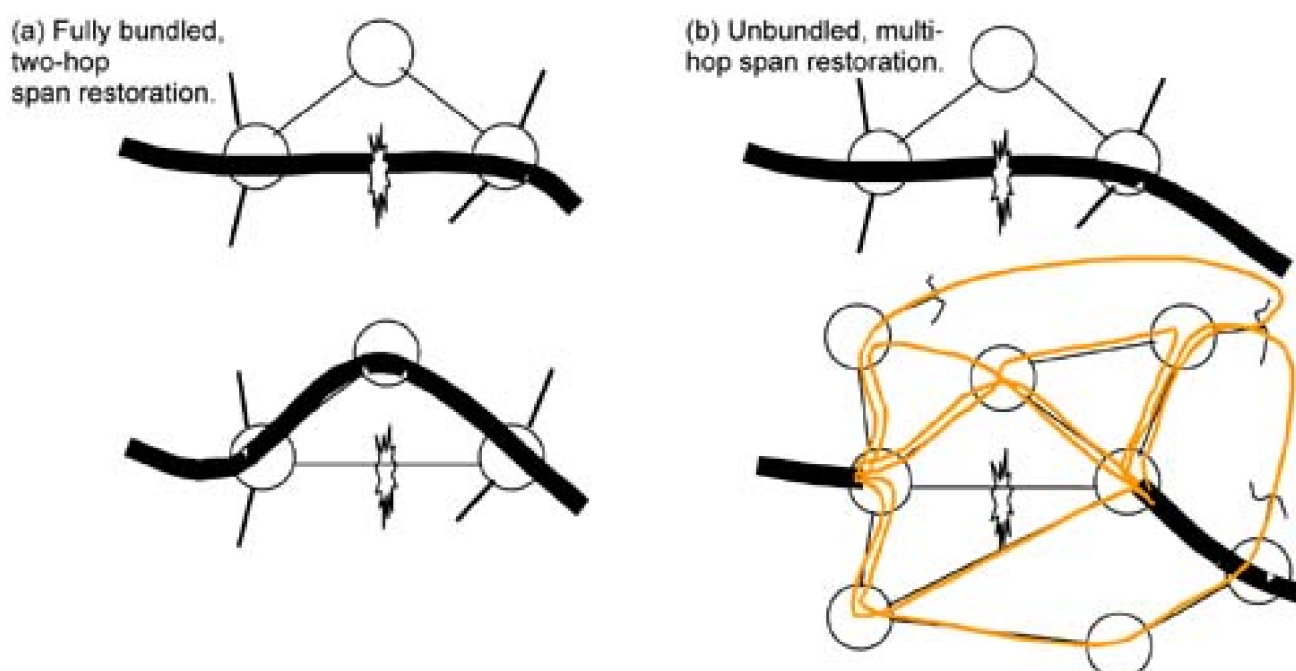
1. For capacity-planning purposes the span cut is assumed to completely sever all the working and spare capacity on the failure span. Operationally, however, if not all working channels fail, only the failed channels will be rerouted.
2. In the special case of a single channel failure, the restoration process will act like a 1:1 APS, finding and immediately using a surviving spare channel on the same span as the failed channel. The same applies for multiple independent channel failures until there are no remaining spares on the direct span. Any further failed channels are rerouted through the network just as they would be under restoration against an entire cable cut.
3. The set of individual restoration paths shown in [Figure 5-1\(a\), \(b\)](#) are each collectively referred to as the restoration "path-set" for the respective failure. The set of routes over which the path-set is implemented is called the restoration "route-set."
4. The restoration path-set is formed entirely within the multigraph whose links represent the spare channels on surviving network spans. The working capacity layer determines the number of paths needed for restoration (i.e., w_i if span i fails) but the restoration path-set is formed entirely within the spare capacity multigraph only. w_i denotes the number of working channels to be protected on span i of the network.
5. The multigraph representing the number and topology of all spare channels in the networks is sometimes also called the "reserve network."
6. The two nodes adjacent to the failed span are together referred to as the "custodial nodes" with respect to the particular span failure.
7. Restoration paths are formed from individual cross-connections between discrete spare channels on each span.
8. The restoration path-set is not limited to employing a single replacement route for all failed capacity. It may, as needed, comprise paths on all distinct routes (up to a prescribed hop or distance limit).
9. The theoretically ideal performance model for a span restoration mechanism is that (in the absence of a limit on the w_i requirement) it should be capable of deploying a path-set that is equivalent in total flow to the max-flow capacity of the reserve network between the custodial nodes without using any capacity on the direct span. i.e., $restoration\ flow(span\ x-y) = \min(w_{x-y}, max-flow_{x-y}(G))$ where x,y are the custodial nodes in reserve network G .

Figure 5-1. A visual appreciation of span restoration, and how spare capacity is shared.



Point 8 touches on an apparently common misunderstanding about span restoration—that restoration is via a single simple deflection of the total flow from the failed span. Diagrams similar to [Figure 5-2\(a\)](#) all too often seem to portray span restoration this way, but are fundamentally misleading as they fail to convey the far greater generality of rerouting that is part of the concept. As [Figure 5-1](#) or [Figure 5-2\(b\)](#) illustrates, the general concept is to allow the most diverse path-set formation that may be needed to efficiently use the spare capacities available on other spans. This means rerouting at the granularity level of capacity management at the associated cross-connect nodes. The failure may sever multiple fibers, but the restoration path-set can be constructed at the fiber, waveband, wavelength channel, or even OC-n level, depending on implementation. The general concept also allows that restoration paths may be of any length required up to some logical or physical length limit, H . The case shown in [Figure 5-2\(a\)](#) is in fact only the limiting case of $H=2$ with "full bundling." The $H=2$ fully bundled model is the least efficient for sharing of spare capacity over different failure scenarios—often only two or three span failures will be able to share the spare capacity on a span. (For example, reconsider the failures of [Figure 5-1\(a\)](#) and [\(b\)](#) if only a single two-hop reroute is allowed). Spare capacity levels are also higher with full bundling because a single restoration route must have all of the spare capacity to handle the failure rather than allowing the spare capacity of many routes to be employed. Having stressed these basic points, practical implementations can involve limitations on hop-limits so that realistic restoration path-sets would typically not be as complex as the "concept" example in [Figure 5-2\(b\)](#). Similarly, partial bundling can enhance speed by reducing signaling volumes with little or no loss in overall capacity efficiency. The main point is, however, is to completely dispel the commonly encountered view of span restoration implying what [Figure 5-2\(a\)](#) portrays as the restoration principle.

Figure 5-2. A common but misleading portrayal of the span restoration concept is the special case of $H=2$ with full bundling in (a). Unbundled multi-hop restoration in (b) is more efficient.



5.2.2 Concept of Distributed Self-Healing Mesh Restoration and DRAs

Span-restorable networks lend themselves well to the application of a distributed restoration algorithm (DRA) that operates with state-driven signaling interaction on the overhead bytes of each lightwave channel, or through signaling on the OSC for all channels in the associated fiber. DRAs differ remarkably in implementation and performance from explicit message-passing distributed protocols in the IP control plane, such as OSPF-TE or CR-LDP. None of the current IP-centric control applications are really "distributed" protocols because they rely on having a current and complete global database view of the network state in every node. Each node is really solving an instance of the centralized computational version of the respective routing problem at hand. In contrast DRAs have no database and form restoration path-sets through a truly distributed self-organizing event-driven interaction of state-based signaling bytes on spare channels. The physical state of the network is the database within which they execute.

The differences are essentially the same as comparing OSPF-TE/CR-LDP which is a software and database intensive high-level protocol suite to the SONET K1-K2 state-driven byte-signaling protocol for automatic protection switching ([Section 3.4.2](#)). It may be difficult to think of these as being comparable but they are in the sense that they both solve a restoration rerouting problem. OSPF-TE coupled with CR-LDP is representative of IP-centric control protocols which have tens of thousands of lines of source code and rely on accurate global state databases. In contrast the K1-K2 byte protocol can be implemented in a few hundred logic gates, including the trivial finite-state machine (FSM) "program" that controls APS switching. How does this relate to DRAs? DRAs are the mesh rerouting equivalent of the simple K1-K2 byte-driven APS protocol. They effect span-restorable path-set formation with FSM-driven node protocols that can be implemented in firmware, not greatly more complex than the K1-K2 state-driven APS protocol. First developed in the 1990s for SONET, this approach has been largely overlooked for optical networking applications, which have been rather captive to the software and database intensive IP-centric approach.

Distributed restoration effects a much more primitive, lower-level, autonomous and collective response to a failure. This gives it speed and database independence well beyond that achievable through high-level packet messaging protocols. The network state as it exists at the moment of failure *is* the database, not a centralized software abstraction of the global network. This lets us realize the rerouting computation for restoration without any database requirements. Every node will perform in an apparently isolated manner, but the independently deduced cross-connection decisions of each node collectively constitute efficient multipath rerouting plans.

A DRA does not replace centralized supervisory control over the network. Rather, the DRA is an embedded real-time assistant to centralized administrative systems. Network elements will react autonomously to the emergency, coordinating themselves under the DRA, but they then use normal IP messaging to a network operation center (NOC) for ratification of restoration actions, and for all other normal provisioning and maintenance purposes. All DRAs use some version of forward flooding (or selective forward flooding) and reverse linking as introduced in [\[Gro87\]](#) [\[Gro89\]](#) (for span restoration DRAs), and employed again for a path-restoration DRA in [\[Gr00\]](#). The flooding is not, however, a simple repeat broadcast as in LSA dissemination. It is also state-based not message-based. The flooding in a DRA which obtains all paths of the restoration path-set in parallel, such as the Self-Healing Network (SHN) protocol, is controlled by a compact set of nodal rules which determine which incoming restoration state bytes (or "statelets") are relayed.^[1]

^[1] The SHN protocol was first proposed in [Gro87](#) but an improved description, in [Gro97](#), is recommended for readers interested in seeing how it works. Many still tend to think that what the SHN achieves is impossible without preplanned databases or global topology-discovery techniques. The SHN spontaneously assembles restoration path-sets as a low-level self-organizing interaction amongst nodes with no global database or topology knowledge. [Gro97](#) gives a better explanation of how this works than the earlier [Gro87](#).

5.2.3 Self-Organizing Nodal Interaction via Statelets

In the SHN Protocol [\[Gro89\]](#) and related protocols employing this form of simple state-driven signaling [\[Gr00\]](#), [\[StGr98\]](#), [\[GrSt98\]](#), nodes interact solely through *statelets* on the channels (or channel bundles) between them. A statelet is not an inter-processor data message in the usual sense—rather it is a channel-associated overhead state byte much more like the K1-K2 bytes in SONET than, say, like an OSPF data packet. In fact, one can say that the K1-K2 bytes in SONET *are* the statelets for the SONET APS protocol. One study by Bellcore [\[Bell93\]](#) tested a variety of DRA proposals for SONET and concluded that signaling of this type, rather than DCC-channel based messaging, was probably essential to meet the 2-second speed of restoration target at the time. Related work and conclusions are also reported in [\[WuKo93\]](#). DRAs based on statelet-type signaling are just extensions of the SONET APS protocol to distributed mesh restoration. Statelets are invisible to traffic on the link but are physically inseparable from the transmission signal itself. Statelets are

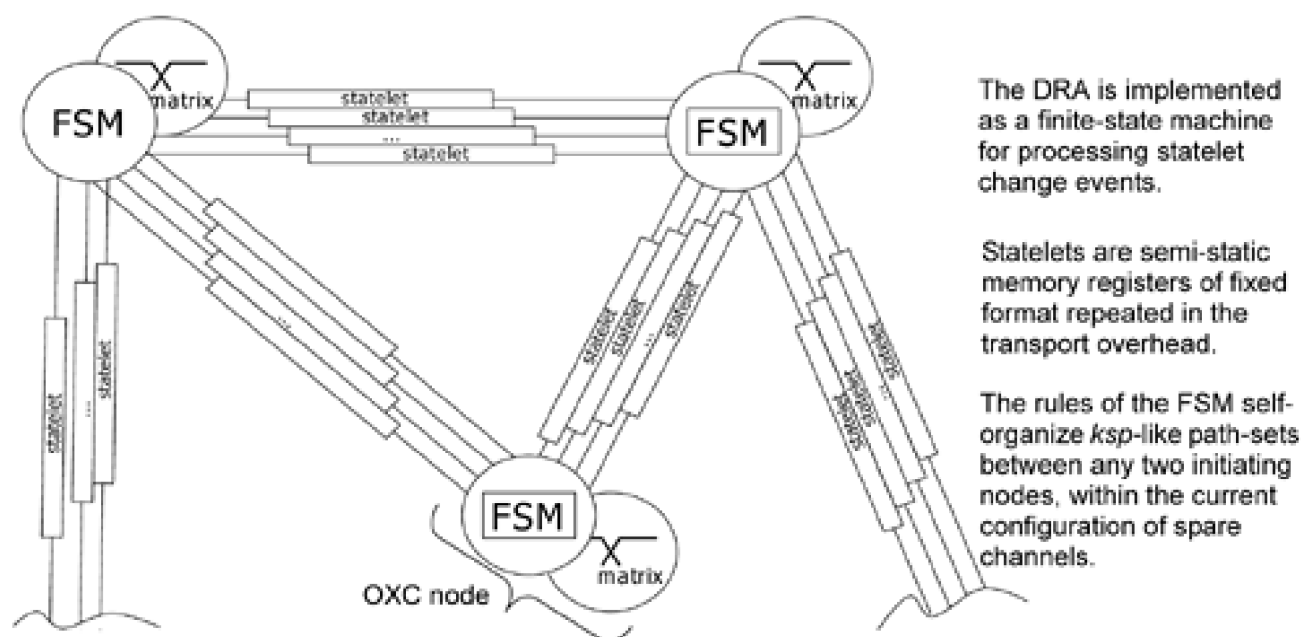
comprised of fixed information fields, some of which are specific to the application that the basic protocol is used for. A statelet is not addressed as a message is; it arrives at whatever node and in whichever port the transport signal is connected. Similarly a node originating a statelet on a link does not explicitly know who will be influenced by its direct or indirect effects.

SHN nodes do not rely on a data communication link between node processors to coordinate their restoration efforts. Rather, the SHN task sees the outside world only through the statelets appearing locally on the links incident to it. Nodes are effectively isolated and act solely as a specialized processor of statelets. The self-healing effect occurs as the network-level side effect of the apparently isolated actions of each node following a set of rules for reaction to and creation of statelets. The action of nodes, particularly of the nodes not directly adjacent to the failure but involved with cross-connecting spare channels to form restoration paths (called Tandem nodes), appear more like the conduct of a card game than any explicit algorithmic attack on the rerouting problem at hand. This game is defined by rules for operation on statelets and by a goal of seeking certain combinations of statelets among several families of statelets that will be present. A condition of matched statelets authorizes a node (a player)—in total unawareness of the actions at any other node—to make a cross-connection. The independently derived cross-connection decisions at each node will collectively form coherent restoration path-sets.

This clearly different approach to restoration addresses the demanding requirements of restoration in terms of speed and accuracy (in the sense of never acting with out-of-date information on network configuration). Speed arises from the properties of event-driven processing of statelets rather than software handling of queued messages. Statelets are semi-static information fields repeatedly impressed on the bearing signal in hardware. As a result the entire network state information needed for restoration is stored on the links of the network itself. Nodes sit within this database viewing only the part of it that is adjacent to them. Changes in statelets are detected in hardware in the port cards of the OXC nodes, allowing parallel evolution of the collective state of all network links to proceed with little more than the statelet repetition and propagation delays. No inter-processor communication stacks are involved. A change in a statelet is trapped in hardware and raises a flag for service. To the Operating System (OS) of host OXC the SHN task looks simply like a specialized interrupt handler for handling statelet change events.

The SHN protocol thus executes within a memory space which is comprised of the array of incoming and outgoing port card statelet registers and neighbor nodes are directly coupled in a shared-memory sense. When a node creates or changes a statelet, the memory state within the adjacent node is directly updated. The effects of protocol execution at one node are automatically reflected in the memory state within which adjacent nodes are concurrently executing. This couples all nodes as a system but eliminates the real-time burden and delays of serial inter-processor messaging and protocol stacks with error recovery, message ordering, and queuing if conventional packet messaging was used between nodes. [Figure 5-3](#) illustrates this logical model of nodes coupled through the shared memory which are the links of the network itself with state information in their statelets. The spatial position of links on which various statelets exist also encodes information relevant to the process.

Figure 5-3. Shared memory execution paradigm of the SHN protocol for adaptive real-time restoration or prefailure distributed preplanning.



In addition, the protocol action is always defined by the current state of links surrounding the node, not by previous messaging histories, so statelet change events in a port need not be queued. It is only the current state of the link when the SHN task considers it that matters to the network wide evolution of the result. There is no receive message buffering: a new statelet overwrites, rather than follows, any

previous statelet. It is important to see that each change in a statelet is not a packet data message to be processed in the usual sense. Statelet-driven interaction can also be made almost arbitrarily robust to errors without retransmission needed in ordinary packet messaging protocols. Simple triplication testing is a highly effective form of error correction in an environment of fiber transmission where 10^{-6} BER is already considered an alarm condition. But a checksum can also be embedded in the statelet to absolutely validate it. A 64-bit statelet including a checksum can be subjected to three-times persistence validation in the receiving port to virtually eliminating a false statelet event and requiring only 3 ms to propagate an event over a single 64 kb/s statelet overhead channel.

Other DRAs and Common Misunderstandings

Local interaction via statelets leading to emergent parallel formation of complete and efficient sets of restoration paths is a unique feature of the SHN protocol above. Other DRAs have been developed and characterized that use more conventional inter-processor messaging and explicit route-seeking algorithmic approaches. Examples include FITNESS [YaHa88], NETRATS [SaNi90] and Kominé's DRA [KoCh90]. An in depth discussion of DRAs is found in [Gro94]. Studies reporting validation and performance assessment of DRAs include [SaAl98], [Gro89], [KoCh90], [ChKo91], [GrVe91], [IrGr00]. Much of the work on DRAs occurred during the SONET era and seems to have been overlooked for optical mesh networks, although it is logically just as applicable to DWDM-based optical networking— especially with "digital wrapper" or other forms of optical service channels to convey the physical layer channel-associated signaling that makes these schemes so fast compared to Internet-style messaging. An important point established by the literature just cited is that distributed restoration can be assured with 100% restorability of predefined levels of working capacity. This is relevant to the following discussion of the "working capacity envelope concept" for dynamic demand provisioning because a DRA can be the basis of an adaptive restoration response that, operating within a pre-designed environment of spare capacity, is *assured* to fully restore working capacity up to a known maximum level on each span. These maximum working levels associated with a given spare capacity distribution define an operational envelope for dynamic demand provisioning. But the adaptive nature of a DRA give extra advantages over protection schemes. These points need to be stressed in light of two misunderstandings sometimes expressed in papers on optical mesh networking. These are:

- **Misconception 1:** *that only protection schemes (not restoration) can give an advance assurance of 100% recovery.* While this is true of GMPLS auto reprovisioning (as we explain in [Chapters 3](#) and [6](#)), DRAs for span restoration (and one for path restoration [IrGr00]) *have been well validated* to reliably achieve 100% of the predefined working capacity restoration goals. Although dynamic and adaptive, these remain *assured*, not best-efforts, schemes when operating within well-know limits on working capacity.
- **Misconception 2:** *that distributed span restoration schemes don't support dynamic demand.* This is also not true, but presumably arises because many past studies employed a static demand forecast matrix for design studies. The logical extension to apply to dynamic demand environments is, however, direct: the static demand quantities need only be reinterpreted as traffic-based estimates of maximum requirements. The "static" capacity design then produces a pool of protected working channels on each span, available for dynamic demand provisioning. We develop this point further as the "working capacity envelope concept" below.

5.2.4 Distributed Protection Preplanning with a DRA

The concept of Distributed Preplanning with a DRA was mentioned in [Chapter 3](#) where we pointed out why it is too simplified to classify all mesh schemes as either protection or restoration. Let us now describe the concept of DPP more fully. DPP works as follows:

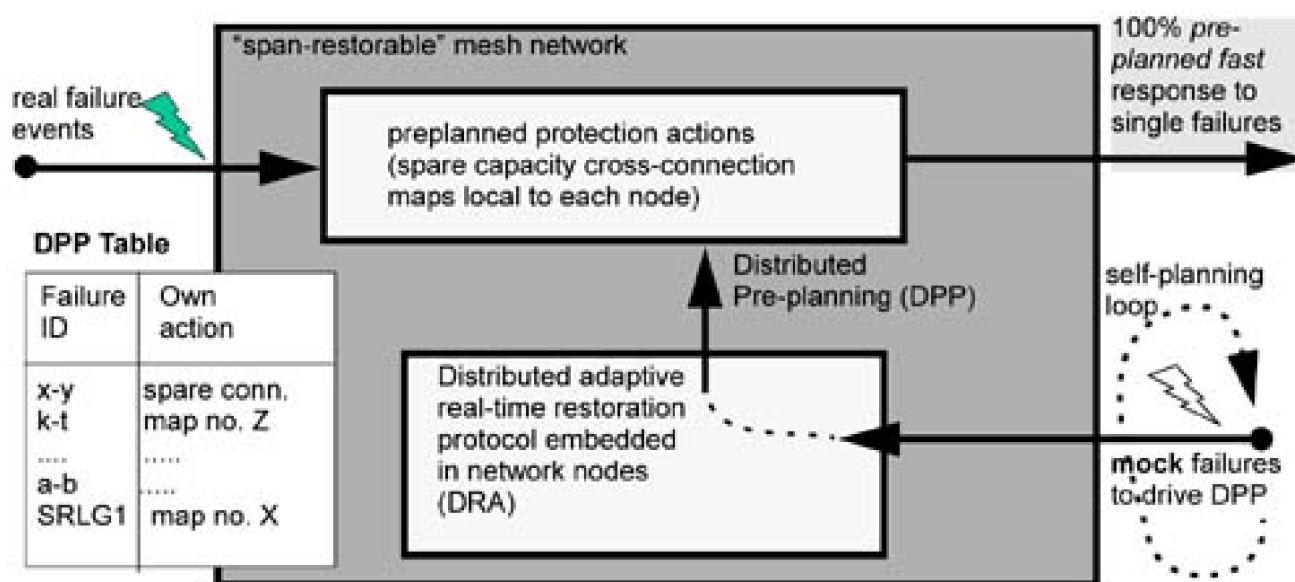
1. A DRA (such as the SHN protocol) is executed sequentially in a background mode for each possible span failure or SRLG in the network. This is a dress rehearsal in which no cross-connections are actually made. The failure scenario to simulate may be downloaded from central control or enumerated by the network nodes in round-robin by themselves.
2. Each node records the cross-connections *that it would have made* between spare channels for each trial run in which the DRA recruited it into making one or more restoration cross-connections. The node records the custodial node name pair that identifies the failure and the local spare capacity cross-connection map that corresponds. This results in a "DPP table" of instructions at each node for that node's portion only of the DRA response to each respective span cut of the network.
3. When a real span failure occurs, the custodial nodes observing the failure emit a single alerting statelet containing the span

identity and an indication equivalent to saying "*span (x,y) fail: this is not a drill!*" Alerting can be accomplished over an activation loop pre-established through all nodes or by disseminating a simple flooding alert which requires one statelet event in each node.

- Any alerted node that has non-null actions for a listed failure in its DPP table immediately makes the internal cross-connections between spare ports that are listed in its table, and changes the local status of the links used from spare to "in use." With MEMS or other optical array switch core technologies it may be possible for the whole spare capacity cross-connection map that corresponds to the given failure to be asserted in parallel.

Figure 5-4 illustrates this relationship between a span-restorable mesh network that contains an embedded DRA for real-time restoration and a corresponding preplanned span *protection* scheme supported in the same spare capacity environment. This works through distributed preplanning of protection reactions with the same DRA that would be called upon for real-time restoration if needed. We refer to this overall scheme, in which preplans are developed (and adapted) through a continual series of *mock restoration trials*, as *span-restorable distributed preplanning* or SR-DPP.

Figure 5-4. Concept of Distributed Preplanning: A DRA continually runs mock failure trials and nodes store only their actions for use as fast protection plans.



When a real failure occurs it is only necessary to disseminate the names of the end-nodes adjacent to the failure (which serves as a fault ID). All nodes then immediately deploy their corresponding preplanned cross-connection actions in parallel. Thus, in the same sense that an APS system is like a pre-stressed trap, ready to be sprung by a failure, the mesh network is also completely pre-wired with a collective restoration path-set for each failure that can be triggered into existence in parallel at all nodes just by dissemination of the fault notification. One possibility to very rapidly alert all nodes is with an activation loop, created by concatenation of some existing spare links in an arrangement that visits all nodes. A loop ensures that the span-cut cannot itself isolate any nodes from the alerting path. In the case of two cuts on the activation loop, or in any case where the pre-placed instructions do not yield 100% restoration, the DRA can then be triggered in real-time as a follow up. But simple flooding is also a fast form of activation. Even with simple flooding, instead of an activation loop, all nodes can be alerted into action to deploy the last stored self-planning result faster than they could have been by collectively executing the DRA itself.

DPP has considerable advantages over centralized preplanning since the database remains the network itself and there are no centralized computational or downloading of preplan files. The network operations center (NOC) has only to command (or pre-schedule) the cycle of background trial runs that continually update the distributed preplan. The NOC may alternately command updates selectively when network changes occur. If the DPP only protects the working paths as they are provisioned then there may be a finite "window of vulnerability" in the preplans between the time when provisioning occurs and when the relevant updates to the DPP are effected. If one assumes a rotation schedule which provides regular 10 second slots (which is generous) for each span to run a DRA dress rehearsal then a 100-span network would have its DPP regenerated entirely every 17 minutes. Any new service path would be automatically integrated into the fast protection preplans with at most this much delay.

In a more dynamic demand provisioning environment where full pre-provisioning of channel capacity is assumed (that is, all per-channel equipment is in place on all channels of the transmission system prior to use), new service paths can be established over an already protected "envelope" of working capacity on each span. There is literally no delay in establishing protection in this case, as the DPP

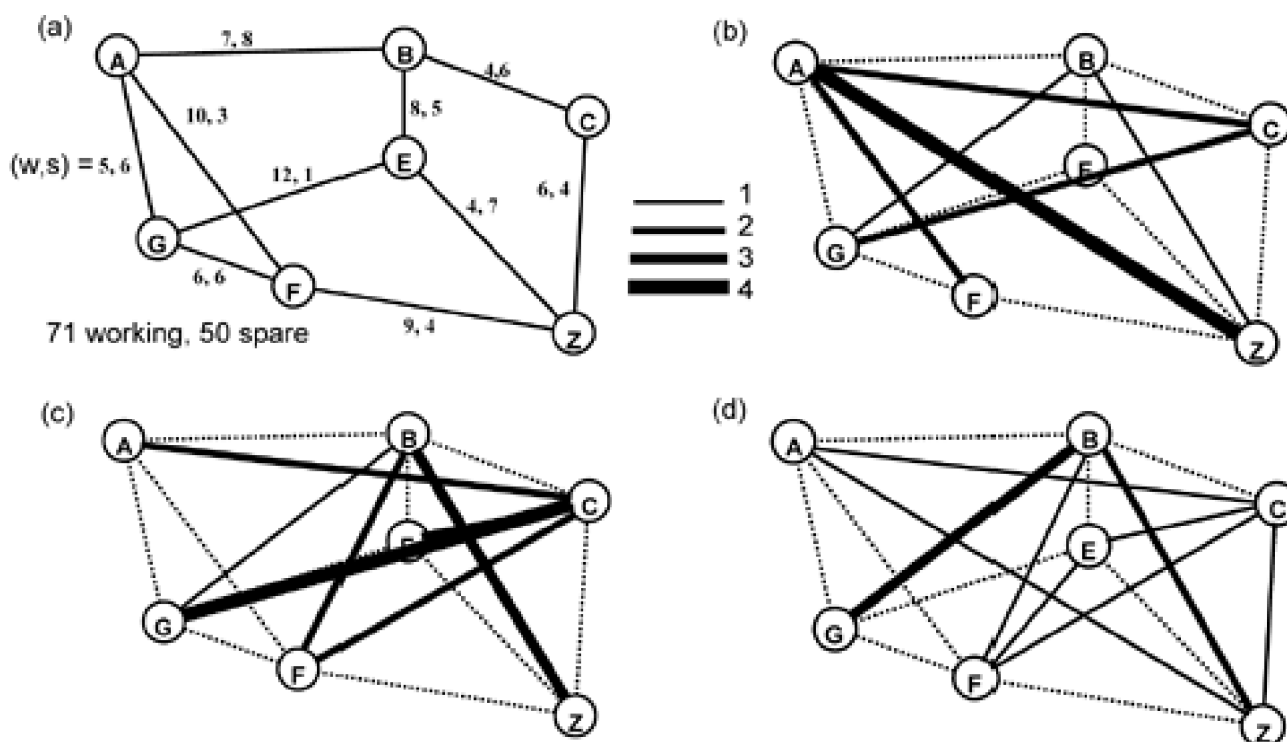
process is continually preplanning protection for the entire working envelope, not for individual working channels as they are provisioned.

5.2.5 The Working Capacity Envelope Concept for Dynamic Demands

Another attractive aspect of span-restorable networks is the operational simplicity of the *protected working capacity envelope* (WCE) paradigm to support dynamic service provisioning. In contrast to MPLS or optical layer shared backup path provisioning, where working and backup paths must both be determined in the provisioning process, and shared backup capacity must be coordinated with all other paths, span-restorable networks *protect the bearer capacity of the network directly*. This creates an inherently-protected amount of working capacity on every span that is a resource pool for provisioning. The set of all such protected capacities on network spans creates an "operating envelope" within which a vast number of simultaneously provisioned path states are feasible and inherently protected without any further consideration or even checking. Service provisioning is simplified to the equivalent of routing a new path in a previously non-protected network of point-to-point transmission systems. "Point and click" or fully automated dynamic service path provisioning are both greatly simplified because *provisioning need focus only on the working path*. As long as the path fits in the envelope, it is protected. And fitting in the envelope only requires that at least one unit of unused working capacity is available on each span of its route. Thus, with one envelope of working span capacities, a vast number of dynamically changing network connection states are feasible and inherently protected against span failures. The service layer merely routes the path over a shortest route on which working capacity is available. If multiple protection classes are desired (as follows in [Section 5.9](#)), a status indicator borne by the signal will indicate if it is in the protected envelope on each span or uses unprotected (or best-efforts) working capacity.

[Figure 5-5](#) illustrates the point. In (a) is a simple example of a mesh network design (obtained using the methods that follow) that is 100% restorable to any single-span failure at 70% redundancy. The working and spare capacities on spans are indicated as (w_i, s_i) values, respectively, in [Figure 5-5\(a\)](#). [Figure 5-5\(b\)](#), [\(c\)](#) and [\(d\)](#) just illustrate three of the vast number of different point-to-point demand combinations that are supported and inherently protected by routing within the protected working capacity envelope of [Figure 5-5\(a\)](#). Unlike path protection there is no determination of backup routes or coordination of spare capacity on backups required to provision each changing demand as it arrives. If it can be routed under the w_i of (a), it is protected, end-to-end, by virtue of embedded span replacement rerouting through the s_i quantities in (a), should a failure occur.

Figure 5-5. (a) A span-restorable mesh network and examples (b), (c), (d) of different (dynamic) demand patterns feasible under the protected working capacity quantities of (a).



In fact, the extent of the working capacity envelope is not fully portrayed by [Figure 5-5](#) because within the physical totals ($w_j + s_j$) of capacities present on the spans, there is yet further latitude if needed to revise the boundaries on spans between working and spare capacity to accommodate even more deviation in a dynamic demand pattern from the pattern for which the capacities were initially designed. Such "stretching of the envelope" is approached with the theory to follow.

In an optical network with OXC-based SR-DPP protection at the lightwave channel, waveband, or whole-fiber level, this means that rerouting for survivability is completely transparent to the payload types on each wavelength and there are no explicit provisioning operations required in the service layer to provide for survivability. The service layer merely routes over a shortest path through the available working capacity quantities, designates the protection priority or class, and receives a confirmation that the path is protected when routed within the current envelope of protected working span capacities. In contrast to SBPP, where explicit arrangements for protection are referred into the service provisioning problem for every path and a large amount of global network state is needed to coordinate sharing on backup paths, span restoration offers a simpler operational service provisioning paradigm: that of *provisioning over protected capacity versus explicitly provisioning for protection*.

Another important aspect of the working capacity envelope (WCE) concept is that it relates the mathematical models for capacity planning (i.e., most of what follows in this chapter) for "static demand" matrices to the emerging view of highly dynamic demand arrival and departure processes under fully pre-provisioned inventories of channel capacity. Under the WCE view of dynamic operations, the static "demand matrix" which we specify for a capacity design problem is simply reinterpreted as the generating exemplar that dimensions the working capacity envelope within which we want to serve dynamic demand. In this role a demand matrix no longer expresses the planner's forecast of exactly which future demands he/she expects to support, but rather his/her view of either:

1. The most demand expected to have to be supported on each O-D pair simultaneously, or,
2. In an analogy to traffic engineering, the number of lightpath "trunks" needed to serve the average instantaneous demand on each O-D pair at a target blocking level.

Regarding the second scenario, it is relevant to note that while blocking probability $P(B)$ is quite nonlinear as a function of offered traffic A for a fixed number of servers (N), it is easily shown that under Erlang B the number of "servers" required (here lightpath requirements) to retain a low constant blocking probability is essentially linear above a few Erlangs. For example: $N = 5.5 + 1.17 A$ is accurate within ± 1 server for $P(B) < 0.01$ from $5 < A < 50$ Erlangs. Similarly $N = 7.8 + 1.28 A$ approximates $P(B) = 0.001$ engineering within ± 1 server over the same range. The relevance is that design methods developed to solve apparently "static demand" problems also apply directly to the dynamic demand environment, through traffic theory. Simulation studies are not needed to generate the capacity requirements. If one knows the mean traffic intensities (A) that the simulation would have used, these values can be used directly in the equations given to generate the lightpath number requirements for the target blocking levels. Moreover, because N for a constant $P(B)$ is nearly linear with A , the individual O-D pair path requirements can be added on spans that are common to the routes taken for various O-D pairs, thus generating the w_j quantities used for the subsequent span-restorable envelope protection design.

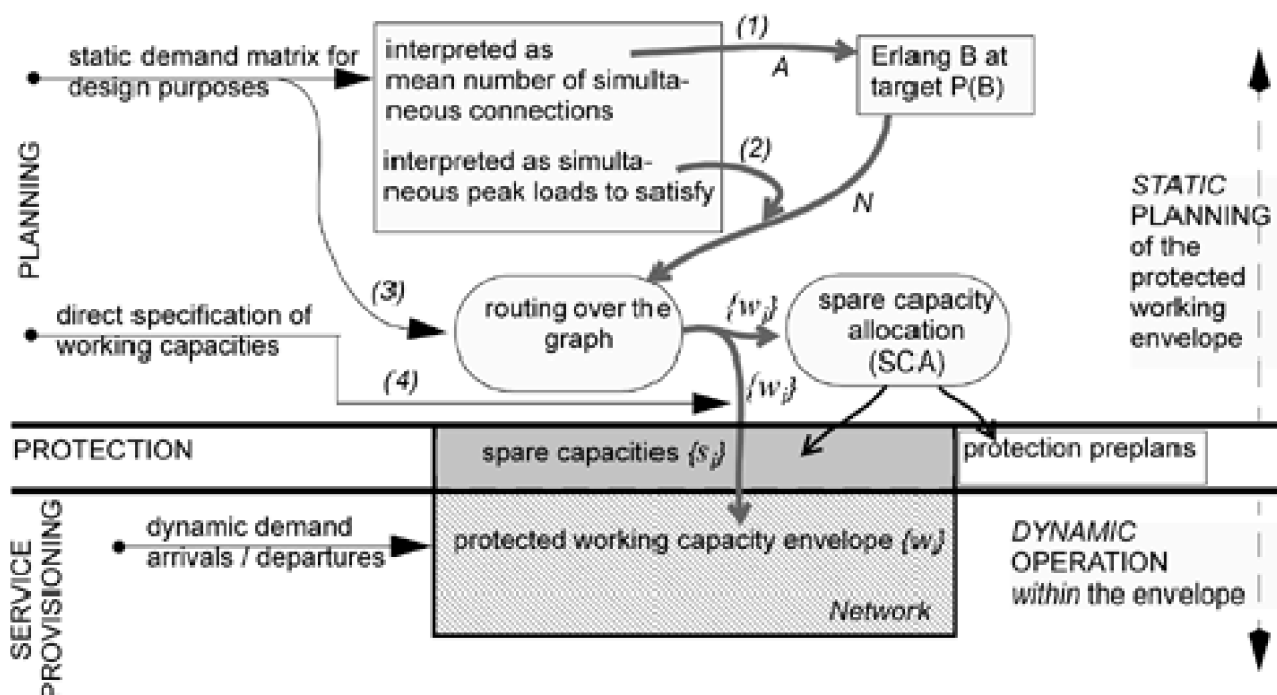
In either case above, a process of shortest path mapping of these requirements onto the network graph generates a family of working capacity requirements on each span. This is in effect sizing the envelope: it fleshes out the number of wavelength channels to pre-provision for working services on each span. A separately solved spare capacity allocation problem (to follow) can then stipulate the additional required number of spare channels to pre-provision so that the entire network envelope of working capacities is protected, regardless of the actual pattern of dynamic demand connections being supported at any time.

[Figure 5-6](#) illustrates the overall relationships. The main horizontal lines divide the space into planning, protection, and real-time service provisioning activities or operations. In planning there are four possible avenues or planning paths that lead to generation of the working capacity requirements on each span:

1. A forecast is made of the average lightpath demand (simultaneously required lightpath connections) on each O-D pair in Erlangs. Actual traffic is understood to be random in time, but characterized by these mean traffic intensities.
2. A specification of the simultaneous peak connection numbers required between all O-D pairs is given. Again traffic is understood to be random in time, but this characterizes the worst-case simultaneous peak connection loads that it is desired to support.
3. A forecast of exact static demand requirements expected between each O-D pair.
4. No forecast of demand is given or used. The amount of provisioned working capacity on each span is specified directly, based on availability or investment considerations and simply represents an inventory of installed capacity with which to provide services.

Figure 5-6. How static planning models support dynamic provisioning: dimensioning the

protected working capacity envelope that supports simplified dynamic path provisioning.



Point 3 is what is referred to as the traditional "static" planning model, which is useful for many study contexts, but does not directly consider the dynamic nature of demand patterns expected in future. Points 1 and 2 are both better descriptions of a dynamic demand environment. And yet in a sense, they are just as static as 1, when the random models used are characterized by their mean or peak intensities, which are assumed to be known. In fact, many studies using simulation methods to model various aspects of network performance under dynamic demand are logically no more models of an *uncertain* future demand than the completely static forecasts because the assumption of known mean traffic intensities per node pair is as predictive as a static demand model. Capacity planning methods used in conjunction with static demand matrices can be just as applicable to dynamic demand contexts if we simply assign new meaning to the demand specifications and understand that the capacity designs that result are dimensioned to provide suitably sized *envelopes for dynamic operation*, rather than capacity allocations to serve only one exact demand pattern. A related point is that true uncertainty about future demand patterns is not addressed by simulation of dynamic random demands to assumed mean traffic measures. In the telephone network every call is a random arrival, but the forecast would be considered perfect if the mean traffic intensities on each node pair were exactly known as in so many recent simulations of dynamic lightpath arrivals. Thus, when some researchers model lightpath demand as Poisson processes of known mean values (or any other random models of known parameters), they are no more addressing uncertainty about the *pattern* of demand when using simulations to generate required capacity than when a static demand model is assumed for a capacity design problem. True planning uncertainty and simple randomness of arrivals are not the same thing.

In the overall concept portrayed in [Figure 5-6](#), each of the different planning approaches leads to a specification of required working channel quantities, w_i , on each span. Whether the interpretation is a forecast of static demands, or a forecast of mean dynamic traffic intensities, both lead ultimately to an as-built requirement that is static once placed on the ground. Once the planning of working quantities is complete, through whatever path is taken above, we enter the protection planning domain. If the w_i quantities are given, then an optimization problem called Spare Capacity Allocation (SCA) dimensions the minimal total spare capacity that will protect all w_i capacity on each span. Usually a side-effect of this process is to also produce all the restoration rerouting plans that define the use of the spare capacity for each planned failure scenario. This information can be downloaded to each node in a centrally controlled scheme but is not required in a network that employs distributed preplanning.

Below the second line in [Figure 5-6](#), the network is viewed in its operational service provisioning phase. The w_i quantities define the resource pool or envelope that is used for dynamic service provisioning. The s_j quantities, protection preplans and protection mechanism are unseen by the provisioning process, but ready if needed. The service provisioning process need not directly address protection concerns at all within its routine of establishing and removing dynamic connection requests. The service provisioning process is using inherently protected capacity, instead of having to explicitly provision protection for each service path it establishes.

The overall process thus dimensions and protects a working envelope of capacity within which dynamic path provisioning operates without any further immediate concerns about survivability. The envelope itself is structurally protected by design. Regardless of the minute-by-minute changes in the actual mix of services and composition of the bandwidth allocation in links of each span, every service provisioned through the envelope of protected capacity is inherently also protected. With the protected working envelope concept, protection preplans are far less dynamic than the traffic itself. Only when the *envelope* requirements change do the protection plans

possibly need changing. But evolution in the envelope requirements occurs only in response to evolution of the total demand pattern itself, not "call-by-call" events. It takes a shift in the *statistics* of the demand pattern to require a logical change in the working envelope. And it takes a large shift in total demand or its pattern to ultimately require additional spare capacity. But envelope changes occur on a very different time scale than the underlying call-by-call events themselves and, importantly, exhibit correlated observable trends that can drive appropriate ongoing planning processes.

Thus, the envelope and its protection plans are only slowly changing—or even static—over long periods of time, even in the most frenetically dynamic network. No matter how rapidly individual lightpath demands come and go at random, the *envelope* requirement will not change at all if the system is at statistical equilibrium. The envelope is only sensitive to non-stationary drift in the underlying pattern of random arrival/departure processes. Which, thus, is truly a more "scalable" architecture for handling a dynamic demand environment? A span-protected working capacity envelope or a scheme where every arrival requires its own individual protected arrangements, coordinated in real-time with every other (equally dynamic) path in the network? The envelope needs to track only non-stationary drift in the statistics, not each arrival and departure event individually. This is highly advantageous compared to the incrementally provisioned path model under which protection preplans have to be updated following each new service path establishment. It makes a span-protected WCE network considerably more scalable than a corresponding SBPP network in terms of growth in both network size and service provisioning volumes handled per unit time. Under SBPP every connection establishment requires explicit consideration of a working path and a disjoint shared-backup path arrangement based on global network topology and shareability data. Under a span-protected WCE every connection set-up is simply a single shortest-path routing problem within available w_i capacities of the protected envelope. We think this is one of the most important advantages of span-oriented protection, but to our knowledge, the case for WCE-based dynamic provisioning has never been argued before.

More formally, we can define the state-space of all demand patterns (instantaneous combinations of connection states) that are inherently protected by a given working capacity envelope as follows.

- $G(N,S)$ is the network graph comprised sets of nodes N and spans S .
- w_j is the number of working channels on span $i \in S$ which are protected under the current distribution of spare channels on other spans $j \in S, j \neq i$. It is a separate problem to determine w_j by any number of means depending on the protection mechanism, the network graph, and the spare capacity distribution. Suffice it here to say that the maximum number of working channels that can be protected by any given reserve network is computable yielding $W = \{w_i | i \in S\}$ which is the network's *protected working capacity envelope*.
- D_k is the matrix of point-to-point demand requirements in demand scenario k . There is an essentially infinite number of possible demand scenarios.
- $M(G,D_k)$ is a process or algorithm for working path routing. It takes the graph G and maps the demand matrix D_k onto routes through graph. The output is $w_{i,k}$ —the number of working channels on span i employed by $M()$ to map demands of demand scenario k . $M(G, D_k)$ can be invoked as a function to form the set $A_k = \{w_{i,k} | i \in S, k \in K\}$. These represent the actual usage of working channels on span i when demand pattern k is routed over the graph by process $M()$.

It is feasible to entirely serve demand matrix k through the current envelope W if $w_{i,k} \leq w_i \forall i \in S$. If this condition is true, we can say that D_k is "inside" envelope W under routing process $M()$ —a relationship we can denote as $M(G,D_k) = A_k \subseteq W$ or just $M(G,D_k) \subseteq W$. Then the state-space for automated provisioning is definable as the set of all demand matrices D_k that are served and protected "inside" the working capacity envelope. That is:

Equation 5.1

$$\mathcal{D}_W = \{D_k | (M(G, D_k) \subseteq W)\}.$$

By this we mean that \mathcal{D}_W is the set of all possible demand matrices where every demand is served and protected within the working capacity envelope W using a given process $M()$ to route demands. Spare capacity does not come directly into these considerations even though all services are protected. This is the simple beauty of the WCE concept: dynamic service provisioning has to consider only working path routing. There are no explicit considerations about spare capacity allocation or protection path arrangements because the

envelope is by definition a protected operational working-space. As long as we operate within the working capacity envelope, we are automatically protected. The "stress" or operating proximity to limits of the envelope can be monitored at all times—the vector of operating margins is $\{w_j - w_{i,k}\}$ —but nothing needs to be done on the time scale of the individual path arrivals themselves.

In this framework every sequence of random demand arrivals and departures can be seen as a random-walk trajectory from one point k to a next point k in the space D_W . The entire operating life of a dynamic network consists then only of single steps within this $N(N-1)/2$ -dimensional space. Each step goes from a current D_k to an immediate neighbor D_{k+1} state where one connection in D_k is removed to reach D_{k+1} or to a neighbor state where one new connection is added to D_k to arrive at the next D_{k+1} . At all times during the operating walk, the routing process proximity to the envelope is measurable via the working capacity margins $w - A_k = \{w_j - w_{i,k}\}$. At all times operation consists only of releasing paths or routing new working paths. The paradigm for handling dynamic demand is thus simplified to the equivalent of routing working demands on a point-to-point basis as if they were in an unprotected network of span capacities $\{w_j\}$. In addition, any step to a neighbor in D_W involves at most a unit change in any $w_{i,k}$ value so the potential onset of blocking is always observable (and easily employed to alter the behavior of $M()$ itself to avoid blocking, or, moreover as we consider next, to adapt WCE itself).

5.2.6 Demand-Adaptive Definition of the Working Capacity Envelope

As so far described, we have a WCE defined by a fixed set of w_j capacities. A planning process as outlined would determine w_j values of the required envelope, and this number of working channels would be turned up on the respective spans, commissioning the operational envelope that will be used for dynamic service provisioning. (A lesser number of s_j channels is also computed and turned up to protect the desired operating envelope). More generally, however, pre-existing transmission capacities may be in place that exceed the minimum ($w_j + s_j$) channel requirements of the envelope design. This adds to the operational scope for the WCE concept because it means that there is latitude for the partitioning of total capacity into w_j and s_j quantities to be adapted to suit evolving statistical traffic patterns. In other words, we can configure different working envelopes as long as the spare capacity needed to protect each is feasible under the installed total capacities. This leads to what can be called a dynamic WCE mode of operation. The main difference is that the partitioning of the total provisioned capacity into working channels to form the WCE and spare channels to protect it can be *adapted* by the network operator, but on a much slower time scale than that of individual connection requests. In fact, the simplest mode of operation is to let the routing process $M()$ use as much w_j as it wishes on each span to realize required connection patterns, up to the point where a separate "observer" process indicates that a not-fully-protected state would be entered for if one more channel was seized on span i . To consider this context, let us define the following, in addition to terms above:

- t_i is the total number of equipped channels on span i . $T = \{t_i \mid i \in S\}$ is the set of all such installed capacities.
- $u_{i,k} = t_i - w_{i,k}$ is the number of equipped channels on span i (including any explicitly designated as spare channels) that are not used by the current demand scenario, k .

$U_k = \{u_{i,k} \mid i \in S\}$ is the set of all currently unused capacities.

- $k_i(\{G-i\}, U_k)$ is the maximum number of paths currently feasible through the graph of unused channels U_k , excluding span i . In other words, $k_i()$ is a function that returns the maximal feasible number of restoration paths for span i in the current network state. In practise this could be a max-flow or ksp algorithm or obtained from a "mock restoration trial" call to a DRA embedded in the network.
- $P(G, U_k, A_k)$ is the boolean decision problem that the network observer process executes:

Equation 5.2

$$P(G, U_k, A_k) = \begin{cases} 1 & \text{if } k_i(\{G-i\}, U_k) \geq w_{i,k} \quad \forall i \in S \\ 0 & \text{otherwise} \end{cases}$$

In other words, $P(G, U_k, A_k)$ is the yes/no answer to the question: Are all the actually used working channels $\psi_{i,k} \in A_k$ in the current operating state fully restorable under the unused capacities $u_{i,k}$ remaining in the network? In these circumstances the dynamic operating state-space for automated provisioning is enlarged to become:

Equation 5.3

$$\hat{D}_T = \{D_k | (M(G, D_k) \subseteq T) \cap (P(G, U_k, A_k) = 1)\}.$$

The operating state space is comprised of all demand matrices D_k , which are routable in the available total capacity T (the first qualifier) and for which the working channels used on each span remain fully restorable (the second qualifier) under restoration routing that would use only the unused channels. Intuitively, [Equation 5.1](#) and [Equation 5.3](#) both define a vast domain of feasible protected operating states. The first scheme is simpler in that the WCE is effectively fixed, reflecting a single partitioning of available capacity into working and spare channels. An off-line process could periodically redesign the WCE to track demand evolution in this scheme, but no restorability observer is required. By addition of the on-line restorability observer the second scheme is enabled to continually exploit *all* installed capacity, implicitly adapting its own WCE to the track variations in actual demand patterns to the greatest extent possible. The adaptive WCE scheme amounts to being a network that is a self-planning demand-adaptive mesh restorable network which also monitors its own operating margins.

The restorability-assessing observer process can be either a centralized computation or a DPP-type of background process which runs mock restoration trials in the unused "spare" capacity of the network itself. Each DPP trial provides the currently feasible maximum number of restoration paths for the simulated failure, that is $k_i()$ in [Equation 5.2](#). The restorability decision problem $P(G, U_k, A_k)$ is therefore not a difficult one and a side effect of its execution is *restorability margin* information that lets a network operator see in advance if the limits of the protected envelope are being approached on any particular span. As an example, using k -shortest paths ([Chapter 4](#)) as the route finding process for restoration:

Restorability Observer (G, U, A): P, margins()

```
enter with graph G, unused capacities U, and actual-use capacities A;
for every i in S: {
    source[i] := first end node of span i;
    target[i] := other end node of span i;
    G' := G - {i}; //G' is the reserve network without span i//
    k[i] := ksp (G', source[i], target[i], U);
    if (k[i] < w[i]) then {P=0; return};
        //unrestorability detected:(span i)//
    else margin[i] := k[i] - w[i,k] }
P=1;
return (with margins) //current state is fully restorable//.
```

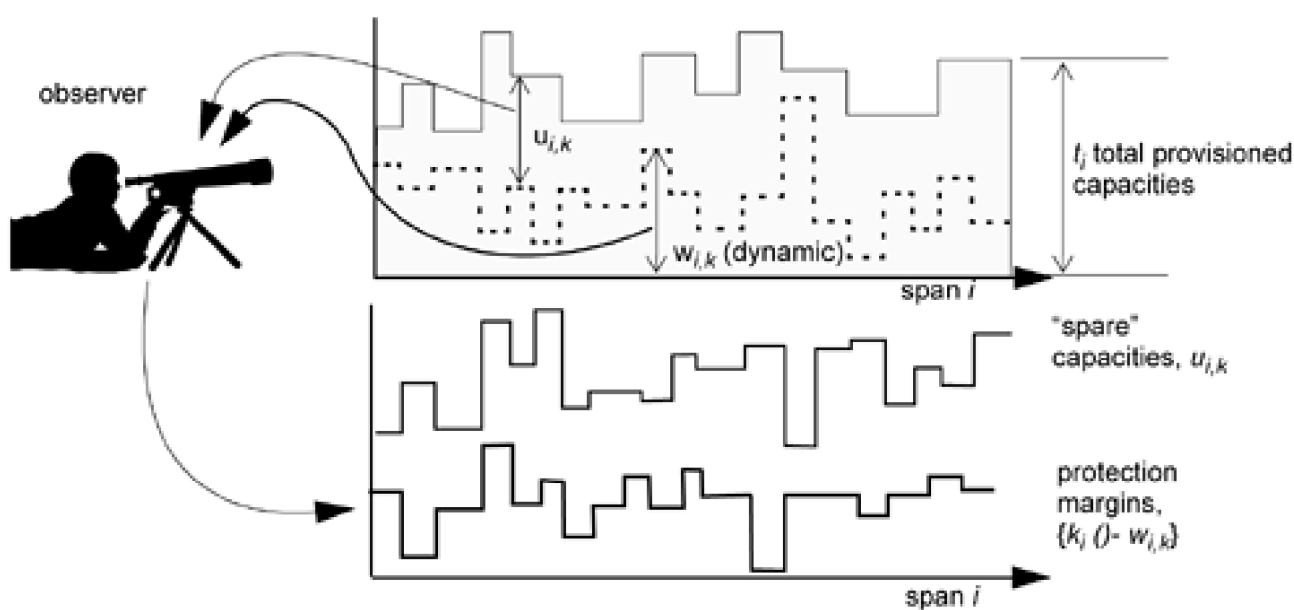
The restorability observer is not run for every single path provisioned. Its job is rather only to sample and track the overall operating state, highlighting any spans where the restorability margin (in effect the remaining potential for expansion of the dynamic envelope) on a span is below some threshold. If the total number of working paths routed over a span just reaches $k_i()$ then that would be the last service path that can be provisioned over that span within the protected envelope. The next path may have to take another route or be blocked. This is therefore the WCE manifestation of blocking that similarly occurs in SBPP networks when a disjoint primary and backup path pair can no longer be found. But an important difference is that the onset of such blocking is observed in advance by the restorability observer process. If the demand pattern really is evolving (shifting its statistical means) so as to stress the envelope in one direction (one span) this evolution is seen in advance as the slow erosion of the average restorability margin on the respective span. So capacity augmentation efforts and/or changes in routing policy can be effected proactively rather than in response to the hard onset of actual blocking.

Rather than simply starting to block certain new connection requests, changes in the routing policy $M()$ can avoid or delay the

encroachment on restorability limits on certain spans, or priority service class policies can also be effected to soften or avoid an approach to the envelope limits. In contrast under SBPP each end node pair runs its own process for primary (working) and backup path protection arrangements and so can suddenly enter blocking conditions without any warning or advance knowledge as the total capacity is consumed by other service path pairs in the same network.

Figure 5-7 summarizes the overall concept of a WCE that is self-sizing within the available total capacity and restorable state boundaries of the network and is observed either centrally or via DPP to define the working capacity limits of its dynamic provisioning operations. Note that the protection margin on a span is not the difference between its own working and spare capacity. Rather, in span restoration it is spare capacity on other spans that protects a given span, so the protection margin is the difference between the maximum number of feasible restoration paths for span i , i.e., $k_i(\cdot)$, and the current working capacity usage on span i , i.e., $w_{i,k}$. So a span that has a lot of spare capacity may itself not have a high protection margin. With the observer, the approach to states where new service paths would potentially have to be blocked is "soft" and may be avoided altogether if a statistical trend in a certain direction is temporary, and not sustained. If the observer reports that restorability margin has eroded below certain limits on a span, further routing can be discouraged, but not ruled out, on that span until the margin rebuilds. If an OSPF/CR-LDP type of routing process is being used then it would suffice for the observer to simply publish an updated "edge cost" for any span whose restorability margin was below some threshold. Similarly, by lowering edge costs it can encourage new paths to take routes over spans with high restorability margins. If centrally computed, the restorability observer task is in $O(|S||N|^2)$ (based on $O(|N|^2)$ for the *ksp* algorithm). However, a restorability observer that is tracking the $w_{i,k}$ state can use a pre-prepared route table to answer the question $P = 1$ or 0 ? in $O(1)$ (i.e., table-lookup time). A table-based method for such fast calculation of the current span restorability state is described in Venables [Vena92].

Figure 5-7. The Working Capacity Envelope concept with dynamic adaptive boundaries—coping with true *uncertainty* about demand, not just its randomness.



5.2.7 Concept of First-Failure Protection and Second-Failure Restoration

Another operational concept that is attractive about span-restorable architectures is the concept of *first-failure protection and second-failure restoration* (1FP-2FR) with an embedded DRA. This builds on the DPP concept, recognizing that the DRA embedded in the network can:

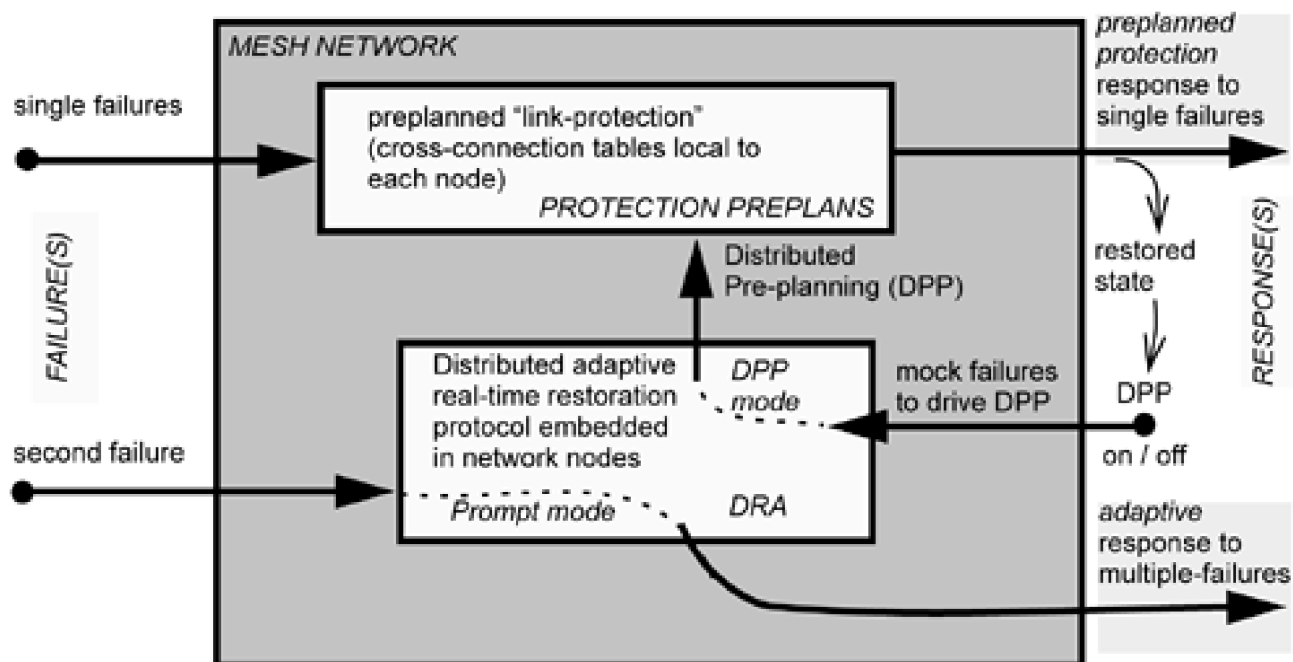
1. Be used continually (or periodically) in the background to generate or update the basic single-failure protection preplans. We can call this *DPP mode execution* of the DRA.
2. Also be invoked directly for state-adaptive real-time restoration against dual-failures or any situation for which the preplanned protection reactions did not yield 100% recovery. We call this *"prompt" mode execution* of the DRA.

The idea is that response to a first failure can be based on fully preplanned fast protection reactions but that the underlying adaptive restoration capability is always available on-call in real-time if needed as well. Usually the DRA is executing in the background to exercise and update the self-healing responses to a series of mock failure trials. This continually updates and adapts the protection plans just by each participating node recording the cross-connections it would have made for each mock-failure trial. As soon as a real failure occurs in the network, mock trials cease and a simple flooding indication alerts all nodes. Nodes locally activate the corresponding spare capacity cross-connection maps from their DPP table. The same alert advises all nodes that the network is now in a single-failure-protected state. While in this state, existing DPP tables are not relied upon further. Following entry into the single-failure protected state, DPP planning trials may begin again, developing a set of second-failure response plans and at the same time observing the restorability levels achievable for each possible second failure. In this state the DRA is also on standby for direct real-time action if needed in the event of a second failure before the DPP process has created new second-failure pre-plans everywhere. Alternately, during the single-failure protected network state, DPP may be suspended altogether and the DRA simply armed for direct adaptive restoration action should a second failure occur during the repair interval. In the latter case, even if the direct execution of the DRA is slower than speed of activation for in-place DPP preplans, it is the state-adaptive nature of the DRA executing in its direct action mode that is of considerable value in facing a second failure.

Thus, the overall picture is one of a network that self-plans its first-failure reactions and responds reactively, but adaptively, to a second failure. The DRA serves as the engine for constant background preplanning of a fast protection reaction against single failures. In the event of a second failure (or any time the protection level is not 100% such as if a dual failure hits directly), the DRA executes directly in real-time where its completely adaptive nature to the outstanding failure channels and available spare capacity is exactly what is required to produce the highest possible overall recovery level. This gives a significant availability enhancement, especially for priority-designated paths under multiple failure scenarios where the overall recovery level (even with the fully adaptive second response) is not expected to be 100% due to simple capacity limitations. This is an important advantage over centralized preplanned protection schemes or SBPP-type backup planning schemes where there is no automatic adaptive restoration response on call for dual-failure situations.

[Figure 5-8](#) illustrates this additional role and value of a distributed restoration algorithm (DRA) embedded in every node. Having the recourse to call on the DRA either for self-planning or for prompt direct action is like having two layers of backup. In [Chapter 8](#), where we consider the dual-failure availability analysis of restorable mesh networks ([Section 8.4.4](#)) we will see that this double line of defense can be the basis for ultra high availability services with no more spare capacity than required for 100% single failure restorability.

Figure 5-8. The span-restoration DRA provides an adaptive real-time response to backup the protection preplans against multiple failures giving ultra-high availability.



5.3 Spare Capacity Design of Span-Restorable Mesh Networks

Having introduced the span restoration concept and touched on the operational advantages and options that a span-restorable networking approach provides, let us now consider the problem of routing and capacity design for such networks.

5.3.1 Overview of the Problem

To start with, imagine a restoration rerouting mechanism that was highly adaptive and would make the best use of any available spare capacity that you gave it to restore signal paths against failures. An interesting question is then: how would you decide just where and how much spare capacity to invest so that such a mechanism would be able to achieve 100% restorability, but with the minimum of total spare capacity possible? In other words, how could you be as "stingy" as possible and still let the restoration mechanism achieve the intended goal? This is essentially the problem of spare capacity design for a mesh-restorable network.

The first basic problem is that of assigning spare capacity to spans for 100% restorability to any single-span cut assuming that the working demands have already been routed over the graph. Depending on the source, this problem is referred to as the *spare capacity placement* (SCP) problem, e.g., [\[VeGr93\]](#), the *reserve network* design problem, e.g., [\[Wyna01\]](#), the *spare-channel* design problem, e.g. [\[SaOk92\]](#) or the *spare capacity assignment* (SCA) problem e.g., [\[HeBy95\]](#). We will refer to it henceforth as SCA.

A statement of the basic SCA problem is as follows: Let a network $G(N,S)$ having set S (or E) of spans and N of nodes, and a set of working capacities, w_j on each span. The w_j quantities may arise from shortest path routing of the working demands over the network graph, or from any other demand routing process. For now, the w_j are given as inputs. In effect, each w_j represents the number of replacement paths that the spare capacity assignment must support for each respective span cut. The SCA problem is then to specify spare capacities, s_j , so that the total cost of spare capacity is minimized, while for every span j taken one at a time as a failure span (more generally for every defined failure scenario), the number of replacement paths through the surviving spares of the network, meets or exceeds the number of failed working channels on span j . The restoration paths are circuit-like connections each using an individual spare channel on each span along their route. w_j and s_j must be whole numbers.

Formally, SCA is an integer capacitated network design problem for non-simultaneous single-commodity flows. Historically, the problem of network synthesis for non-simultaneous *multi-commodity* flow has arisen when considering minimum cost allocations of (working) capacity to satisfy a family of different demand patterns arising in different time periods [\[Mino81\]](#) [\[GeCr99\]](#), also known as multi-hour traffic engineering. In our case it is not a time varying demand matrix that creates the various flow patterns, but the fact that each failure scenario presents a new subset of the demand matrix to be restored by routing through the spare capacities of the surviving portion of the reserve network. And each non-simultaneous failure in SCA for the most common form of span restoration problem is of a single-commodity nature but there is one such flow problem for each failure scenario.

5.3.2 Basic Node-Arc Formulation

Various network flow problems in [Chapter 4](#) were formulated as "node-arc" models. This means that the flow variables are associated with each edge of the graph as opposed to association with specific paths over the network, or, with cutsets of the network, which follow. This is the classic approach to network flow problems because it arises from the transportation problem ([Section 4.15.1](#)) and its special property of total unimodularity. Let us, therefore, start with a node-arc version of SCA.

SCA (node-arc)

Minimize

Equation 5.4

$$\sum_{\forall (i,j) \in S} s_{i,j} \cdot c_{i,j}$$

subject to:

Equation 5.5

$$\sum_{\forall j \in N | (i,j) \in S, (i,j) \neq (s,t)} (x_{i,j}^{s,t} - x_{j,i}^{s,t}) = \begin{cases} w_{s,t} & \text{if } i = s \\ -w_{s,t} & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N \quad \forall (s,t) \in S$$

Equation 5.6

$$x_{i,j}^{s,t} \leq s_{i,j} \quad \forall (i,j) \neq (s,t) \in S \quad \forall (s,t) \in S$$

Equation 5.7

$$x_{i,j}^{s,t} = x_{j,i}^{s,t} \geq 0 \quad \forall (i,j) \in S \quad \forall (s,t) \in S$$

Equation 5.8

$$s_{i,j} \geq 0 \quad \text{integer} \quad \forall (i,j) \in S$$

Recall that in a node-arc formulation it is nodes that are indexed by the subscripts. Edges are specified by an (i,j) node-name pair. Here

(s,t) denotes the end-nodes of the failure span. $x_{i,j}^{s,t}$ is the (directed) flow on arc (i,j) (direction from i to j) in response to the restoration requirements of edge (s,t) as a failure span. $c_{i,j}$ is the cost of a unit capacity on (i,j) and $s_{i,j}$ is the spare capacity assignment to edge (i,j) .

The first two parts of [Equation 5.5](#) assert full restoration of each failure scenario by making the total flow on edges incident on the source and target nodes equal to the working capacity $w_{s,t}$ of the failed span. The third part of [Equation 5.5](#) is the classic expression of flow

conservation (also called trans-shipment) at all of the "tandem" nodes relaying the restoration flow. The next constraint system ([Equation 5.6](#)) sizes the spare capacity allocations on spans to implicitly support the largest restoration flow across each edge over all (s, t) failure scenarios. [Equation 5.7](#) asserts equal bidirectional restoration flows on each edge for each failure scenario.

If the set of failure scenarios is all single-span cuts, this formulation will have $2|S|(|S|-1)$ constraints in $\sim |S|^2$ variables. Compared to the arc-path and cut-oriented approaches that follow, this can be the most compact model, especially for "sparse" networks (those where the number of spans is only somewhat more than the number of nodes, i.e., where $\bar{d} \approx 2 + \epsilon$).

Because of its transportation-like structure it is reasonable to enquire if SCA is totally unimodular (TU). It turns out that SCA by its structure alone does not assure total unimodularity but that in practice many instances of SCA will exhibit the TU property. To appreciate the overall structure of SCA note that, taken individually, each specific failure scenario poses an instance of a min-cost flow problem. Alternately if the edge capacities were not sufficient to support 100% restoration of the failure, each subproblem would be an instance of a max-flow problem aiming for the highest restorability possible with the given spare capacities. Thus, one point of view is that SCA is like an ensemble of MCNF subproblems linked together under a requirement for a spare capacity allocation that makes each of them feasible at the required restoration flow level. This view is only approximate, however, because the minimum overall cost of spare capacity need not necessarily employ the *minimum* cost flow solution for any one of the individual failure scenarios. In fact this would be counter to using routing decisions for each failure that make the sharing of a single set of spare capacities highest. There is no need for any individual flow patterns to be MCNF-like: rather there must only be a feasible routing of the required total flow under the spare capacity. TU is not thus inherited directly by structure from the transportation problem or MCNF.

Nonetheless several workers including ourselves, Wynants [[Wyna01](#)], Kennington [[KeLe98](#)] and Sakauchi et al. [[SaOk92](#)] have observed that relatively few fractional flows and spare capacity values tend to arise in SCA (if relaxed to an LP) and that they are easily "repaired" if they do. Wynants has also shown that the integer SCA problem is "strongly NP-complete" but that it is also of a type where, given an integer solution to the spare capacity, we can always *construct* a feasible integer solution to the restoration flow variables [[Wyn01](#)] (p. 36., p.20). It is for that reason that we show SCA above as requiring integer capacities, but only non-negative (i.e., relaxed) restoration flow variables.

One practical drawback of the node-arc formulation is that the flow variables do not directly prescribe the routes to be taken by each restoration path. Restoration flows are also inherently unconstrained in terms of the range they may take over the graph. This makes it difficult to design in considerations such as a restoration distance limit or specific behavioral models for restoration routing characteristics. Especially in optical networks, it may be important for signal quality reasons to make sure restoration occurs over a specific set of pre-qualified or "eligible" restoration routes for each failure. This is the strength of the arc-path design model.

5.3.3 Basic Arc-Path Formulation

Any node-arc network flow problem can also be formulated in a corresponding "arc-path" model (or "arc-chain" or "edge-path" formulation, depending on the source). The arc-path approach to SCA is posed as one of assigning restoration flow for each failure scenario to a set of *eligible routes* so that all flow assignments are feasible under a minimum total of spare capacity. Where routes with simultaneous flow assignments share a span in common, the restoration flows superimpose and push up the spare capacity requirements on that span. If one considers the other non-simultaneous span failures that may occur, the flow assignment for one failure may be more efficient if it takes into account spare capacity already required on spans due to larger flows on routes crossing them for other failures. Thus, one gets a picture of the spare capacity variables being pushed up by each flow assignment for individual failures, and of the routing assignments being made in a way that tries to avoid pushing up the spare capacity to new maximums to support the imposed flows. In an arc-path model the solution will be in terms of flow variables associated with routes. An initial expression of the arc-path version of SCA is:

SCA (arc-path)

Minimize

Equation 5.9

{ }

$$\left\{ \sum_i c_i \cdot \max_{i \neq j} (f_{i,j}) \right\}$$

subject to:

Equation 5.10

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

Equation 5.11

$$f_{i,j} = \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p \quad \forall (i,j) \in S^2 | i \neq j$$

Equation 5.12

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i$$

Equation 5.13

$$f_{i,j} \geq 0 \quad \text{integer} \quad \forall (i,j) \in S^2 | i \neq j$$

As an arc-path model the indexing is now on spans rather than nodes. Generally, henceforth when indexing spans we will tend to use i to designate a span in the context of a being failure span and j to designate other, surviving, spans in that failure scenario. P_i is the set of all distinct routes that may be used for restoration of failure i . These are subsequently referred to as the "eligible routes" for restoration of

span failure i . $\delta_{i,j}^p$ is an input parameter which is 1 if the p^{th} route available for restoration of failure i includes span j , and 0 otherwise.

f_i^p is the restoration flow assigned to the p^{th} route available for restoration of failure i . $f_{i,j}$ is the total restoration flow over span j in response to the failure of span i . In this formulation the $f_{i,j}$ variables (the largest of which over all i effectively sets the required s_j) are integer

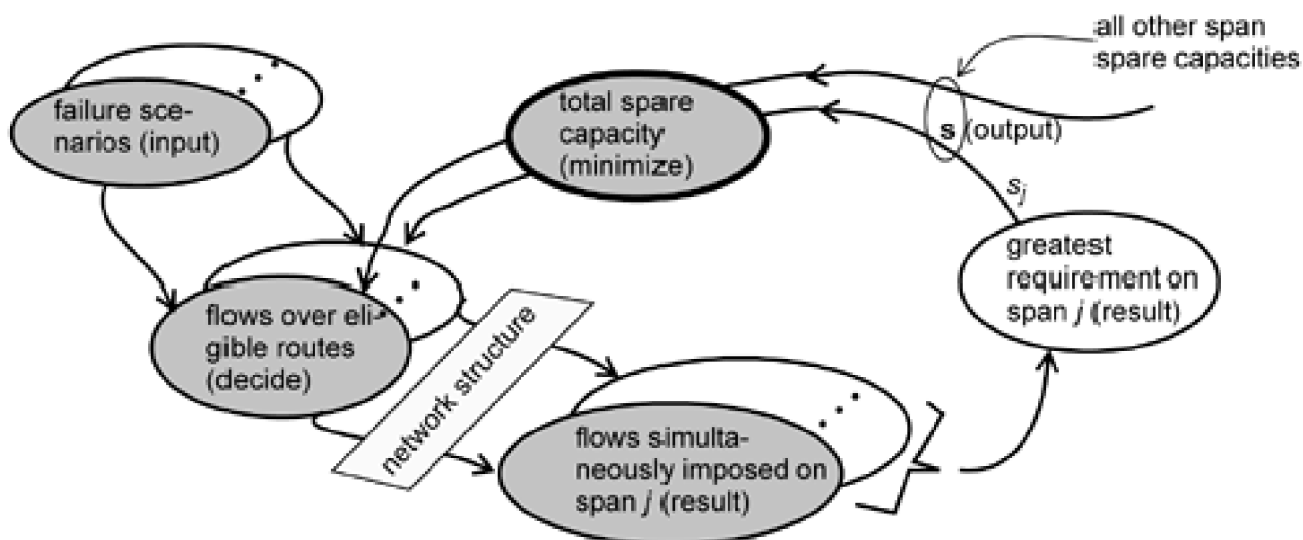
but, as above, we can relax the restoration flow variables. f_i^p

The objective function is equivalent to $\sum c_j \cdot s_j$ but we write it in the manner shown to convey more understanding about the problem structure. The point is that each s_j quantity is determined by the largest restoration flow $f_{i,j}$ over that span taken over the set of all failure scenarios not involving that span itself as a failed element. Thus, the spare capacity assignment to each span j arises in general from a different finite-flow subproblem, i.e., that for some other span i which happens to require the largest restoration flow over j .

A basic appreciation of SCA is fundamental to many subsequent mesh-restorable capacity models, so we devote [Figure 5-9](#) to portraying the relationships involved in SCA. Each failure scenario implies a requirement for a certain total restoration flow (equal to w_i) to be realized

over the routes $p \in P_j$ eligible for use in restoration of that failure scenario. The detailed assignment of flows to routes (f_i^p) to realize this total requirement interact with each other through the graph topology (represented by $\delta_{i,j}^p$ parameters) generating the total flows $f_{i,j}$ that will cross each other span j , under each failure scenario. The largest of these, over all failure scenarios, is the s_j requirement for span j . The flow assignment decisions are the decision variables that are searched to lower the total spare capacity. Later, there are important properties of span-restorable networks, such as their innately high dual-failure restorability, and hybrid design efficiency with rings, that can only be fully understood in reference to the fact that it may in general be only one specific failure scenario that winds up setting the $\max(f_{i,j})$ quantity that dimensions the required spare capacity.

Figure 5-9. Understanding the span-restorable mesh SCA problem.



5.3.4 Herzberg and Bye's Hop-Limited Formulation

The arc-path orientation to the SCA problem was used by Herzberg and Bye [HeBy94] for span-restorable SCA where the working demands are already routed and a "hop limit" concept is introduced to define the set of eligible restoration routes. The introductory explanation of the arc-path model above assumed that all distinct routes were contained in each P_i set, for the respective failure scenario. Enumeration of all distinct routes is, however, exponential in complexity with network size. As mentioned, the complete set of distinct routes may also include routes that exceed desirable length limits or violate other transmission objectives. Herzberg therefore proposed to limit the size of the eligible route sets according to hop limit considerations for restoration. This recognizes that in practice there are length limits above which echo-delay phenomenon are aggravated for voice connections, and that there are other engineering desires to limit the maximum length of a rerouted signal path (for instance, optical signal path quality and limiting the number of steps of signal path reversion after physical repair, etc.). In centralized control short restoration routes also reduce the number of commands to be downloaded to effect restoration, speeding the process up. However, Herzberg also showed the SCA problem exhibits a threshold effect in terms of increasing hop limit beyond which optimal capacity designs are obtained even though the complete set of distinct eligible restoration routes is *not* provided for the solution. The formulation also uses a set of explicit spare capacity variables and corresponding inequalities to replace the $\min(\max)$ form of objective function:

SCA (arc-path)

Minimize

Equation 5.14

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

Equation 5.15

$$\sum_{p \in P_i} f_i^p = R_i \cdot w_i \quad i \in S$$

Equation 5.16

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p \quad \forall i, j \in S^2 | i \neq j$$

Equation 5.17

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i$$

Equation 5.18

$$s_j \geq 0 \quad \text{integer} \quad \forall j \in S$$

Restoration flow variables f_i^p and restoration route sets P_i are as in the arc-path SCA above but now we introduce integer spare capacity variables s_j . The objective function is now the cost-weighted sum of all spare capacity. This model has integer spare capacity variables but relaxes the restoration flow variables under them, so it is a true mixed-integer programming (MIP) model. There are $|S|^2$ constraints ($|S|$ from [Equation 5.15](#) plus $|S|(|S|-1)$ from [Equation 5.16](#)) and $|S| + \sum |P_i|$ variables. This model also provides for restoration target levels R_i . For simplicity in going forward we will not, however, explicitly represent R_i further. In effect we will treat all problems as ones in which the target restoration level is always 100% ($R_i=1$) for each failure. In practice $R_i < 1$ can be included in any of the design models if partial recovery levels are intended.

If the SCA problem is too large for direct MIP solution, Herzberg also gives a procedure to round up, then "tighten" the spare capacity variables in a complete LP relaxation of the problem. First every non-integral s_j value is rounded up to the next higher integer. *Amax-flow* routine is then used to check, for every failure scenario, whether the feasible restoration flow in the initial non-integral design exceeds its restoration requirement by enough that we know it cannot be affected by the subsequent spare capacity tightening phase. The criterion for such a classification is:

Equation 5.19

$$\text{maxflow}(i) - R_i \cdot w_i \geq \sum_{j \in S} \{s'_j - \max_{i \neq j} (f_{i,j})\}$$

where $\text{maxflow}(i)$ is the maximum feasible flow between the end nodes of span i , excluding span i itself, in the original design where spare values may be real-valued and s'_j are the rounded-up amounts. When true the criterion identifies spans which are not of concern for a loss of restorability in the subsequent tightening phase because if (5.19) holds, then the feasible flow already exceeds the greatest amount of capacity that the tightening phase might subsequently remove from the network. That is to say that the excess capacity any tightening process might remove is bounded by the original real-valued maximum of the imposed restoration flows on each span. Spans that do not satisfy (5.19) have to then be considered in more detail for possible impact from the tightening phase. If the latter spans are designated in a subset S_2 and the set of all spans is S , then the tightening procedure (after rounding up) is as follows:

tighten(S , S_2)

```
for every span  $j$  in  $S$  {
     $s_j := s_j - 1$ 
    for every span  $i$  in  $S_2$ :  $i$  not equal to  $j$ 
        if  $\text{maxflow}(i) < R_i w_i$  then  $s_j := s_j + 1$  }
```

The procedure tests each possible unit capacity removal off of network spans. In each tentative removal state, it further tests the restorability of each span in S_2 as a prospective failure span. If any span in S_2 fails to meet its restorability target, the removal cannot be accepted. (The process is essentially the same as *add_0sub_1* in the tightening phase of the SLPA heuristic which follows.) In [HeBy94](#) a max-flow LP was used to test for feasibility of the required restoration flows under each single-unit capacity removal state, but *aksp* routing solution could be substituted, especially if the latter is a better model of the actual restoration mechanism than max-flow. Following the initial rounding up of fractional s_j values the tightening process provides an integral set of s_j that usually has the same objective value as the original LP solution.

5.3.5 Large-Scale Practicality of Hop-Limited Arc-Path SCA

The size of an SCA problem tableau (in terms of number of variables and constraints) obviously grows with the number of nodes and network average degree, but hop limit considerations can help manage the problem complexity, as well as reflecting the engineering limitations mentioned. The initial use of hop limit criteria in this way will be later seen to grow into a whole family of custom tactics to selectively enumerate eligible route sets.

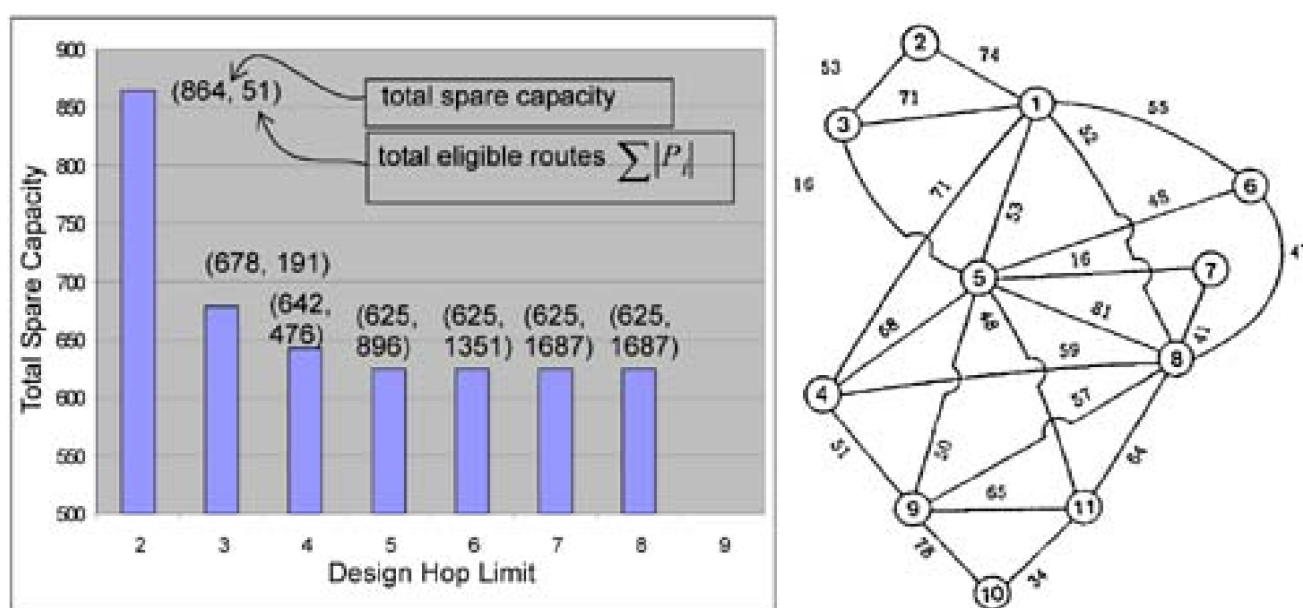
To obtain the set of all distinct routes up to some hop limit H one can use the depth-first search approach for all distinct routes in [Section 4.5.1](#). For a hop limit of H in a network of average nodal degree \bar{d} there will be $H-1$ stages of the search tree, each with average fanout of \bar{d} , making the finding of required route sets in $O(S(\bar{d}-1)^{H-1}) \approx O(N(\bar{d}-1)^H)$. In the face of this, the good news is that in practice both \bar{d} and H are fairly strictly limited. Herzberg illustrates with an example formulation for a 100 node network with $\bar{d}=4$ (implying $|S|=200$ spans) at a hop limit of $H=7$. This produces $\approx 40,000$ constraints (i.e., $O(S^2)$) and $\approx 260,000$ variables (i.e., $|S| + |S| |P_i, H|$). While an ILP at this scale may be difficult, this size of LP is fairly easily solved in a package such as CPLEX. The repair procedure above can then be used if strictly integer flows and capacities are needed, but often in comparative planning studies or research the objective value may be all that is needed in any case. Thus, Herzberg's approach offers a practical design capability. In fact the arc-path approach using selectively enumerated eligible route sets will be the basic approach we use henceforth for many other mesh capacity-related design

issues.

5.3.6 Concept of a Threshold Hop Limit

The concept of a *threshold hop limit* is also highly relevant to managing problem size and representing eligible route sets. The threshold hop limit (H^*) is a hop (or length) limit above which the SCA solution will be optimal even though the eligible route sets do not contain all possible distinct routes. This behavior is illustrated with data from [HeBy94] represented in Figure 5-10. The network for this result is shown in the right panel of Figure 5-10 with w_{ij} values annotated on edges. It is a widely used test case of 11 nodes and 23 spans and is considered fairly typical of metropolitan area networks. For this network, the threshold hop limit emerges at $H^*=5$. This means that no restoration route has to be longer than five hops to achieve full restorability with the minimum of total spare capacity. The number of eligible routes in the corresponding SCA formulation would keep going up if a higher H is allowed, but no further improvements in restoration flow assignments will be found that can reduce the total spare capacity further than with $H=5$. Note also that at H^* this network (which has $\sum w_{ij} = 1252$) is fully restorable against any single-span cut with only 625 spare channels, or a redundancy of only 49.9%. This illustrates in passing the impressive efficiencies in terms of redundant capacity requirements needed for restorability in a reasonably well-connected mesh network.

Figure 5-10. Concept of a threshold hop limit as illustrated in results from [HeBy94].



Not all networks will have a graph topology that provides for such a low threshold hop limit, however. In more irregularly connected networks most spans may be restorable within five or six hops but a few spans in the same network might require up to, say, 20 logical hops. One context where this can happen is in a long-haul network that tends to have densely connected regional subnetworks but a sparse structure linking these regions. Net 5 in Figure 6-10 (ahead in Section 6.8) is an example of a network (encountered in real planning studies by the author's group) that is not "well-behaved" in this sense—one reasonably low hop limit will not serve for the whole design. It is simply impractical to set $H=20$ and attempt to represent the set of all distinct routes up to this limit in the formulation. We will therefore sometimes need additional tactics, other than a simple hop limit, for generating a practically-sized eligible route set. We return to this later.

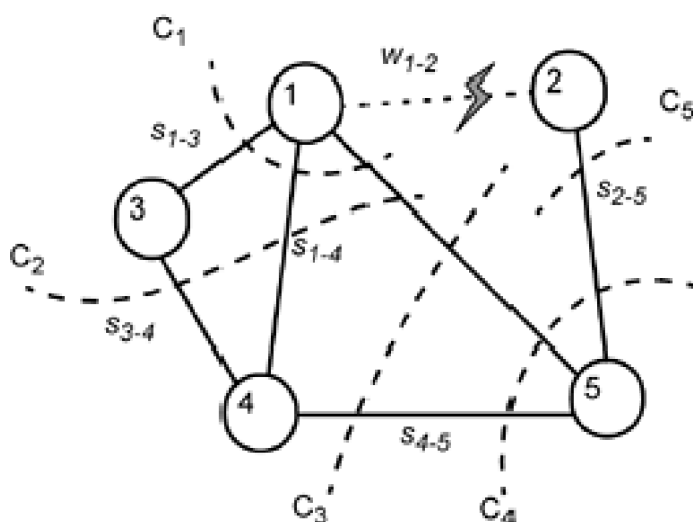
5.3.7 Cut-Oriented Formulations of the SCA Problem

Sakauchi et al. [SaNi90] approach the SCA problem in terms of the spare capacity needed on spans of the minimum cuts that regulate the restoration flow between custodial nodes based on a direct application of the *min-cut max-flow theorem*. The logic is that, regardless of

routing details, the s_j values must support the required restoration flow on every min-cut of the capacitated graph between restoration end-node pairs in each failure scenario. The basic concept is illustrated in [Figure 5-11](#) for a single failure scenario. Span (1-2) is failed with the loss of w_{1-2} units of working capacity. Restoration feasibility then depends on the capacity of the minimum cuts^[2] of the surviving network between nodes 1 and 2. Specifically, it is the *partial cutsets* of the graph with respect to $\{s, t\}$ that are relevant to $\{s, t\}$ restoration. These are defined as the cutsets that put s and t in disconnected subgraphs when the direct span $\{s, t\}$ is removed from consideration.

[2] Recall that a partitioning of the graph into two disjoint sets of nodes $\{\{Vs\}, \{Vt\}\}$ is a *cut* of the graph. The set of edges $C(\{Vs\}) = \{(i, j) | i \in \{Vs\}, j \in \{Vt\}\}$ is the *cutset*. The *capacity* of the cut is the sum of individual capacities on edges in the cutset.

Figure 5-11. The concept of cutset dimensioning to generate the spare capacity (s_j) values.



Cut	Constraint
C_1 :	$s_{1-3} + s_{1-4} + s_{1-5} \geq w_{1-2}$
C_2 :	$s_{3-4} + s_{1-4} + s_{1-5} \geq w_{1-2}$
C_3 :	$s_{4-5} + s_{1-5} \geq w_{1-2}$
C_4 :	$s_{4-5} + s_{1-5} + s_{2-5} \geq w_{1-2}$
C_5 :	$s_{2-5} \geq w_{1-2}$

In the example, five such partial cutsets are identified that would be relevant to the restoration of span (1,2). Each cut generates a statement about the sum of the spare capacity values in the cutset that must be true for restoration to be feasible. Specifically, the sum of the spare capacity on every partial cut must equal or exceed the lost working capacity to be restored. This leads to construction of the inequality statements in the table in [Figure 5-11](#). Depending on how the s_j values are assigned, one or more different cuts will actually be minimal. These will be the active constraints in the solution. The min-cut max-flow theorem assures us that the target restoration level will be feasible if the minimum capacity cut meets or exceeds the respective w_j requirement. Thus we can see below how an LP or ILP could be constructed to assign s_j values so as to minimize total spare capacity subject to the min-cut constraints that assure restorability.

SCA (cut-oriented)

Minimize

Equation 5.20

$$\sum_{i \in S} c_i \cdot s_i$$

subject to:

Equation 5.21

$$\sum_{m \in S} \delta_{i,c}^m \cdot s_j \geq w_i \quad \forall c \in C_i \quad \forall i \in S$$

Equation 5.22

$$\delta_{i,c}^m \in \{0, 1\} \quad \forall (i, m) \in S^2, i \neq m \quad \forall c \in C_i$$

Equation 5.23

$$s_j \geq 0 \text{ integer} \quad \forall i \in S$$

C_i is the set of partial cutsets relevant to the restoration of span i and $\delta_{i,c}^m$ is a parameter equal to one if span m is part of the c^{th} cutset for failure of span i . Other symbols are as previously defined. The technical challenge with this model will be in populating a suitably complete constraint system without actually enumerating all possible partial cutsets for each failure. In SCA (arc-path) the preparatory effort goes into finding the set of all distinct routes P_i for each span failure i . In SCA (cut-oriented) the corresponding preparatory step is (in principle) to enumerate all partial cutsets for each span failure. Notably both of these problems are in $O(2^N)$ to obtain the complete sets in each case. Thus neither method avoids the fundamental information requirements for formulating the problem. But each offers different practical solution approaches. In SCA (arc-path) the basic tactic is to limit the number of eligible routes with a hop limit or other eligible-route criteria (to follow). In SCA (cut-oriented) the corresponding tactic is to start with initially too few cutsets, which will not result in the required level of restorability, and develop the set of cutsets iteratively until a fully constrained problem statement is developed. The procedure is an example of what is called "constraint generation" in O.R. terminology.

Iterative development of the cutset constraints

Let us now refer back to [Figure 5-11](#) and consider a single-failure instance of the SCA (cut-oriented) formulation where only cutsets one and three are initially available to provide constraints. For a numerical example, assume $w_{1-2} = 10$. Then we would have:

$$\begin{aligned} & \min (s_{1-3} + s_{1-4} + s_{1-5} + s_{3-4} + s_{4-5} + s_{2-5}) \\ & \text{s.t. } s_{1-3} + s_{1-4} + s_{1-5} \geq 10 \\ & \quad s_{4-5} + s_{1-5} \geq 10. \end{aligned}$$

By itself this has several equivalent optimal solutions, one of which is $s_j = \{3.33, 3.33, 3.33, 0, 6.66, 0\}$ (variables implied in the same order as in the objective function) with $\sum s_j = 16.66$. Not only is this a non-integral solution, but it is quickly seen that if we test span (1-2) for restorability, the feasible restoration flow is zero. As long as span (2-5) has zero spare, no restoration flow is possible in [Figure 5-11](#). Inspection might suggest that we therefore add cut five to the problem. The tableau then becomes:

$$\min (s_{1-3} + s_{1-4} + s_{1-5} + s_{3-4} + s_{4-5} + s_{2-5})$$

$$\begin{aligned} \text{s.t. } s_{1-3} + s_{1-4} + s_{1-5} &\geq 10 \\ s_{4-5} + s_{1-5} &\geq 10 \\ s_{2-5} &\geq 10 \end{aligned}$$

for which the solution is $s_j = \{3.33, 3.33, 3.33, 0, 6.66, 10\}$ with $\sum s_j = 26.66$ and restorability = $6.66 / 10$. (there are flows of 3.33 on each of routes 1-4-5-2 and 1-5-2).

Clearly, however, the problem is still not fully constrained. An automated method might next find cutset four, but it will have no effect as the corresponding constraint is redundant and already satisfied by $s_{2-5} = 10$ which is asserted by cut five. Another *binding* constraint is needed, that is, one that will be active in the solution. Let us say, therefore, that cut two is added next. The tableau then becomes:

$$\text{min } (s_{1-3} + s_{1-4} + s_{1-5} + s_{3-4} + s_{4-5} + s_{2-5})$$

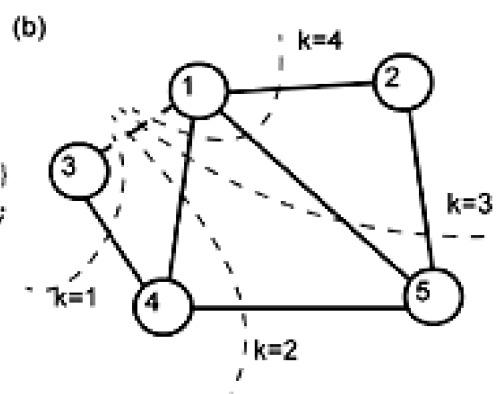
$$\begin{aligned} \text{s.t. } s_{1-3} + s_{1-4} + s_{1-5} &\geq 10 \\ s_{4-5} + s_{1-5} &\geq 10 \\ s_{2-5} &\geq 10 \\ s_{3-4} + s_{1-4} &\geq 10 \end{aligned}$$

with solution $s_j = \{3.33, 3.33, 3.33, 3.33, 6.66, 10\}$, $\sum s_j = 30$ and restorability = $10 / 10$ (100%). Now a fully constrained problem tableau has been reached, as evidenced by the feasibility of the target restorability level. This is the stopping point of the example showing iterative development of the constraint system by finding additional new binding cutset constraints on each iteration. Note, however, that in this example we only generated the s_j values that capacitate the reserve network for one particular failure. In the full problem the $\text{min } \sum s_j$ that simultaneously satisfies all possible failure scenarios is actually obtained.

To mechanize this approach we need a method for determining an initial set of cutset constraints on the first iteration, a method for adding new binding constraints, and a method to ensure an integral solution. In [SaNi90] a specific set of $(N-1)$ partial cutsets are provided initially for each failure scenario. These are the family of "simple cuts" generated (in the absence of span i itself) by the process of adding one more "reachable" node at a time to a set of such nodes (excluding the target), growing from an initial set containing only the source. This is illustrated in Figure 5-12.

Figure 5-12. (a) A process for generating $N-1$ initial simple cuts for each span, (b) example.

```
(a) simple cuts (s, t) :
{Vs} := {s}; {Vt} := (V) - {s};
remove edge (s, t);
for k := 1..N-1 {
  find an edge u, w: u ∈ {Vs} ∩ w ∈ {Vt} - {t}
  s.t. {Vt} remains a connected subgraph;
  {Vs} := {Vs} + {w};
  {Vt} := {Vt} - {w};
  cut(k) := ({Vs}, {Vt})
}
```



Additional cutsets are identified after the first LP iteration by running a max-flow algorithm for each span failure on the s_j values. If, after temporarily deleting span i , the feasible flow between end nodes of span i meets or exceeds w_i , then nothing else need be done for that failure scenario. If this is true for all spans, then the LP phase is complete. When the maximum feasible restoration flow is less than required, however, the saturated spans (where flow equals the current s_j value) are used to provide an additional cutset constraint. Since all spans are tested for restorability, several new constraints may thus be added at each iteration. Some care must be taken, as we will explain shortly, however, in interpreting the saturated edges directly as new constraints. This is one of the improvements made by

Venables below.

To produce an integral design result, Sakauchi et al. round up any real-valued s_j at the end of the LP iteration phase and then test each span for the feasibility of sustaining one spare link removal on each span. This is essentially the same tightening process as in [Section 5.3.4](#). The removal is allowed if all spans remain fully restorable (as tested by a max-flow routine) in the absence of the spare channel removal being tested. Modularity is addressed by modular capacity rounding in conjunction the tightening step. The total working plus (possibly real-valued) spare capacities on each span are rounded up to the nearest modular value, then the tightening phase seeks removal of single modules from spans where possible without impairing restorability.

5.3.8 Venables' Iterated Cutsets Method for SCA

In [Vena92](#) [VeGr93](#), Venables made some enhancements to the basic method above. Briefly these are:

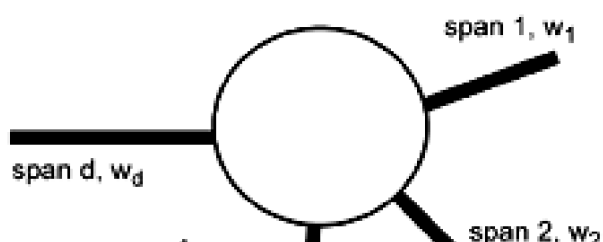
1. Initial cutset constraints are based only on the incident edges at every node.
2. Improved detection of additional binding cutset constraints.
3. Identification of flow-containing regions that simplifies cutset generation.
4. Including integrality and *ksp*-like restoration considerations within each iteration.
5. Improved method for effecting modularity.

Taken together these enhancements lead to a highly effective cut-oriented method for practical solution of SCA problems. We will discuss each in turn in this section.

Incident Cutsets

It is a matter of empirical observation that in span restoration the number of feasible restoration paths for a given failure span is most often limited by the total spare capacity at one or the other nodes adjacent to the failure. Typically in experiments with either *ksp* or max-flow restoration routing models, this is observed in 90% or more of cases in test networks with \bar{d} in the 3 to 4 range and that are optimally designed to have only the minimum of total spare capacity. The point is easily appreciated because spare capacity at the end nodes is directly reduced by the failure itself and provides an obvious potential bottleneck for restoration flow to escape from the vicinity of the failure. (This concept arises again and is further explained in the bounding discussion ahead in [Section 5.4.2](#).) Venables built upon this observation to suggest that the $N\bar{d}$ incident cutsets, illustrated in [Figure 5-13](#), might provide a more effective and compact set of initial constraints than the $N(N-1)$ simple cutset constraints above. Certainly for large network problems $N\bar{d}$ will be much less than $N(N-1)$. There are $N\bar{d}$ such cutsets because on each of N nodes there are \bar{d} incident spans to consider as failure spans. One might expect that starting with the fewer initial constraints will require more LP iterations but since an LP typically runs in a time proportional to the number of constraints, each of the initial iterations will be $O(N-1)$ times faster for large networks (\bar{d} being constant). In addition, such incident cutsets tend to be more often constraining so the increase in iteration is not high.

Figure 5-13. The incident cutset constraints for a node of degree d .



d spans generate d incident cutset constraints for each node:

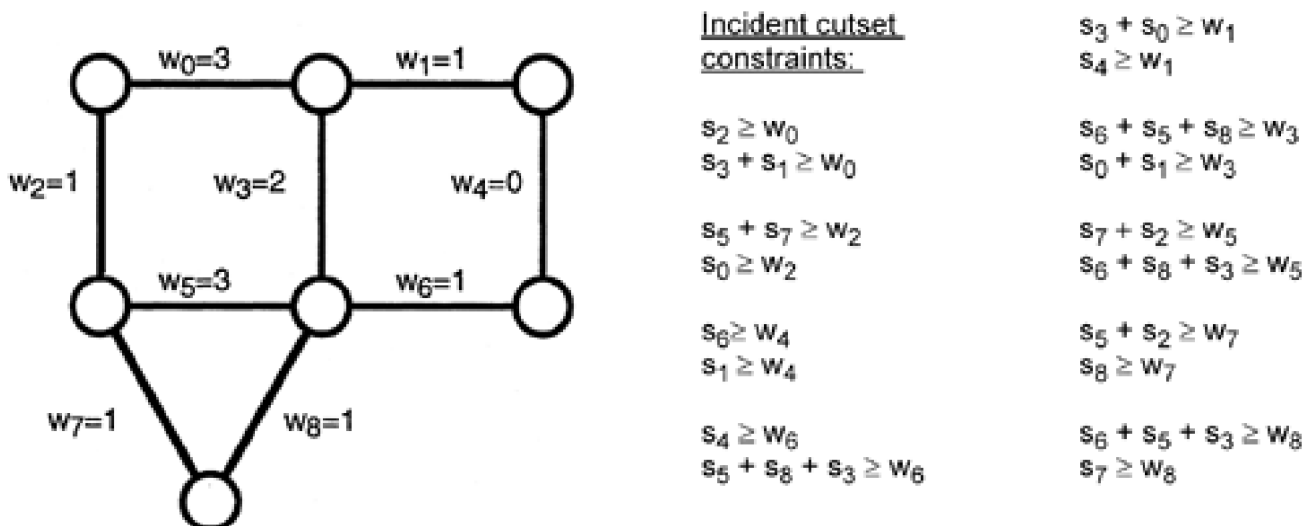
$$\begin{aligned}
 (1) \quad & 0 + s_2 + s_3 + \dots + s_d \geq w_1 \\
 (2) \quad & s_1 + 0 + s_3 + \dots + s_d \geq w_2 \\
 (3) \quad & s_1 + s_2 + 0 + \dots + s_d \geq w_3
 \end{aligned}$$

span 3, w_3 | \dots

$$(d) \quad s_1 + s_2 + \dots + s_{d-1} + 0 \geq w_d$$

A complete set of incident cut constraints for a small example is shown in [Figure 5-14](#).

Figure 5-14. The *incident cutset* constraints for a small network example.



The efficacy of this was borne out by experiments in [Vena92](#). Whereas starting with the full $N(N-1)$ set of simple cut constraints on a variety of test networks would typically require 2 or 3 LP iterations to complete, starting with only the $N\bar{d}$ incident cut constraints typically required 5 or 6 iterations—but of a much smaller LP at each iteration. It was also observed that the number of constraint generating iterations, and the expansion factor of the initial to final sizes of the constraint sets, drop progressively as the network average nodal degree increases. This is attributed to the increasing tendency for restoration to be end-node limited as the connectivity of the graph increases. In test cases on random networks of various sizes but with \bar{d} varying systematically from 3 to 6, the mean number of LP iterations fell from 6 to 3.5 and the overall increase in the number of constraints (through constraint generating iterations) fell from 40% to 25%. That is to say, for example, that a 100 node network of $\bar{d} = 3$, would have 300 initial (incident) constraints and would complete after 5 or 6 iterations, having added typically another 120 constraints in the process. This is to be compared with starting with 9,900 initial simple cutset constraints and doing 2 or 3 iterations of the much larger LP.

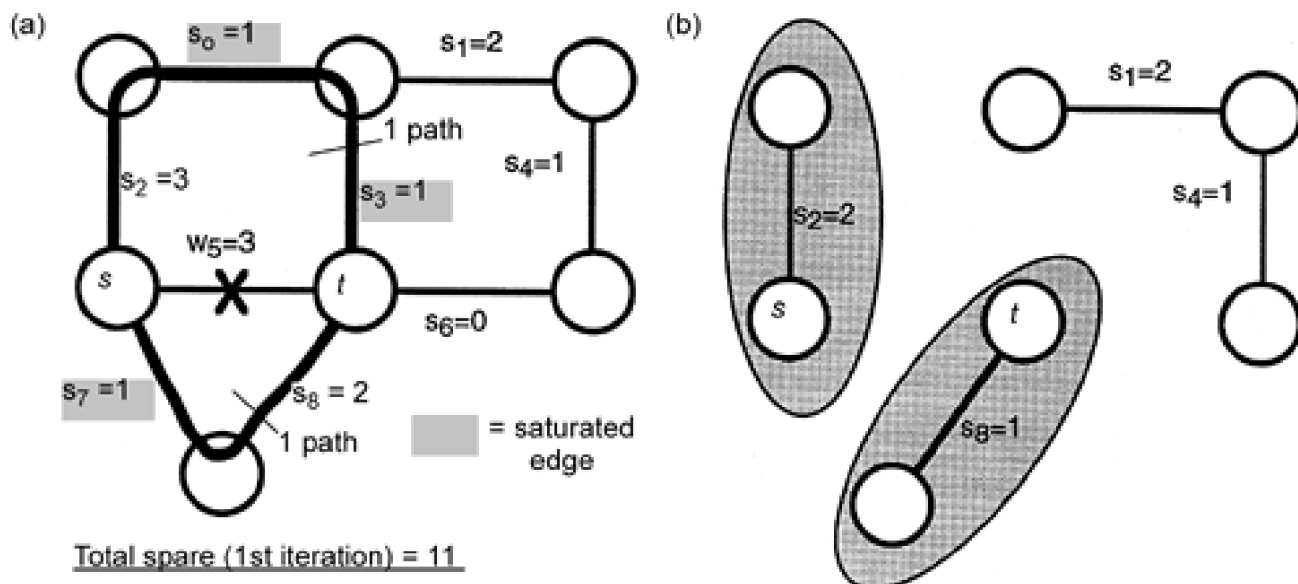
Generating Additional Binding Constraints

Venables also showed that taking all of the saturated spans following a max-flow test that revealed insufficient flow for restoration of a given span is not always effective in producing additional binding constraints. We extend the example just introduced to illustrate this and then explain an alternate procedure for identifying new binding cuts.

After the first LP iteration on the initial tableau of [Figure 5-14](#), the s_j variables are as shown in [Figure 5-15\(a\)](#), on which a maximal feasible restoration flow for span 5 is superimposed. Span 5 is not fully restorable (only 2 out of 3 required paths are feasible) so another constraint has to be generated. If, based on the information in [Figure 5-15\(a\)](#), we simply itemize all saturated spans as constituting another

constraint, we would add $s_7 + s_3 + s_0 \geq w_5$ to the LP tableau. We can see by inspection, however, that this cannot be a *binding* constraint because such a constraint is already satisfied in the LP solution after the first iteration, i.e., $s_7 = s_3 = s_0 = 1$ after the first iteration.

Figure 5-15. (a) The test for restorability of span 5 following the first LP, (b) the disconnected components resulting from removal of flow-saturated spans.

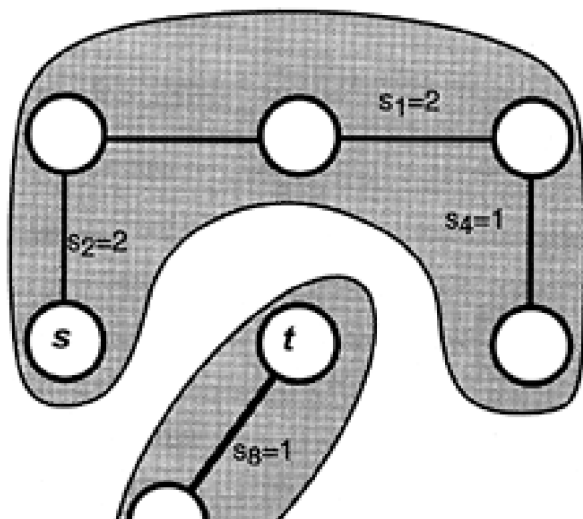


Why is it that the spans saturated by the max-flow do *not* constitute a useful extra constraint in this case? The reason is seen in [Figure 5-15\(b\)](#). These particular saturated edges not only disconnect the graph as expected, but in fact they divide it into *three* disjoint subgraphs. An additional useful or binding cut is found in these circumstances only by unifying two or more of the disconnected subgraphs that do not contain node *t* with the subgraph that contains *s* (or vice-versa). A simple DFS search can identify the nodes in each disconnected component. After removal of all flow-saturated edges, a DFS beginning on *s* will identify the connected subgraph component in which it is contained. Call that subgraph $\{V_s\}$. Another starting on *t* will enumerate all nodes in $\{V_t\}$. If $|\{V_t\} \cup \{V_s\}| = N$ then all nodes are accounted for and a desired additional cut constraint has been discovered. However, if $|\{V_t\} \cup \{V_s\}| < N$ then we have a sign that the graph has been multiply disconnected by the saturated edges.

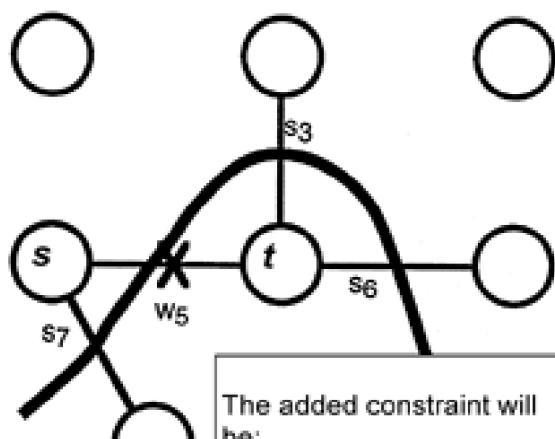
In this case Venables' algorithm seeks to extend one of the subgraphs containing *s* or *t* to include all of the remaining nodes by re-admission of one saturated edge per union; only when that fails is the other subgraph extended. The bias to unify as many non-*s*, non-*t* subgraphs as possible with the first-chosen *s* or *t*-containing subgraph restricts the non-extended subgraph to a smaller geographical area in an attempt to minimize the number of spans of the new cutset constraint. [Figure 5-16\(a\)](#) illustrates the unification process for the example, starting (arbitrarily) with the *s*-containing subgraph by readmitting saturated edge 0 to unify the subgraphs. When only two disjoint subgraphs remain, one containing *s*, the other *t*, then a desired additional cut is defined. [Figure 5-16\(a\)](#) shows the unification step and (b) details how this reveals the additional binding constraint to address the incomplete restorability of span 5. In the example of [Figure 5-15](#) spans 3, 0 and 1 are also found similarly lacking in restorability after the first iteration and the same process leads to added binding constraints in the next iteration as well.

Figure 5-16. (a) Unification of a third disconnected subgraph with the *s*-containing subgraph to define an additional binding cutset constraint.

(a) Unification of two disconnected subgraphs:



(b) The extra binding constraint this identifies:



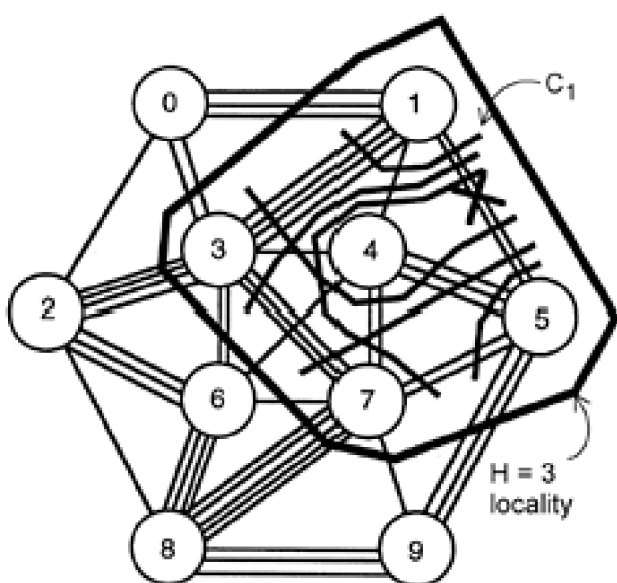


$$s_7 + s_3 + s_6 \geq w_5$$

Restoration Path-Length Limits

A practical concern about the cutset approach is that it capacitates the reserve network so that the required restoration flow is strictly assured only by a max-flow routing process. There is no direct consideration of the individual restoration routes or path lengths that may be implied in a path-set constructed to actually realize the max-flow. This is a potential problem for practical applications (i.e., if a restoration system may have to produce paths of arbitrary length to achieve the expected restorability). One way to reduce the maximum path length requirement that may be implied by a strictly min-cut capacity based design is to use *locality information* to truncate the cutsets being generated as the LP iterates. A locality restriction reduces the number of edges in a cutset, thereby forcing the required flow to be feasible at least with a bounded worst-case hop length. [Figure 5-17](#) illustrates.

Figure 5-17. Defining an H -locality for containment of the restoration flow for any failure span.



Example:

With an $H=3$ locality, all of the following partial cutsets of the complete graph:

- $s_{1-3} + s_{1-4} + s_{0-1} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{0-2} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{2-6} + s_{2-8} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{2-6} + s_{6-8} + s_{8-7} + s_{8-9} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{2-6} + s_{6-8} + s_{8-7} + s_{7-9} + s_{5-9} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{3-6} + s_{6-4} + s_{6-7} + s_{6-8} + s_{2-8} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{3-6} + s_{6-4} + s_{6-7} + s_{8-7} + s_{8-9} \geq w_{1-5}$
- $s_{1-3} + s_{1-4} + s_{0-1} + s_{0-3} + s_{2-3} + s_{3-6} + s_{6-4} + s_{6-7} + s_{8-7} + s_{7-9} + s_{5-9} \geq w_{1-5}$

are reduced to:

$$s_{1-3} + s_{1-4} \geq w_{1-5}$$

The example shows how the " $H=3$ locality" of span (1-5) consists of spans {(7-5) (4-5) (3-7) (1-3) (1-4) (3-4) (4-7)} only. Span (6-7) for instance is not in the $H=3$ locality of span (1-5), but would be within its $H=4$ locality. Thus for a given notional hop limit every span (as a failure span) can have another subset of spans defined which are within its locality for restoration. To then maximize the tendency for restoration path lengths to stay at or below the desired H , we truncate any partial cutset row constraints to admit only those spans that are in the locality, thereby assuring that, at least from a pure max-flow feasibility standpoint, restoration will be feasible *within the H -locality* of each failed span.

Note the careful distinction between containing the flow to within the H -locality and the more direct constraint that restoration paths be individually at or under H hops in length. In practice these will often be the same but while the *total* flow may be contained within the H -locality, individual restoration paths could, strictly speaking, still have somewhat serpentine routes, of length longer than H , while nonetheless being contained in the H -locality. Such is the nature of the routing sometimes required to realize the max-flow. In practice, however, one would rarely expect individual paths longer than H plus 1 or 2 when the complete flow is contained within the H -locality, so H -locality containment is a fairly close surrogate for a hop limit of H .

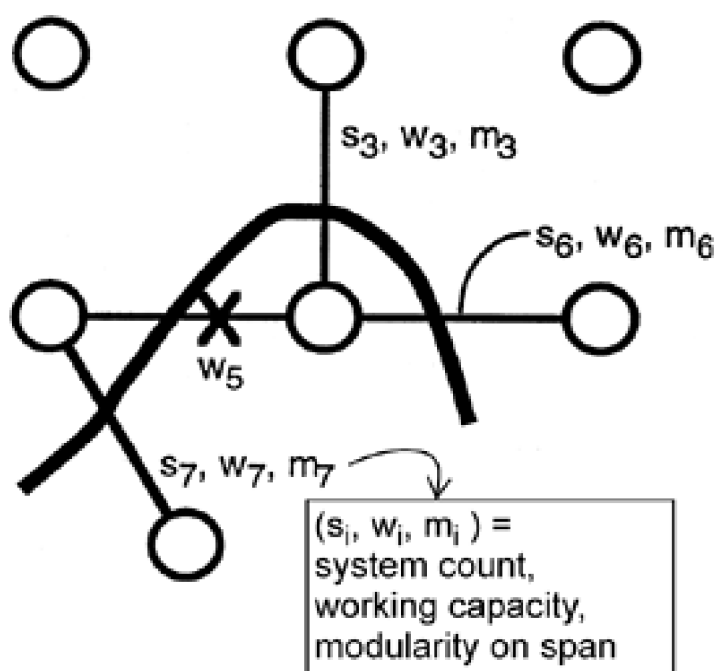
Integrity and *ksp* Restoration Compatibility

If one has a relatively fast means to test all spans for their restorability (such as the $O(n \log n)$ *ksp* algorithm [MaGr94]) then an alternative to rounding and tightening of the final LP result is to adjust any fractional capacities at each iteration. At each LP iteration, any fractional capacities are rounded up to the next higher integer and then all spans are tested individually to see if a unit-capacity removal can be sustained while retaining the restorability level. When this is done at each iteration, the interim adjustments toward integrality are incorporated into the solution at each next iteration, reducing the final gap relative to a true ILP solution. In addition, if the test for restorability at each iteration is actually done with a *ksp* routing process (rather than pure max-flow), the design will similarly tend to develop a placement of capacity that supports full restorability under *ksp* routing not strictly only under an ideal max-flow routing model.

Incorporating Modularity

In Sakauchi's algorithm, modularity is taken into account in the final post-processing of the LP result. It is possible, however, to adapt the initial and generated constraints at every stage to also reflect an inherently modular capacity design environment, including several different modular system capacities. To illustrate, refer back to [Figure 5-16\(b\)](#) where an additional cutset constraint is produced in a non-modular context. [Figure 5-18](#) shows how the corresponding cutset spare capacity constraint is written with modular system capacities. While w_j quantities do not change their meaning, and remain countable in single units as is the nature of actual demand flows, there is no longer an explicit set of spare capacity variables. Rather, the variables to be solved for become the number of modular transmission systems to place on each span. The sparing for restoration is now simply the *non- w_j* part of the modular capacity quantities. As an example, $w_7 \text{ modulo } m_7$ assumes span 7 is assigned a total modular capacity of $o m_7$. $w_7 \text{ modulo } m_7$ thus gives the unused remainder of the total modular capacity, which is taken as the spare capacity.

Figure 5-18. Modifying constraints for modular capacity.



Example:

Previous non-modular constraint:

$$s_7 + s_3 + s_6 \geq w_5$$

Modular constraint:

$$w_7 \text{ mod } m_7 + w_3 \text{ mod } m_3 + w_6 \text{ mod } m_6 \geq w_5$$

5.3.9 Summary: A Complete Cut-Oriented SCA Methodology

We can now offer the following summary recommendation of a complete cut-oriented SCA approach built upon the original work by Sakauchi et al. and as enhanced by Venables:

1. Produce the incident cutset constraints for every node.
2. Solve the SCA(cut-oriented) LP with incident cutset constraints plus any added constraints from below.
3. Round up all non-integral capacity values.
4. Obtain the current restorability level of each span, $R_S(i)$, by *ksp* or *max-flow* algorithms (depending on the restoration model).
5. Test each span j for a unit capacity (or module) removal without any loss of restorability $R_S(i)$ on other spans. Remove any redundant channels or modules found this way.
6. Retest each span for its absolute $R_S(i)$ level in the network following 5. If $R_S(i) = 1$ for every span i , then exit.
7. For each span where $R_S(i) < 1$, generate an additional cutset constraint using the method shown in [Figure 5-16](#) and go to step 2.
8. Optional: For modular system design, form each initial and added constraint in the manner of [Figure 5-18](#).
9. Optional: Truncate all cutset constraints according to locality considerations to limit the range of flows. (If employed this should best be done within step 7.)

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

5.4 Jointly Optimized Working and Spare Capacity Assignment

In the treatments of SCA so far the problem has been to optimize spare capacity to make restorable a set of already determined working capacity levels w_j on each span. Typically the w_j values would arise from shortest path routing of the demand matrix, but they could also be the result of other demand routing processes. The important point is that in SCA the routing of working demands is determined independently and without regard to the subsequent problem of survivability. It makes sense in principle that there may be ways of routing the working demands that somehow makes the related survivability design less costly. Of course total cost responds to both working and spare capacity, so any deviation of the working path routes from their shortest paths should be associated with an even greater corresponding reduction in total spare capacity, or there is no net gain. This section introduces the design model for such *joint* optimization of the working path routes and spare capacity placement, called JCA for Joint Capacity Allocation.

5.4.1 Joint Capacity Allocation (JCA) Model

The aim in JCA is to minimize the *total* cost of working and spare capacity in a span-restorable mesh network. Parameters and sets already defined are unchanged, but we have new sets, parameters, and decision variables involved, mostly to represent the aspects of the now-variable routing decisions for each working demand. As the other parameters and variables were introduced somewhat incrementally, let us now restate the full set of standard parameters, sets, and variables that we will use henceforth. These are:

Sets:

- S is the set of spans in the network. S is usually indexed by i to designate a span in the context of a failure span or indexed by j in the context of a surviving span or to enumerate all spans in general. Unless stated otherwise, S is also the set of failure scenarios being considered in a design.
- D is the set of non-zero demand quantities for each peer-to-peer service path relation in the network. Usually D is no larger than the set of all O-D node pairs, but in some cases D can represent multiple demand exchanging entities within each node, such as when distinct processes or entities within the same O-D pair are identified as the end-nodes of the demand requirement. In all cases each pair of entities exchanging a unit of demand is also called a "relation." The members of D are demand values d^r (integer) with index r .
- Q^r is the set of routes eligible for carrying demands for the r^{th} demand pair, index q .
- P_j^i is the set of routes eligible for carrying restoration flow for the i^{th} span failure, index p .

Parameters:

- c_j is the cost of a unit of transport capacity (a channel or a module depending on the context) on span j . If cost is length-dependent the effect of length of a span is included in c_j unless length-dependent costs are given explicitly in which case c_j may be given a per-unit-length cost interpretation. The unit of capacity is problem specific, e.g., it may be a wavelength, a fiber, an OC-n, etc.
- $\xi_j^{r,q} = 1$ if the q^{th} eligible working route for relation r crosses span j , zero otherwise.
- $\delta_{i,j}^p = 1$ if the p^{th} eligible restoration route for failure span i crosses span j , zero otherwise.

Variables:

- w_j is the number of working capacity units allocated to span j (integer).
- s_j is the number of spare capacity units allocated to span j (integer).
- $g^{r,q}$ is the amount of working flow allocated to the q^{th} eligible working route for relation r (can be integer or real depending on context or solution methods).
- f_i^p is the amount of restoration flow allocated to the p^{th} eligible route for restoration of span i (integer or real depending on context or solution methods).

The formulation for joint optimization is then: ^[3]

^[3] Henceforth we will not always explicitly indicate "for every" when enumerating set elements in summations. For example $q \in Q$ in [Equation 5.25](#) implies $\forall q \in Q$.

JCA

Minimize:

Equation 5.24

$$\sum_{j \in S} c_j \cdot (s_j + w_j)$$

subject to:

- The working flow assigned to eligible working routes for relation r fully serves all the demand on relation r :

Equation 5.25

$$\sum_{q \in Q} g^{r,q} = d^r \quad \forall r \in D$$

- Span i 's working capacity is sufficient to meet the prefailure demands of all relations having working flow across it:

Equation 5.26

$$w_i = \sum_{r \in D} \sum_{q \in Q} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall i \in S$$

- The total restoration flow for span i meets the required restoration level (100% here):

Equation 5.27

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

- The spare capacity on span j supports all simultaneous restoration flows imposed on it.

Equation 5.28

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p \quad \forall i, j \in S^2 | i \neq j$$

Equation 5.29

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i \quad g^{r,q} \geq 0 \quad \forall r \in D \quad \forall q \in Q^r$$

Equation 5.30

$$s_j \geq 0 \quad w_j \geq 0 \quad \text{integer} \quad \forall j \in S$$

The main differences between JCA and SCA are the addition of [Equation 5.25](#) and [Equation 5.26](#), in which w_j quantities appear as variables, and the addition of a new set of eligible routes having to do with the routing decisions for working demands. [Equation 5.25](#) asserts that for every demand pair, the total working flow assignments to eligible routes must match the total demand. [Equation 5.26](#) generates the w_j working capacity variables by intercepting all of the assignments of working flow to eligible routes that cross span j . Note this implies that the total demand exchanged by a relation may be split up over possibly several different routes. This makes flow leveling benefits (to be discussed) accessible to the solution. On the other hand, we will have to be rather pragmatic in how many eligible working route choices we try to represent because, while there are only $|S|$ sets of eligible restoration routes, there could be up to $N(N-1)/2$ O-D pairs exchanging non-zero demands, each requiring a set of eligible working routes. In total a JCA problem therefore involves an expansion from $|S| + \sum |P_i|$ variables in SCA to $2|S|$ capacity variables and $\sum |P_i| + \sum |Q^r|$ variables for restoration flow assignment plus working flow assignment. The number of constraints does not expand quite as badly, however, because the working capacity decisions depend only on the one "no-failure" scenario whereas the spare capacity decisions on each span are influenced by a separate array of constraints for each possible other span failure, giving the $|S|(|S|-1)$ spare capacity constraints. Thus, to the $|S|^2$ constraints in total for SCA we add $|D|$ for working flow assignment and $|S|$ for working capacity generation for JCA.

On the face of it (there being $O(N^2)$ O-D pairs), the total number of eligible working routes to represent, $\sum |Q^r|$, could be much larger than the number of eligible restoration routes. In practice, however, we are helped by the strong tendency for working paths to depart very little from shortest paths even under joint optimization. This means that a policy of representing all equal-length shortest routes and perhaps only the next three longer route options performs well for working paths. This is in contrast to the eligible route sets for restoration flows, where we may typically want to represent at least 20 eligible routes, deviating greatly from the shortest path between failure end nodes.

We return to this topic when we later give a general policy for route set generation.

At least three studies [IrMa98], [XiMa99] [DoGr01] have found that joint optimization of working routes along with the placement of spare capacity can (for span restoration)^[4] yield a significant reduction in total capacity relative to an SCA design using shortest path routing of the working demands. For some of the test networks in [IrMa98] the figure was as much as 28% in total capacity. Interestingly, however, it is also observed that in JCA designs the working routes still tend to deviate only a few percent from the true shortest-path routes. This seemingly contradictory situation is explained by understanding that JCA benefits primarily result from load leveling over essentially equal shortest routes, not from any dramatic departure from shortest paths.

^[4] For path restoration (Chapter 6), the effect of joint optimization is much less.

An argument to appreciate why it is unlikely for working paths to go much beyond their shortest route even under joint optimization is to consider what an uphill prospect it is for this to realize a net gain. If a working path takes a route that is $X\%$ longer than its shortest route, then to first order $X\%$ extra working capacity cost is incurred. For this to represent a net gain it has to be offset by more than the same extra cost from the spare capacity. But typically a span-restorable mesh may have 50% redundancy. This means that on average, every increased capacity usage on a working path has to be related to an improvement that yields *twice the same percentage improvement* in the spare capacity.

To illustrate why load leveling effects are more important under JCA than deviated working routes, we need to first make an important digression to introduce some basic insights about how network connectivity and working capacity balance both influence capacity efficiency. Following this we will return to complete our discussion of JCA.

5.4.2 Isolated Nodal Bounding Considerations

To explain the comments about working capacity balance, consider Figure 5-19 which shows an isolated node of degree d . The conditions at a node in isolation do not entirely determine overall properties of a span-restorable capacity design, but they do allow for some "thought experiments" that yield certain useful insights about this class of network in general. Now consider the failure of span 1, having w_1 working capacity. Obviously for this failure to be restorable the node must have enough spare capacity on other spans $2 \dots d$ to support a total restoration flow of w_1 . We can say this without even considering any wider details of the network of which node A is a part. Similarly, in the absence of other more global considerations we can say that each span i requires that the total amount of spare capacity on other spans at the node must be at least the working capacity on the failed span because the initial egress of the failed working paths onto restoration paths is only possible if the sum of the spare capacity on the $(d-1)$ surviving spans equals the working capacity on the failed span. Thus a minimum (i.e., necessary but not sufficient) set of conditions for restorability is:

Equation 5.31

$$\sum_{\forall j \in \{1 \dots d\} | j \neq i} s_j \geq w_i \quad \forall i \in \{1 \dots d\}.$$

Figure 5-19. Isolated nodal view of restoration considerations leading to the $1/(d-1)$ bound.





Purely Topological Lower Bound

Consider the case, however, where these conditions are true and the remaining body of the network is not limiting in terms of restoration flow between this node and the other end of the failed span. Then, if the working capacity on each span was also exactly equal, full restoration is possible with a nodal redundancy of:

Equation 5.32

$$1/(d-1).$$

This can be seen either by inspection of [Figure 5-19](#), or through [Equation 5.31](#) by setting all $w_j = 1$ (or any other constant amount). It follows that in the best case from an efficiency standpoint, every span could have $w_j = w_1$ in which case the ratio of spare to working capacity (the *redundancy*) becomes:

Equation 5.33

$$\frac{\sum_{i \in 1 \dots d} s_j}{\sum_{j \in 1 \dots d} w_j} = \frac{(d \cdot w_1)/(d-1)}{d \cdot w_1} = \frac{1}{(d-1)}.$$

This is a simple lower bound on the redundancy required for span-restorable designs, based on purely topological considerations.^[5] We will frequently rely on it as a basic concept in reasoning about survivable networks. Although it takes a purely isolated nodal view, it is found experimentally that in an efficiently designed span-restorable network, the limiting cutsets are most often incident to one or the other end-nodes of the failure span, giving practical validity to this simple line of reasoning in [Figure 5-19](#) about how nodal degree will affect spare capacity.

^[5] In other sources, readers may see $1/d$ claimed as the corresponding lower bound. There is no discrepancy however as $1/(d-1)$ applies if redundancy is defined as $\sum_{s \in S} \sum_{w \in W} w$ and $1/d$ is equivalent if redundancy is defined as $\sum_{s \in S} (\sum_{w \in W} w + \sum_{s \in S} s)$.

It should be added that because of the way it is derived, it is only true to say that this is a strict lower bound for redundancy of a single node, or of a network where all nodes are of equal degree and capacity. When applied to a general network as a whole, with \bar{d} representing only the *average* degree of all nodes, $1/(\bar{d}-1)$ cannot strictly be claimed as an unbreakable lower bound. This is because for any given \bar{d} , an example may be constructed containing an arbitrarily highly connected subnetwork for which $1/(\bar{d}-1)$ goes to zero. If this subregion also contains more capacity than the offsetting sparse portion of the topology which brings \bar{d} to its said average, then

theoretically the capacity-redundancy can be brought arbitrarily close to zero. This is, however, essentially only a theoretical argument. In practice, where the d of nodes in any real transport network is highly limited, $1/(\bar{d}-1)$ does serve essentially as lower bound for any span-restorable network. And when differential capacity effects on spans are taken into account, as in the next section, it is even harder to approach $1/(\bar{d}-1)$ in practical networks.

5.4.3 Effect of Capacity Balance on Redundancy

We say that the $1/(d-1)$ result is a purely topological bound on nodal redundancy because it reflects only the limiting effect of the number of spans at a node, without any capacity weighting to spans. But let us now consider the role of working capacity values on the spans on the isolated nodal redundancy, recognizing that in general they will not all be equal. Consider again the largest span w_1 and the second largest w_2 in [Figure 5-19](#). If we write the corresponding inequalities on the surviving spare capacity for restorability, and then add them we obtain:

Equation 5.34

$$\begin{aligned} s_2 + s_3 + \dots + s_d &\geq w_1 \\ s_1 + s_3 + \dots + s_d &\geq w_2 \\ \hline s_1 + s_2 + 2 \cdot (s_3 + \dots + s_d) &\geq w_1 + w_2 \end{aligned}$$

This tells us that $w_1 + w_2$ is therefore itself a form of upper bound on the total required spare capacity at the node (because s_1 and s_2 are present directly in the sum and the total of all other spare capacity is accounted for twice). So for an upper bound on redundancy that does not assume that all spans have $w_j = w_1$ as above, we could take the above overestimate of the total spare capacity and combine it with an optimistic view of the total working capacity at the node. The most optimistic case for the latter, where all w_j are not simply equal, would be for all $w_j: i \neq 1, i \neq 2$ to be equal to w_2 . Then, the ratio of spare to working capacity becomes:

Equation 5.35

$$\frac{\sum_{j \in 1 \dots d} s_j}{\sum_{j \in 1 \dots d} w_j} \geq \frac{w_1 + w_2}{w_1 + w_2 + (d-2) \cdot w_2} = \left(\frac{w_1}{w_2} + 1 \right) / \left(\frac{w_1}{w_2} + (d-1) \right).$$

This is an *upper* bound on the redundancy because it is formed from an upper bound (an overestimate) of the required spare, combined with a lower bound (an underestimate) of the working capacity such spare protects. Note the way [Equation 5.35](#) behaves if we postulate increasing disparity in working capacity values at the node: as $w_1/w_2 \rightarrow \infty$ (and by implication $w_1/w_i \rightarrow \infty \forall i \geq 2$) at the node), the expression reaches a limit of 1. That is to say that in the worst case, a ring-like redundancy of 100% is required, regardless of nodal degree! This is an important additional understanding of what affects efficiency in a mesh-restorable network: it says that even with a high nodal degree, if virtually all the working capacity is on one span, then the spare capacity, however it is distributed over the other spans, still sums to w_1 . In the limit with no other working capacity present to share that spare capacity, we get ring-like (i.e., ~ 100%) redundancy in the worst case.

Thus, redundancy in the isolated node thought-experiment can vary from as little as $1/(d-1)$ up to 1, depending on the balance of the w_j

values. A richly connected topology is in principle beneficial but *the potentially low redundancy of $1/(\bar{d}-1)$ can only be approached if working span quantities are also balanced.* Regardless of nodal degree, redundancy could range up to 100% as the disparity of working capacity values increases. Thus, we are ideally seeking a high average nodal degree and balanced working capacities at nodes in an SR network.

Approximate Estimator for a Network as a Whole

Consider again the isolated node with d spans, and but now let the w_j be random values. This will lead to an expression to approximate the overall redundancy of a network of a given network average nodal degree and network average imbalance in w_j . Over the set of d inequalities expressed in [Equation 5.31](#) as a whole, there will be some maximum w_j denoted $w_{j\max}$. Since this w_j exceeds all others, it also follows that the corresponding row-wise sum on the LHS of [Equation 5.31](#) also exceeds all other sums of spare capacity and is the binding constraint. Thus $w_{j\max}$ itself is a surrogate for the total spare capacity required over the $d-1$ other spans at the node. Consequently a separate lower bound on the isolated nodal redundancy is:

Equation 5.36

$$\sum s / \sum w \geq \frac{\max(w_j)}{\sum w_j}$$

where both the sums run over spans incident to the node.

This lower bound does not add in any sparing on the span where $w_{j\max}$ itself occurs and so can be tightened a bit further (below), but is of value in its present form to also demonstrate how the achievable redundancy is degraded by the diversity of w_j values on spans at the same node. The separate bounding considerations on nodal degree and w_j disparity can be brought together and tightened into a two-variable model for span-restorable mesh networks as a whole if we introduce the notion of a peak-to-mean w_j factor, x . x is defined

such that $\max(w_j)$ is $x \cdot \bar{w}_j$ where \bar{w}_j is the average w_j of spans over the network as a whole. Then a restatement of [Equation 5.36](#) becomes:

Equation 5.37

$$\sum s / \sum w \geq \frac{\xi \cdot \bar{w}_j}{(d-1) \cdot \bar{w}_j + \xi \cdot \bar{w}_j} = \frac{\xi}{(d-1) + \xi}$$

To add the unaccounted-for spare capacity mentioned above, on the span of $w_{j\max}$ itself, we note that spare capacity on this span is driven by $d-1$ sharing of the restoration requirement generated by the *second-largest* w_j at the node. For a worst-case model, we can let this value be as large as $w_{j\max}$ itself. [Equation 5.37](#) is then extended to become:

Equation 5.38

$$\sum s / \sum w \geq \frac{\xi \cdot \bar{w}_j + (\xi \cdot \bar{w}_j) / (d-1)}{(d-2) \cdot \bar{w}_j + 2 \cdot \xi \cdot \bar{w}_j} = \frac{\xi \cdot \left(\frac{1}{d-1} + 1 \right)}{d-2 + 2 \cdot \xi} = \frac{\xi \cdot d}{(d-1) \cdot [d-2 + 2 \cdot \xi]}$$

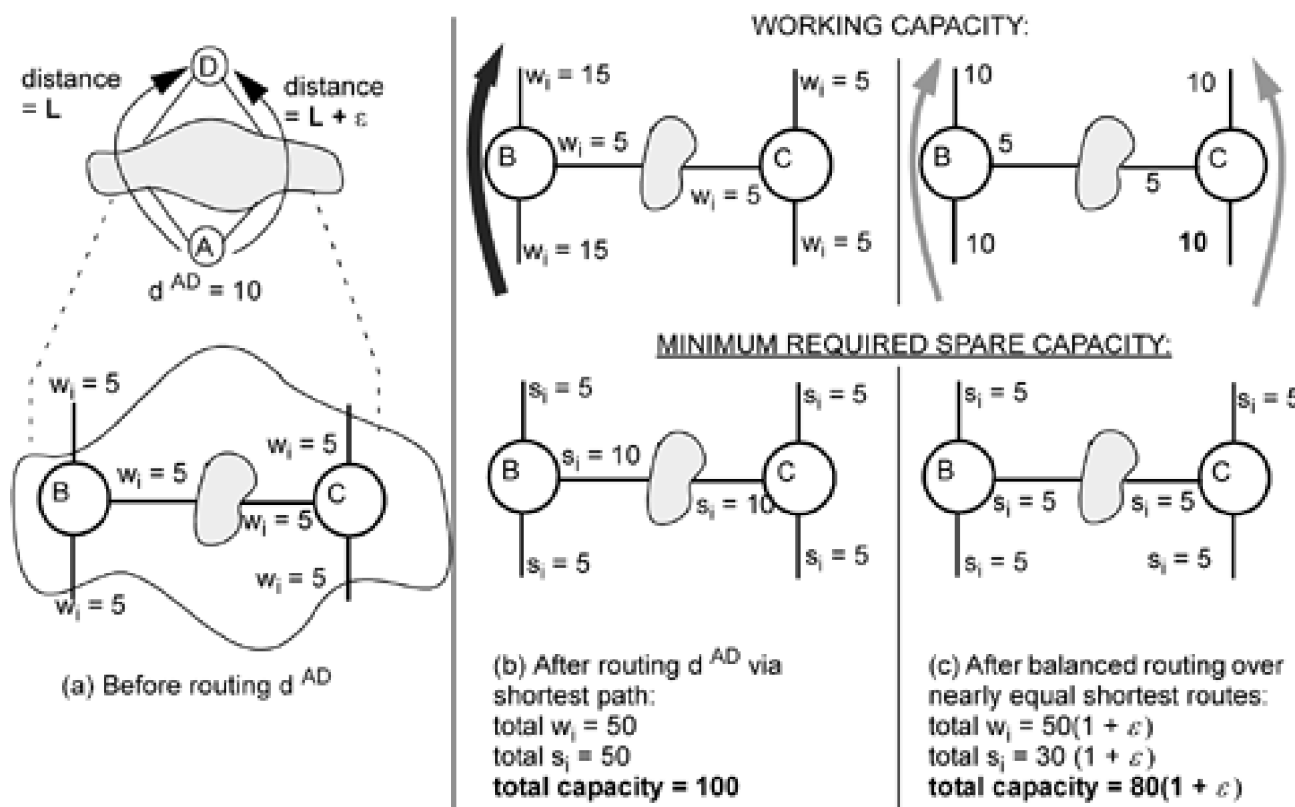
Thus a network with $\bar{d} = 4$ and peak w_i that are 3 times the network average would be estimated to require 50% redundancy, whereas its topological lower limit would be 33% redundancy. These numbers are actually fairly characteristic of real networks. Depending on the model for the distribution of w_i values, and the second largest w_i relative to the largest, a variety of other estimators or empirical models can be derived from [Equation 5.38](#).

In summary we see that network average nodal degree and working capacity leveling are somewhat interchangeable in terms of network efficiency. Nodal degree sets an absolute lower limit on redundancy, but working capacity balance decides how closely the bound can be approached. Even if nodal degree is increased by adding spans, the disparity in w_i quantities can frustrate or limit the intended reduction in redundancy. Thus, in practice, if the w_i values have high variance at nodes, a less expensive measure than adding new spans would be first to try routing changes that will reduce the disparity in w_i quantities. This is largely what JCA does.

5.4.4 Understanding the JCA Benefit

Let us now return to the discussion of JCA designs and understanding how they produce the improvements they show over SCA. The above has shown us that flow leveling is almost as important to efficiency as having a high nodal degree. But more leveled w_i can be achieved even with little or no increase in working path lengths by choosing intelligently among equal-length shortest path options to lower the disparity between largest and smaller w_i quantities at nodes. As argued above, it is an uphill battle for JCA to go far from shortest paths because it needs typically a 2:1 return percentage-wise on spare capacity savings to break even. The more efficient the network is to start with, the harder it is to obtain a net payback on spare capacity by deviating to longer working paths. JCA capacity savings are therefore much more likely to be obtained through intelligent load-balancing over nearly-equal routes. [Figure 5-20](#) gives an example. This is the main explanation for JCA savings in total capacity.

Figure 5-20. How JCA reduces total capacity through intelligent working flow balance.



[Figure 5-20\(a\)](#) shows a generalized situation where nodes A and D have two essentially equal-length shortest routes between them through an arbitrary subnetwork. ϵ is introduced to acknowledge in general that the second A-D route's length will never be exactly the

same as the strictly shortest route. We let it be "slightly longer" therefore, i.e., $(1 + \epsilon)$ times the shortest route for the sake of a more general argument. The subnetwork between A and D is assumed to contain two nodes B and C that are each on one of the routes from A and D. In [Figure 5-20\(b\)](#), we look at the working and spare capacity implications on the nodes B and C under strictly shortest-path routing. The top panel shows the revised working capacity requirements to support the new d^{AD} flow of 10 units. The bottom panel shows the corresponding "isolated node" spare capacity allocations to spans incident on B and C. [Figure 5-20\(c\)](#) shows the corresponding working and spare requirements if the d^{AD} working flow is instead split over the two nearly equal routes. The total working (capacity times distance) requirements are essentially the same. An increase of ϵ occurs because the second shortest route is longer, but there is a 40% reduction in total spare capacity! The key message is not that bundles of working paths between node pairs should necessarily always be split. Rather, the important effect is that this particular bundle-splitting decision leads to more balanced w_j quantities on spans incident to nodes B and C.

5.4.5 Operational Strategy for JCA-Based Incremental Capacity Planning

The insight that JCA efficiencies are derived primarily from flow-balancing considerations, rather than from any radical deviation from shortest-path routing, bears favorably on the practical deployment of JCA-based designs. The *a priori* concern with jointly optimized capacity planning in general is that while we can solve for a complete design to a forecast demand matrix, it is not clear that we would know how to route individual demands as they arrive incrementally on a day-to-day basis, if they are routed in any way *other* than by shortest paths. Conceptually, if a demand forecast is accurate, then shortest-path routing of individual demands as they arrive during a planning period would essentially reproduce the w_j quantities that an SCA design planned for that demand forecast. The apparent problem is, however, that if the precise routing of each demand in a JCA design strictly depended on the global optimization, then how would we know how to route each demand as they arrived individually?

But the issue is addressed in practice by the fact that most JCA-related changes to working routes are essentially of a flow-balancing nature, meaning that JCA design efficiencies could still be approximated in practice by extracting the two or three "preferred" routes for each O-D pair from an optimal JCA design for the forecast period. The idea would be to take the demand forecast for the next planning period, solve for the optimal JCA target design at the end of the forecast period (recognizing existing capacities and flows) and then analyze the resulting design to extract recommended incremental routing policies for each O-D pair. These policies would then be applied to new demands as they arrive incrementally in the next operational epoch.

More specifically, the routing policies would be extracted from the JCA design by observing the routes actually taken by working flow for each O-D pair, and the relative amounts of working flow assigned to each route. This information would then be used to structure the allocation of new demands on the given O-D pairs to the associated routes. If three routes are used in the JCA solution between nodes A-B, bearing say 10, 7 and 3 demands each, then a corresponding route selector can be updated to count in the same proportion as 10, 7, 3 as each new demand is provisioned (or as demands come and go during the next period) so the overall allocation approximates this ratio.

[\[Team LiB \]](#)

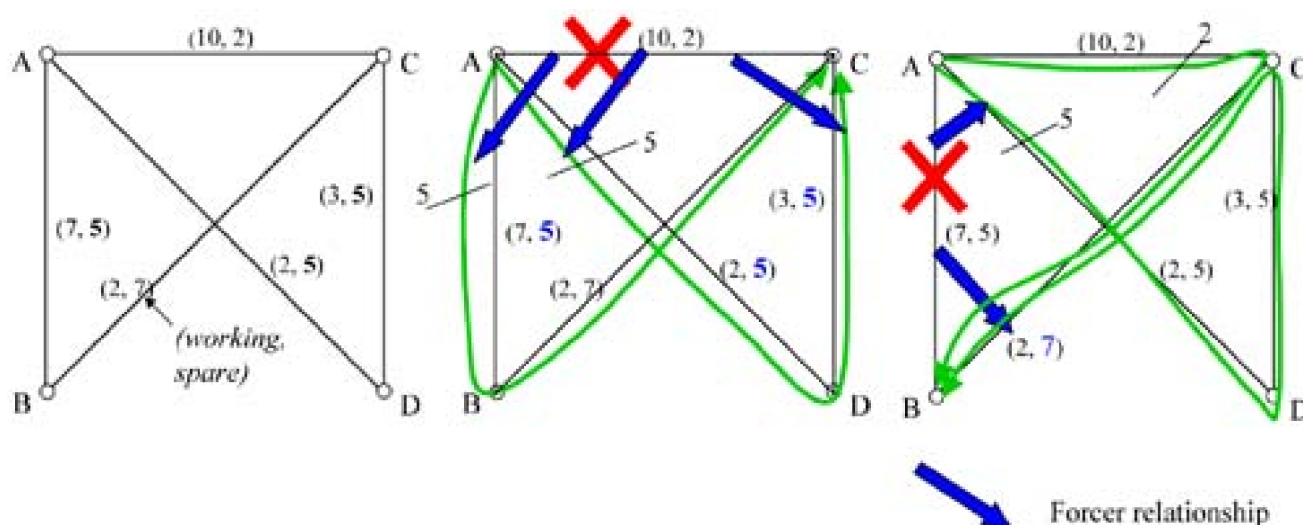
◀ PREVIOUS NEXT ▶

5.5 The Forcer Concept

The forcer concept is a useful technique for analysis of the capacity structure of a span-restorable network. Fundamentally it tells us which working span quantities in a span-restorable mesh are causally related, or in a sense responsible for, other spare capacity requirements in the network design. This knowledge has several subsequent uses in provisioning [GrLi99], in hybrid network design [GrMa00], and in network evolution planning.

To illustrate the basic principle let us start with a small example. Figure 5-21 shows a four-node network with a minimal set of spare capacity allocations such that the working capacities are fully restorable. If span AC is cut, its 10 working channels will be restored via five replacement paths on route ABC and five on route ADC. At least five spares are therefore needed on spans AB, BC, AD and DC. However, span BC has 7 spare channels. This is explained by considering span AB as a failure span. Should span AB fail, its seven working channels can only be restored by paths involving span BC. A feasible restoration path set is five paths on route ADCB and two paths on route ACB. Span BC is common to both of these simultaneously required routes, and therefore must have at least seven spares. Thus, regardless of the fact that no other span needs more than five spare channels on BC, span AB (as a failure span) forces there to be seven spare channels on BC for 100% restorability under all span cuts. Thus, we say that span AB is the *forcer* of span BC in the network in Figure 5-21. In other words, no other span requires more spares on BC than span AB does. Others may have equal requirements.

Figure 5-21. Illustrating forcer relationships in a small example.



Note also that span AC has the largest working capacity in the network, but it is the smaller demand on AB that is the forcer for span BC. Similarly, inspection shows that span AC is the forcer of spans AB, AD, and DC. In other words, if the working capacity of span AC was increased by one, then the sparing on at least one of these spans would also have to be incremented to retain full restorability. Should span AC's working capacity be altered in this way, increasing the sparing on span AB to 6 would be the least costly way to ensure full restorability. Because span AC does not force span BC, the addition of a single spare channel on span AB alone creates the needed path (ABC). A more costly alternative would be to increase the sparing on spans AD and DC to 6 (providing an additional path ADC).

The forcing relationship conversely implies that if a working channel were to be removed from AC, then a spare channel could be *removed* from the spans in any discarded restoration path that is *uniquely forced* by AC. In this case, AB would be such a span in the discarded restoration path ABC. By comparison, removing path ADC would not result in a reduction of sparing, because span AB also forces (or we will say *co-forces*) spans AD and DC.

In general, for any span j , there will always be some other span i which will require a number of spare channels on_j that is greater than (or equal to) that required by any other failure span. When this relationship is true, we say that span i is the forcer (or a co-forcer) of span j . Strictly, because s_j and w_j are whole numbers, more than one span may require the same number of spares on span j so the forcer relationships among spans may be many-to-one, i.e., a span may have more than one co-forcer. For example in Figure 5-21 span AC and

AB are equal forcers of AD and DC. Often in practice, however, s_j values in an SCA plan will be attributable to the forcing effect of just one other particular span. We therefore give the following formal definition of a forcer span in an integral but non-modular mesh-restorable network:

A forcer span is any span for which an increase in network total spare capacity is required to maintain restorability if the span's working capacity is increased.

Conversely, a non-forcer is a span on which at least one working channel may be added without requiring any additional spare capacity for the network to remain 100% restorable.

Thus every span is either a forcer or non-forcer. In the presence of modular capacity units, the concept is the same but the definition is: if an increase in working capacity on span i requires an increase in total network modular design capacity to retain restorability, then span i is a modular-forcer. Where working paths are shortest-path routed prior to spare capacity placement, such an increase in total modular capacity can arise only through increases in spare channel counts forced by the underlying w_j quantities which remain non-modular in nature. Therefore under modularity, the forcing relationships are still present between w_j and s_j quantities, which are integers, but the effects are only manifest in the total modular capacity in a stepped manner. In other words, the external effects are suspended until module boundaries are exceeded. Forcing relationships may be altered in detail by taking into account unused capacity in modules which can be used as spare capacity. Every span, however, remains either a forcer or non-forcer, as classified by the criteria stated on the logical working and spare capacity of the network.

Clearly forcer relationships depend on both the working capacities of the spans and topological effects. Changing the working values can change the relationships. For example, removing 4 working channels from span AC would change its forcing relationship with span AB. It would no longer force that span, and span BC would be the new forcer of span AB (forcing it to have 2 spare channels). In the network's original state, span BC is a non-forcer, because it can more than restore its working channels using the sparing that exists for other span failures. The subset of spans which are forcers is called the *forcer skeleton* because these spans alone are sufficient to generate the entire spare capacity plan. All non-forcer w_j could be set to zero and the same total spare capacity would still be required for survivability of the forcer spans alone. In [Figure 5-21](#), for example, the forcer skeleton consists of spans AC and AB only.

5.5.1 Forcer Analysis Procedure

The discussion of [Figure 5-21](#) suggests that we should be able to mechanize the forcer analysis of larger networks. In the process we should also be able to identify the number of working channels on each forcer that would have to be removed before that span became a non-forcer. The reduced level of working channels at which a forcer would become a non-forcer is called the *forcer magnitude* of that span. Such an analysis tells us a lot about the structural inter-dependencies in a mesh restorable network. It shows which spans have working channel quantities to which the network restorability design is directly sensitive to growth. Conversely for each span, it shows which other span(s) are acting as forcers for the sparing on that span. We now outline a polynomial time procedure through which the forcer structure can be obtained, in addition to measures (to be defined) of the forcer strength of various spans in the network.

One way to find forcer magnitudes is to remove working channels from a selected span while continually resolving the SCA problem.

When the n^{th} removal does not result in a further drop in total spare capacity, the threshold of that span as a forcer has been found. A more practical procedure, however (and not exactly equivalent), is to extract the forcing relationship from a round of *ksp* restoration rerouting "experiments". Each span is visited once and treated as a prospective failure span, x . We then run the *ksp* algorithm as a simulation of the restoration mechanism seeking a path-set for the w_x failed channels on span x . For each other span i that is in the path set found we record $s_i(x)$, which is the number of spare channels used by restoration paths that traverse span i in response to failure x . If $s_i(x) < s_j$ then span x is not a forcer of span i . If, for every other span i not equal to x , $s_i(x) < s_j$ then span x is globally a non-forcer. On the other hand, if there is one or more spans i for which $s_i(x) = s_j$ then span x is a forcer in general and in particular is a forcer of span i . (If $s_i(x) > s_j$ it would mean the network was not 100% restorable but $s_i(x) > s_j$ also cannot happen as *ksp* (or max-flow) is limited to using available capacities in the design). A forcer may force several other spans, and may also be a co-forcer on one or more of the spans. The co-forcer relationship recognizes that the sparing on a particular span may sometimes be equally forced by two or more forcer spans.

Having all $s_i(x)$ values from the *ksp* simulation of all span restorations we can also obtain measures of forcing strength in the following sense. If we define the magnitude of a forcer span as the number of working channels that would have to be removed from the span before further removals would give no corresponding reduction in net sparing, then the forcer magnitude of a span x on each other span i individually is:

Equation 5.39

$$F_x(i) = \max_i \{ [s_i(x) - \max_{j \neq i, x} s_j(x)], 0 \}$$

and we define its overall forcer magnitude as the largest amount by which span x forces and other span i : ^[6]

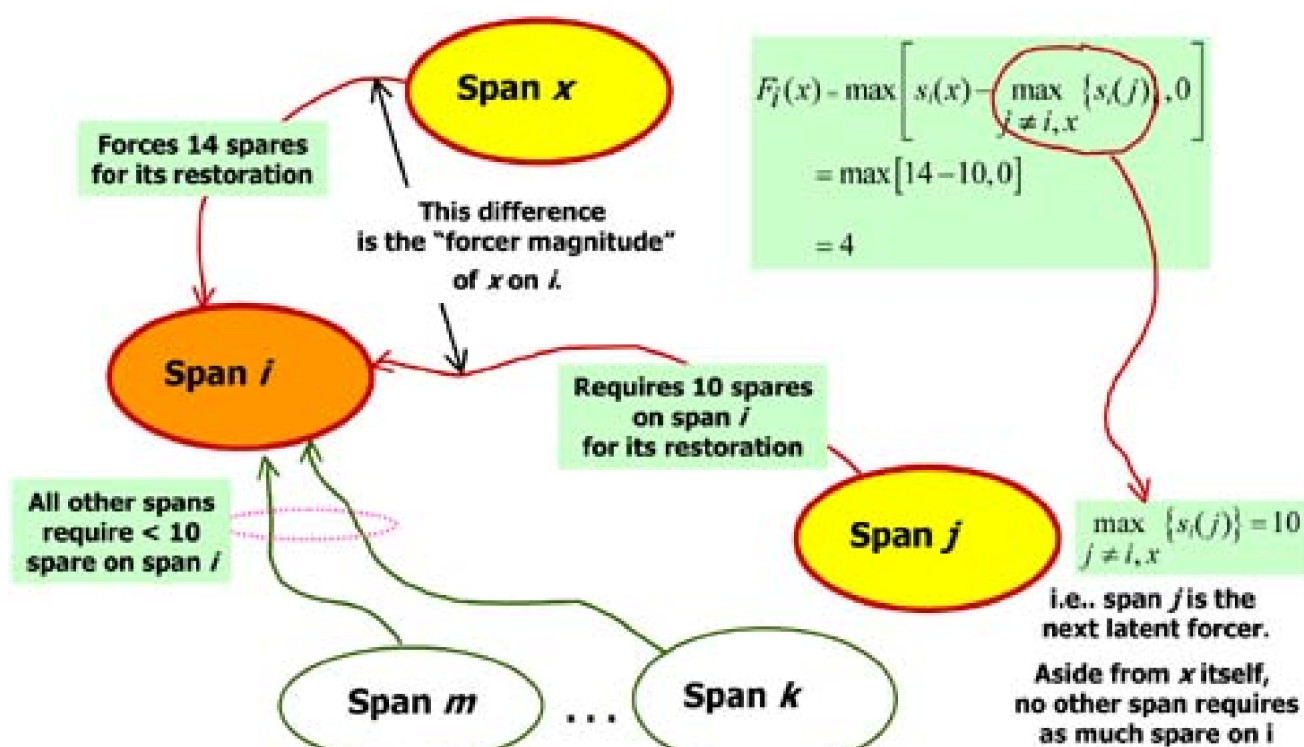
^[6] An alternate possibility is to define the forcing magnitude of a span as the sum of its individual forcer magnitudes on other spans.

Equation 5.40

$$F_x^* = \max_i F_x(i).$$

The second expression in [Equation 5.40](#) obtains the smallest (positive) amount by which span x 's requirement for spares on span i exceeds that of any other span. This is the magnitude by which span x forces span i specifically. In other words, if w_x was lowered by more than $F_x(i)$ units, some other span (the next strongest, or first latent forcer of span i) would emerge as the new forcer of span i . In the upper expression, the largest of these individual forcing magnitudes is taken as the global forcer magnitude of span x . F_x^* represents the number of working channels that would have to be removed from span x before continued removals yield no further reduction in network sparing. [Figure 5-22](#) gives an example showing how this expression is mechanized. Some span x requires 14 spare channels on span i for part of its restoration path-set. Another span j requires ten, and all other spans require ten or fewer. The forcing magnitude of span x on span i is what [Equation 5.40](#) detects; in this case it is the amount of four working capacity units before a next or latent forcer, span j , would become a co-forcer with span x in acting on span i . If the working capacity on span x was reduced by five capacity units, then span j takes over the forcing relationship completely and span x becomes a non-forcer, at least with respect to span i .

Figure 5-22. Example illustrating forcer and latent-forcer relationships.



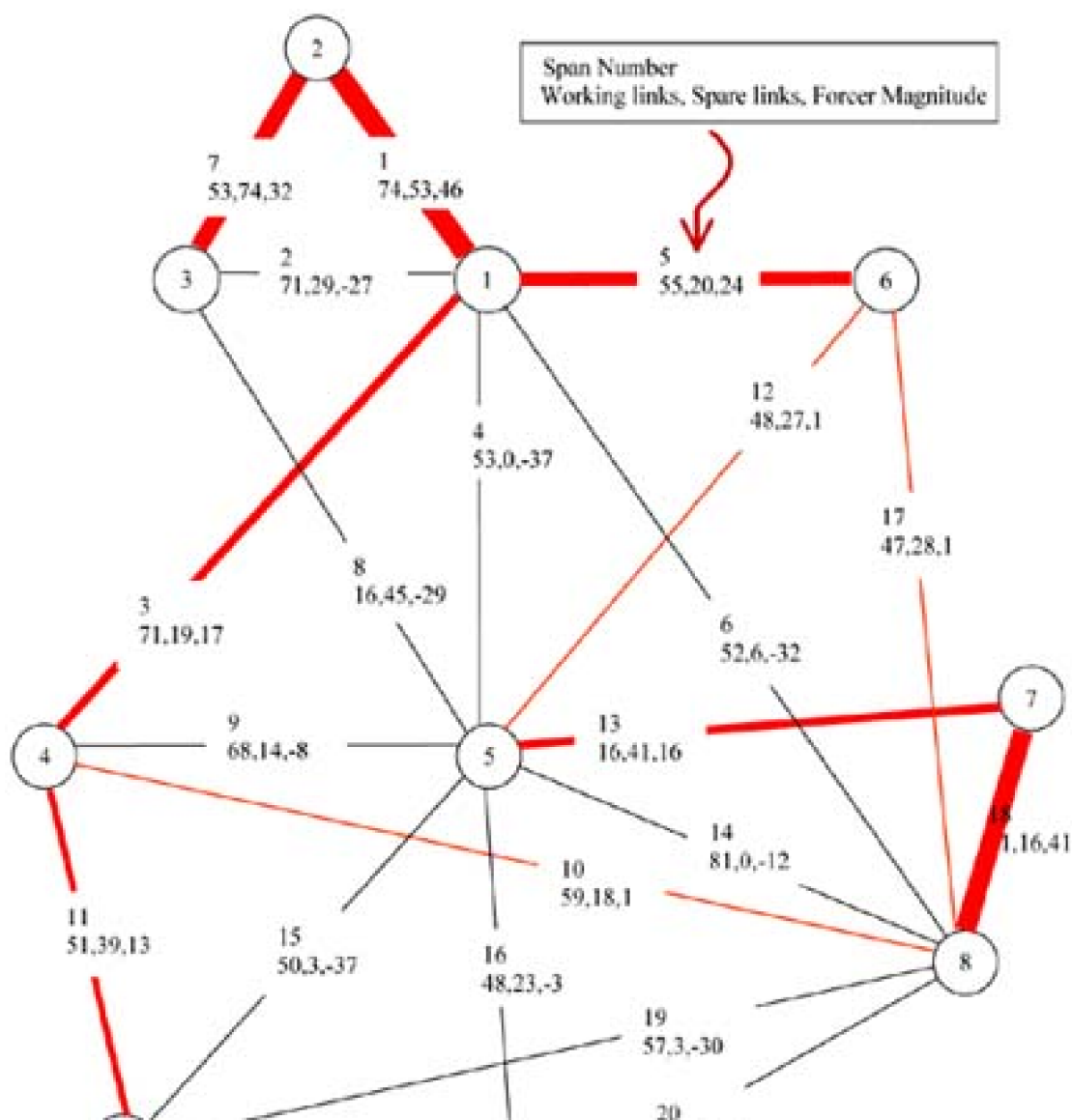
Non-forcers can similarly be characterized by the amount of working capacity by which they are below the level of emergence as a forcer. The negative forcing magnitude is the smallest difference between installed spare capacity on a span i and the spare capacity utilized on that span for failure span x .

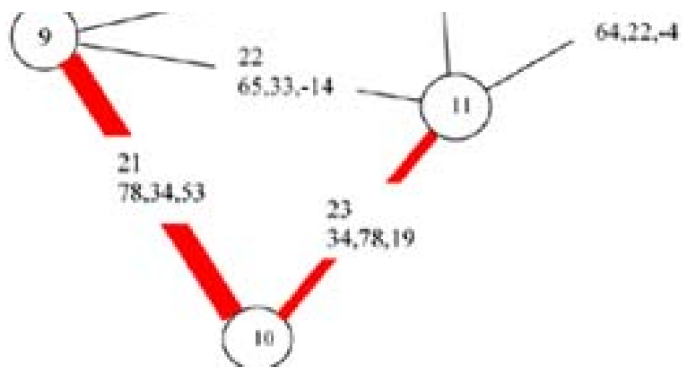
Figure 5-23 is an example result of the full forcer analysis of a less trivial network than we used to introduce the discussion. The top number in each annotation block is the span number. Below it is the number of working channels w_i and spare channels s_i on the span.

This is followed by the forcer magnitude, F_x^* as defined above, in the case of forcer spans. For non-forcers the number of working channels by which the span is below forcer status is shown as a negative forcer magnitude. Span thicknesses are drawn roughly proportional to their forcer strength. The working channel quantities were obtained from shortest path routing of the demand matrix over the topology. The spare channel quantities are those of an optimum SCA plan to support 100% span restoration within a hop limit of 6 spans. The forcer analysis algorithm implements the series of ksp routing trials outlined above with $O(S n \log n)$ time complexity (where S is the number of spans and n is the number of nodes) with an $O(n \log n)$ ksp algorithm. With this approach it takes only a few minutes to obtain the forcer skeleton and forcer magnitudes for networks with up to 100 spans or so.^[7]

^[7] The forcer analysis methods here are complemented at the web site with a more advanced technique that directly discovers forcer status and magnitude of spans through solution of a particular ILP model (used initially for p -cycle forcer analysis in [ShGr03c]). That formulation can also be used for forcer-related maximization of the protected working capacity envelope (PWCE) of a network for a given total investment in spare capacity.

Figure 5-23. Example of a forcer skeleton of a network.





Forcer analysis can be used to guide the routing decisions for provisioning new service growth. By avoiding forcers, one defers updates to the spare capacity plan for survivability. Whenever a span is traversed by a new demand and that span is not a forcer, there is no impact on the survivability of the network. The existing spares are guaranteed still to support 100% restorability despite the new service addition when routed over non-forcers. Where nearly equal length route options exist, the forcer considerations could be applied to choose one route over the other based on minimizing the impact on the network spare capacity plan. Alternately a forcer analysis could be used as feedback for revising the working demand routing in a survivable network design. When the shortest path demand routing has the effect of creating a strong forcer (i.e., forcer threshold far below the w_j value), demand splitting over routes might be effected to reduce the forcer implications, and the SCP design then repeated. In [Chapter 11](#) we will show how forcer analysis can be used in the design of ring-mesh hybrid networks.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

5.6 Modular Span-Restorable Mesh Capacity Design

So far in the design models we have considered, both demands and capacity have been assumed to arise in integer units. While this may be accurate for demands, and for the per-channel costs of turning up new capacity, other aspects of capacity growth have a significant *modularity*. In SONET for instance the practical provisioning modules may typically be only OC-12, OC-48 or OC-192 on a fiber. When DWDM is mature optical transport capacity planning will likely also involve a set of standard modular capacities with technologies optimized for each price-performance point. In fact the actual cost structure of capacity is modular at more than one scale. A certain per-fiber cost may be incurred to turn up a new fiber pair, then a lesser per channel cost for each additional I-channel that is used. In this section we build upon the basic arc-path formulation to realize a considerably more realistic cost model for capacity placement that includes both modularity and economy-of-scale effects and we consider various ways of incorporating modularity in span-restorable mesh design.^[8]

^[8] This section is an adaptation of material published in collaboration with J. Doucette [DoGr00](#).

Economy of scale is an important factor linked to the consideration of modularity. By economy of scale, we mean for example that an OC-192 (four times OC-48 capacity) will generally cost *less* than four times what an OC-48 costs. Similarly the infrastructure to equip a new fiber pair with optical amplifiers and basic filters to support up to 1024 ls should cost less than four times the corresponding infrastructure to support up to 256 ls. The approach here will involve both capacity modularity and economy-of-scale effects directly in the design optimization, not as a post-processing adjustment to an essentially integer solution.

Before proceeding, we should point out that there are at least three contexts where modular capacity designs may not be needed. First, many research studies are based on integer non-modular capacity because the introduction of specific modularity assumptions may soon be out of date and may have a confounding effect on studies aimed at fundamental comparisons of various basic principles and methods. In comparative planning studies for technology assessment in face of many other uncertainties an integer capacity view will usually also give a clearer view of the intrinsic relative capacity requirements of various networking approaches. Assumption of an arbitrary set of capacity modularities in such an advanced planning context can actually obscure the basic comparative insights being sought. In addition, simple post-modularization of integer capacities may also be not far removed from the cost-optimal design if a network is growing rapidly (so any module slack is quickly consumed anyway) or if capacity totals tend to be large anyway relative to the largest modularity (so slack in the last-placed module is a small percentage of total cost). However, it is worth noting in what follows that to fully exploit the advantages of a strong economy of scale in capacity we must do a true modular design and inherently take a joint optimization approach to the capacity planning problem.

There are at least three ways to bring modularity into the arc-path type of SCA formulation. The simplest is to round up the resulting working and spare capacity totals on each span to the nearest (or most economic) combination of available modules. Working demands are routed along the shortest path between their O-D pairs. For comparison to the following approaches, let us refer to this baseline option as Post-Modularized Spare Capacity Allocation (PMSCA).

5.6.1 Modular-Aware Spare Capacity Allocation (MSCA)

A second method is to place spare capacity in a *modular-aware* sense. The working capacity quantities are still based on shortest-path routing, but the design formulation attempts to minimize modular total capacity cost in the allocation of spare capacity. Thus while working capacities remain unaware of modularity, the spare capacity is placed in a way that inherently recognizes that the ultimate capacity totals will have to be modular. This is a compromise in that capacity decisions are modular but any advantages from coordinating working path routing with modular capacity decisions are not captured. The new sets, parameters and variables for this and the subsequent formulation are:

- M = A set of modular capacity sizes, index m .

- z^m = Number of capacity units in the m^{th} module size.
- c_j^m = Cost of a module of the m^{th} capacity size installed on span j . Economy of scale is introduced through these coefficients.
- n_j^m = Number of modules of the m^{th} size added on span j (integer variable).

MSCA

Minimize

Equation 5.41

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m$$

subject to:

Equation 5.42

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

Equation 5.43

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p \quad \forall i, j \in S^2 | i \neq j$$

Equation 5.44

$$s_j + w_j \leq \sum_{m \in M} n_j^m \cdot z^m \quad \forall j \in S$$

Equation 5.45

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i$$

Equation 5.46

$$s_j \geq 0 \quad n_j^m \geq 0 \quad \text{integer} \quad \forall j \in S \quad \forall m \in M$$

The objective function is now the total cost of modules placed. The design employs a set M of available module types, indexed by m , each with an associated capacity Z^m . C_j^m represents the cost of placement of a module of type m on span j which may depend on the length or type of facility route upon which span j is based. It is in these coefficients that any particular capacity versus cost schedule involving economy of scale can be reflected. Although C_j^m can reflect nonlinear capacity versus cost characteristics, note that there is no problem in representing this in an LP or ILP. The nonlinearity is in the sequence of capacity versus cost points formed by the C_j^m , describing the available transmission system options, but the summation on them in the objective function is still linear. n_j^m are the new solution variables. They indicate the number of type m modules to install on each span j . In MSCA w_j continues to be an input parameter. s_j remains in the formulation but is now really just an internal variable representing the logically required number of spare channels on each span. In this regard it is important to note that Equations () and (5.43) are exactly as they appear in SCA. Thus, the notions of logical working flow and logical spare capacity carry on but the s_j requirements now only indirectly drive the objective function. Decisions about spare capacity allocations, and hence the related restoration flow assignments, are now influenced by how the resultant working plus spare totals on each span map into module requirements and costs on each span through [Equation 5.44](#).

5.6.2 Modular Joint Capacity Allocation (MJCA)

This model simultaneously decides on the working path routings and the logical placement of spare capacity so that total cost of the required modules is minimized. This approach has the capability to detour the routing of some working flows (and adjust their corresponding restoration routes) to take advantage of the large but proportionally lower cost high-capacity modules. MJCA is the complete no-compromise design model from the standpoint of planning the routing, restoration and capacity placement for a mesh network to fully exploit economy of scale in transport technologies. An interesting effect to follow is that under moderate to strong economy-of-scale effects, the result can even be to completely disuse certain spans of the network. The MJCA model uses parameters and variables above and those pertaining to the variable routing of working paths defined in [Section 5.4.1](#) for the non-modular joint optimization problem.

MJCA

Minimize

Equation 5.47

$$\sum_{m \in M} \sum_{j \in S} C_j^m \cdot n_j^m$$

subject to:

- routability:

Equation 5.48

$$\sum_{q \in Q} g^{r,q} = d^r \quad \forall r \in D$$

- logical working capacity:

Equation 5.49

$$w_i = \sum_{r \in D} \sum_{q \in Q} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall i \in S$$

- restorability:

Equation 5.50

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

- logical spare capacity:

Equation 5.51

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p \quad \forall i, j \in S^2 | i \neq j$$

- modular total capacity:

Equation 5.52

$$s_j + w_j \leq \sum_{m \in M} n_j^m \cdot z^m \quad \forall j \in S$$

- bounds:

Equation 5.53

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i \quad g^{r,q} \geq 0 \quad \forall r \in D \quad \forall q \in Q^r$$

Equation 5.54

$$s_j \geq 0 \quad w_j \geq 0 \quad \text{integer} \quad \forall j \in S \quad n_j^m \geq 0 \quad \text{integer} \quad \forall j \in S \quad \forall m \in M$$

The objective function and the third and fourth constraint families, which assert restorability and (logical) spare capacity for restorability, are the same as in the SCA and MSCA. The first two constraints are similarly imported from JCA to assert demand routing and working capacity to support demand routing. And they are all linked together under the modular total capacity expression, [Equation 5.52](#), which is also imported from MSCA.

This is the most truly modular-oriented of the three design models PMSCA, MSCA and MJCA. Here, the routes of working paths and the selection of restoration routes are jointly coordinated with regard to the modular capacities and economy-of-scale opportunities. Working and spare capacities are now only logical channel classifications, not directly associated with cost. Although the LP/ILP solver does not work incrementally on the routing decision for each demand, we will be able to understand much of the changed behavior in this model by imagining that under [Equation 5.52](#), a certain module type has already been committed to in the objective function, but not "filled" with $w_j + s_j$. In such circumstances other routes can be pictured being attracted toward this module because it has already-decided but unused (i.e., notionally cost-free) capacity, if thought of incrementally. The price for this extra ability to exploit modularity is essentially that associated with going to a joint optimization: we have to define an eligible route set for routing each *working* demand in the prefailure network, as well as the usual eligible route set for restoration of each failure scenario. In contrast to our prior discussions about populating the eligible route sets for working paths, to fully exploit economy of scale in an environment of strongly modular capacity, it would be more important under MJCA to represent as many distinct route possibilities as possible for both working and restoration flows so that they may be allowed to go farther "out of their way" to take advantage of economy-of-scale effects in large modules.

In MSCA and MJCA as stated so far, logical working, logical spare, and module quantity decision variables are all integer. A practical and effective relaxation is, however, to insist only that module decision quantities are integer, and relax all flow and logical capacity variables. Especially when the scale of the capacity modularity is large relative to a typical d^f quantity, (i.e., the design notionally has a coarse modular capacity structure), a validation test for integer feasibility of demand routing and restorability will almost always be passed without any subsequent repair needed at all, due simply to the additional capacity margins that are provided on most spans in a strongly modular design. An interesting practical implication of this, and in fact an advantage of modular cost-minimized design, is therefore to reverse or dismiss the prior need to test and "repair" flow-relaxed solutions to obtain strictly integer solutions, such as we did with methods in [Section 5.3.4](#) and [Section 5.3.7](#). In effect when a design is highly modular in its capacity structure, and especially with economy of scale, we can skip the intermediate concern about reconstructing integral flows to obtain integral logical capacities and simply validate directly that integral working and restoration flows are feasible under the final set of module capacities.

5.6.3 Comparative Results with the Modular Design Formulations

In [\[DoGr00\]](#) the above approaches to modularity were compared using a variety of topologies and demand patterns. Working route hop limits and restoration route hop limits were assigned so that there were at least five and typically fewer than 20 eligible working routes for each demand pair in the MJCA model, and the same number of eligible routes for each span failure in all models. A family of five module sizes were used, corresponding to 12, 24, 48, 96, and 192 unit-capacities per module. Three different economy-of-scale models were also tested, namely 3x2x, 4x2x, and 6x2x where 3x2x means, for example, that a tripling of capacity results in a doubling of cost, and so on. The important logical features of the modular capacities used is that four doublings of capacity are represented. The capacity units can be interpreted as OC-n levels or DWDM waveband capacities (although it is not yet clear which standard modularities will arise for DWDM networks). All designs support 100% restorability to any single-span failure. The results are summarized for the PMSCA in [Table 5-1](#) followed by results for the 3x2x, 4x2x economy-of-scale cases for MSCA and MJCA in [Tables 5-2](#) and [5-3](#), respectively.

Table 5-1. Post-modularized spare capacity designs (PMSCA)

Network	Req'd Capacity	Size 12 Modules	Size 24 Modules	Size 48 Modules	Size 96 Modules	Size 192 Modules	3x2x Cost	4x2x Cost	6x2x Cost	Module Capacity
JED9807b	106	14	0	0	0	0	1680	1680	1680	168
Bellcore1	711	0	5	18	0	0	6114	5170	4475	984
Bellcore2	1315	3	3	24	4	0	9614	7986	6823	1644
9n17s1	319	0	15	2	0	0	3366	3030	2765	456
9n17s2	361	0	13	4	0	0	3570	3170	2861	504
10n19s1	540	1	5	14	0	0	5082	4330	3775	804
10n19s2	545	1	10	9	0	0	4572	3980	3535	684
11n21s1	653	4	10	11	0	0	5508	4820	4305	816
11n21s2	628	2	8	13	0	0	5472	4720	4161	840

In each table, column 2 shows the total logical design capacity which is the sum of all internal $s_j + w_j$ quantities. The next five columns give an accounting of the exact numbers of 12, 24, 48, 96, and 192 unit-capacity modules used in each design. The "Cost" column presents the total cost of implementing each design. In [Table 5-1](#) this is split into three columns, one for each economy-of-scale model. By comparing logical capacities required by the PMSCA design with the actual modular total capacities after rounding up it was found that 20% to 37% (average of 26.8%) of the total capacity in these PMSCA networks is excess capacity purely due to modular up-rounding effects.

In the tables for MJCA and MSCA results, the "Span Elim." column represents the number of spans actually not used at all (i.e., "eliminated" from the network topology). These are spans that exist in the facility graph but turn out to be assigned zero working and spare capacity in the result. This is an effect which can only emerge from modularity and economy-of-scale cost effects combining to "pull" working and spare capacity away from spans where it would have been in a purely minimum capacity (as opposed to a minimum modular cost) design result. As a result MSCA designs have no span eliminations because shortest path routing of the working demands used every span at least once, while MJCA designs exhibited some true span elimination effects.

Table 5-2. Modular-aware spare capacity designs (MSCA)

(a) 3x2x Economy of Scale

Network	Req'd Capacity	Size 12 Modules	Size 24 Modules	Size 48 Modules	Size 96 Modules	Size 192 Modules	Cost	Span Elim.	Tot. Mod. Capacity	% Cost Improvement
JED9807b	111	14	0	0	0	0	1680	N/A	168	0.0%
Bellcore1	741	1	9	14	0	0	5826	N/A	900	4.7%
Bellcore2	1422	0	1	19	8	0	9226	N/A	1704	4.0%
9n17s1	345	3	13	1	0	0	3066	N/A	396	8.9%
9n17s2	391	3	10	4	0	0	3372	N/A	468	5.5%
10n19s1	566	1	11	8	0	0	4470	N/A	660	12.0%
10n19s2	585	2	10	6	1	0	4274	N/A	648	6.5%
11n21s1	709	0	13	8	1	0	5168	N/A	792	6.2%
11n21s2	668	1	10	11	0	0	5148	N/A	780	5.9%
Average										6.0%

(b) 4x2x Economy of Scale

Network	Req'd Capacity	Size 12 Modules	Size 24 Modules	Size 48 Modules	Size 96 Modules	Size 192 Modules	Cost	Span Elim.	Tot. Mod. Capacity	% Cost Improvement
JED9807b	109	14	0	0	0	0	1680	N/A	168	0.0%
Bellcore1	744	1	7	14	1	0	5009	N/A	948	3.1%
Bellcore2	1424	1	4	13	10	0	7310	N/A	1692	8.5%
9n17s1	362	2	13	2	0	0	2930	N/A	432	3.3%
9n17s2	382	3	11	3	0	0	2950	N/A	444	6.9%
10n19s1	573	0	12	6	1	0	3819	N/A	672	11.8%
10n19s2	580	1	10	7	1	0	3839	N/A	684	3.5%
11n21s1	738	1	13	4	4	0	4646	N/A	900	3.6%
11n21s2	666	0	9	10	2	0	4608	N/A	888	2.4%
Average										4.8%

Table 5-3. Jointly optimized modular mesh designs (MJCA)

(a) 3x2x Economy of Scale

Network	Req'd Capacity	Size 12 Modules	Size 24 Modules	Size 48 Modules	Size 96 Modules	Size 192 Modules	Cost	Span Elim.	Tot. Mod. Capacity	% Cost Improvement
JED9807b	134	12	0	0	0	0	1440	2	144	14.3%
Bellcore1	888	0	2	18	0	0	5556	3	912	9.1%
Bellcore2	1441	1	6	15	6	0	8232	1	1452	14.4%
9n17s1	385	1	14	1	0	0	3012	1	396	10.5%

9n17s2	398	4	11	2	0	0	3102	0	408	13.1%
10n19s1	579	0	13	6	0	0	4146	0	600	18.4%
10n19s2	628	1	8	9	0	0	4200	1	636	8.1%
11n21s1	761	1	9	10	1	0	5120	1	804	7.0%
11n21s2	786	4	4	12	1	0	5126	1	816	6.3%
Average								1.1		11.3%

(b) 4x2x Economy of Scale

Network	Req'd Capacity	Size 12 Modules	Size 24 Modules	Size 48 Modules	Size 96 Modules	Size 192 Modules	Cost	Span Elim.	Tot. Mod. Capacity	% Cost Improvement
JED9807b	174	3	6	0	0	0	1380	5	180	17.9%
Bellcore1	1091	0	1	11	6	0	4844	5	1128	6.3%
Bellcore2	1441	1	6	15	6	0	6774	1	1452	15.2%
9n17s1	398	0	14	2	0	0	2860	1	432	5.6%
9n17s2	398	4	11	2	0	0	2830	0	408	10.7%
10n19s1	615	0	10	6	2	0	3818	1	720	11.8%
10n19s2	777	0	1	11	3	0	3827	4	840	3.8%
11n21s1	908	0	2	12	4	0	4576	3	1008	5.1%
11n21s2	1050	1	0	5	9	0	4371	6	1116	7.4%
Average								2.9		9.3%

The "Tot. Mod. Capacity" column shows the total modular capacity actually provided, i.e., the sum of the capacities of all modules called for in the design. MSCA and MJCA result tables also have a final column where the total cost for each design is normalized to the post-modularized benchmark cost (PMSCA) for that network and economy-of-scale model. Results showed that the required (logical) capacity using MSCA increases but this is as expected to allow exploitation of modules with economy-of-scale benefits. MSCA allows for more indirect restoration routing of failed wavelengths to take advantage of large modules elsewhere in the network, so there is an increase in total capacity required. However, in nearly all cases, the total module capacity decreased relative to simple post-modularized designs and there was a reduction in the proportion of unused module capacity. Thus, we are seeing the restoration path sets being redirected to an extent to exploit modular capacity placement opportunities. This is an important difference of the modular-aware SCA approach over simple post-modularization approaches. Under MSCA only 19.4% of total modular capacity placed is in excess of the logical channel counts supported in the design (i.e., the total modular capacity placed is 19.4% higher than the sum of all $w_j + s_j$ channel counts in the design). Cost savings realized with MSCA over PMSCA averaged 4.7%, and were as high as 12%.

However, the greatest improvements in design efficiency and costs arose with the MJCA formulation (even though in some cases fully optimal terminations were not obtained). Total cost savings as high as 22.5% were realized (with an average of 10.7%) and excess modular capacity was decreased to an average of 4.7%. This is all consistent with the notion that in MJCA both working and restoration paths are being carefully coordinated to share well-filled, cost-effectively chosen modules, of the largest sizes that can be justified by economy-of-scale benefits. A significant number of the test cases also exhibited instances of spontaneous span elimination under MJCA. These are cases where the cost-optimal architecture simply did not place any capacity on certain spans. This occurred most frequently in the 6x2x economy-of-scale model (data not shown here) where the economic pressure to aggregate routes together was at its greatest but also was seen under the weakest capacity versus cost nonlinearity (3x2x). Such span eliminations under MJCA can help us in the determination of the best basic graph topology for mesh networks. This is taken up further in [Chapter 9](#).

5.6.4 Modeling Modularity with Per-Channel Provisioning Costs

As so far presented, the aspect of modular capacity is one where the complete capacity of a system is available to the design if the module is placed. In other words, a modular transmission system is assumed to have all its channels turned up when installed. This fits with the paradigm of full pre-provisioning that is implicit in many studies of dynamic demand provisioning.^[9] In effect the cost is incurred to turn up all channels in advance, in a "build it and they will come" sort of approach. (Capacity installation precedes demand arrival.) Historically, however, channel provisioning for anything as expensive as a WDM lightpath usually waits until the demand actually arises. In the latter environment a second finer level of step-wise cost structure characterizes the transmission systems. The first modular scale (the one we have been talking about) establishes a new system. The second finer modular scale exists if channels are not turned up all at once (which makes their cost just part of the total module cost), but are instead turned up individually to match actual w_j and s_j requirements. For example, a DWDM link system with ultimate capacity of 80 wavelength channels may only have, say, 43 channels currently required and installed.

^[9] Frequently, studies in optical networking adopt network models that assume exactly 8, 16, 32 or 64 etc. wavelength channels available on every span. This is convenient for study of RWA problems (because one instance of the same set of l_s exist on each span), but it is not reflective of real networks in that it presumes: a maximal pre-provisioning investment (all possible channels are turned up on every span) and it fails to recognize that suitably placed differential amounts of capacity on spans is always an aspect of an *economic* design. Even if every node pair exchanges identical demand, once this is routed each span will have different characteristic capacity requirement based on its position in the graph. Even if the DWDM technology supports up to 512 channels, say, it is always still economic only to turn-up the number of channels actually required for networking.

This refinement is not difficult to add to the models, however, because the modular formulations still implicitly resolve both w_j and s_j logical channel requirements and it is these that directly imply the channel counts to be turned up within each modular system. Thus, if desired, we can reflect this by changing the objective function for MSCA and MJCA to:

Minimize

Equation 5.55

$$\left\{ \sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m + \sum_{j \in S} c_j \cdot (w_j + s_j) \right\}$$

where c_j is the per-channel cost of equipping an additional wavelength channel on an already installed system on span j and c_j^m is redefined to represent only the common equipment cost for establishing a system of type m on span j . A further extension to make the per-channel cost also dependent on the system type follows easily. This is not much more complex to solve but requires a more detailed set of cost assumptions where the per-channel provisioning costs are separated from the complete-system common costs. This will tend to push the solutions back in the direction of shortest path working routes, however, also making the outright elimination of network spans less likely. A practical choice the planner would have to make is whether to work with the simpler objective function of [Equation 5.47](#) or to allocate costs to both per-channel and per-module usage as in [Equation 5.55](#). Of course, if full pre-provisioning is the policy to support dynamic demand services, then the former is fully justified and the c_j^m values should be for fully equipped modular configurations. However, there are other circumstances where the single-modular cost structure may still be quite characteristic of the actual economics:

1. If the rate of growth is fast enough, and the cost of dispatching maintenance crews to populate new channel filters, laser and receiver cards, etc., one by one is high enough that it makes sense to simply fully equip the system when installed (same result but different reason for pre-provisioning).
2. If the "per-channel" incremental costs are relatively small compared to the get-started investment required to establish the common equipment parts of the system.
3. If the resulting designs exhibit high system utilization in the solution.
- 4.

If the c_j^m coefficients for the "fully equipped" model are based on common equipment costs plus a characteristic average

fill or utilization factor for the per-channel costs.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

5.7 A Generic Policy for Generating Eligible Route Sets

At this point we have completed our treatment of the basic SCA, JCA and MJCA models for span-restorable mesh capacity design. These are all arc-path types of formulation which require preprocessing of the graph to represent the sets of eligible routes for restoration and, in joint formulations, separate sets of eligible routes for working path assignment. In passing we mentioned Herzberg's hop limit approach to populating the restoration route sets. Ideally, whenever possible this should be followed for the restoration routes: i.e., include all distinct routes up to and including the threshold hop limit. However, for large problems, or cases as mentioned where a few spans may require high individual hop limits, a more practical procedure is needed. In addition we need a policy for generating eligible working route sets for use in joint formulations. In practice high quality solutions to this type of problem result if at least ten or more different routes are possible for each purpose. In cases where it is important to know that one is extremely close to the strict optimum, but the complete route sets are too large to represent, then sensitivity tests should be conducted wherein the route sets are changed in detailed composition and in size to see if the objective value responds more than say 1% or so to the test changes in composition and size of the route sets. Note in this regard that real-world engineering restrictions on the maximum length of any restoration path can be welcome inputs as they may largely make this issue disappear. If engineering considerations specify $H < 6$ (or something equivalent in terms of total distance) then it can be much easier to obtain designs that are optimal, given this external limitation.

5.7.1 Generating Eligible Restoration Route Sets

More generally, for any design problem where the set of failure scenarios is all single-span failures, $|S|$ sets of eligible restoration routes are needed, one for each span failure. Each route in these sets extends between the end-nodes of the corresponding failure span only. Three approaches can be considered for this problem.

1. Generate all distinct routes up to a compromise hop limit, H (typically at most 6 or 7 hops), and supplement or enrich this set with the set of k -successively shortest span-disjoint routes found without any hop limit. The set of supplement k -shortest disjoint routes is relatively small, but ensures full representation of the network's topological connectivity between all nodes, which a simple hop limit may fail to do in some networks. This approach was used in [\[Iras96\]](#) [\[IrMa98\]](#).
2. Set a minimum target number of distinct eligible routes for restoration of each span and adjust H for each span so that at least the target number of eligible routes are obtained. A minimum H may also be specified. A simple program to do this can be based on repeated calls to a hop limited all-distinct routes routine. It starts with the minimum H and increases it until an H is reached at which the target number of distinct routes is met or exceeded. All distinct routes at that H value are then accepted. This policy has the advantage of populating all route sets to at least contain all distinct routes under the minimum H , but adaptively increasing H as needed for the topological conditions surrounding each span, so that in all cases there is at least the target number of distinct routes.
3. Since many different studies are sometimes done on the same basic network graph (topology changes are relatively rare), the most comprehensive approach is to literally generate a master database of all the distinct routes for each span failure. This might take some time and produce a large file, but once the master route set is available, the route representations for specific problem formulations can easily be generated by filter programs according to almost any desired specification. Examples would be: all routes under 3,000 miles or six hops, all routes that exclude certain nodes or spans, all routes up a hop limit that provides at least 15 per span, a set of routes that visit no node more than x times, the first 15 routes when sorted by increasing length, etc. An additional advantage of this comprehensive approach is that the sensitivity testing mentioned above is easily based on testing slightly different types and numbers of selected eligible routes for various trial cases.

None of these methods is necessarily exclusive of the others. One can mix and match batches of eligible routes selected by different criteria to make up the final route sets that the problem is run with. For efficiency duplication should be avoided, however. A good general practice would be to select a baseline set of routes by either method 2 or 3, and always supplement with the k sp disjoint route set, limited only by whatever ultimate engineering limit is stated on any path length. Philosophically, it can be quite justifiable to experiment with route-enumeration strategies to find an effective policy for solution to practical problems rather than starting from scratch

on developing a purpose-specific heuristic. Especially in academic literature, one often hears the sentiment that "the ILP approach is NP-hard, and so we develop pure heuristic." The pure heuristic may then take enormous development effort and still produce results of unknown accuracy. But it is somewhat illogical to consider perfect ILP solutions or pure heuristics as the only options. ILP is really only impractical if we presume complete termination and use of complete route sets, etc. As we have argued elsewhere, what this misses is the usually rich intermediate ground of extremely effective—and almost immediately accessible—heuristics based on the approximate solution of exact ILP models with these kinds of route-rationing and/or run time-limited strategies.^[10]

^[10] In the same spirit we offer the viewpoint that "one of the best (and easiest) heuristics for anything is the first feasible solution that CPLEX produces for it. Spend 15 minutes to get even better." It is almost irrelevant if full termination would take eons after that, especially if the solution gap is already under 5%.

5.7.2 Generating Eligible Route Sets for Working Paths

In joint problems we need up to $n(n-1)/2$ sets of eligible routes for working path assignment where $n = |N|$ if every O-D pair exchanges non-zero demand quantities. The larger number of these route sets is countered by the fact that they usually don't need to be as large as restoration route sets because even in joint designs, working paths tend not to go far from shortest paths. The master database approach is an option for generating these route sets as well, but two more basic procedural approaches are:

1. Set a limit on the amount by which any working path would be permitted to exceed its shortest route over the graph. This should be set as a percentage (typically not more than 10% or 15%) of the shortest path distance. Then, for each O-D pair, find the shortest path, and hence the allowance for excess distance in the eligible working route set. A mixed criterion of both excess distance and an excess hop allowance may be used. Then enumerate all distinct routes between O-D nodes within the excess hop/distance limits.
2. Set a minimum target number of eligible working routes for each O-D pair and, as in [Section 5.7.1](#) (#2), adaptively increase the distance limits while enumerating all distinct routes until this target is reached. Take all distinct routes found at that threshold value.

Results in this book are based on complete enumeration of restoration path-sets up to the threshold hop limit (see [Section 5.3.6](#)) unless indicated otherwise. Where indicated, they are often based on the second type of enumeration above to a (minimum) target number of eligible routes in all cases. When restoration and working routes are both needed we refer to the practice as an "N + M" policy. A typical policy we use is "20 + 10" meaning that there are *at least* 20 restoration routes for each span failure and *at least* 10 eligible working routes for each O-D pair.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

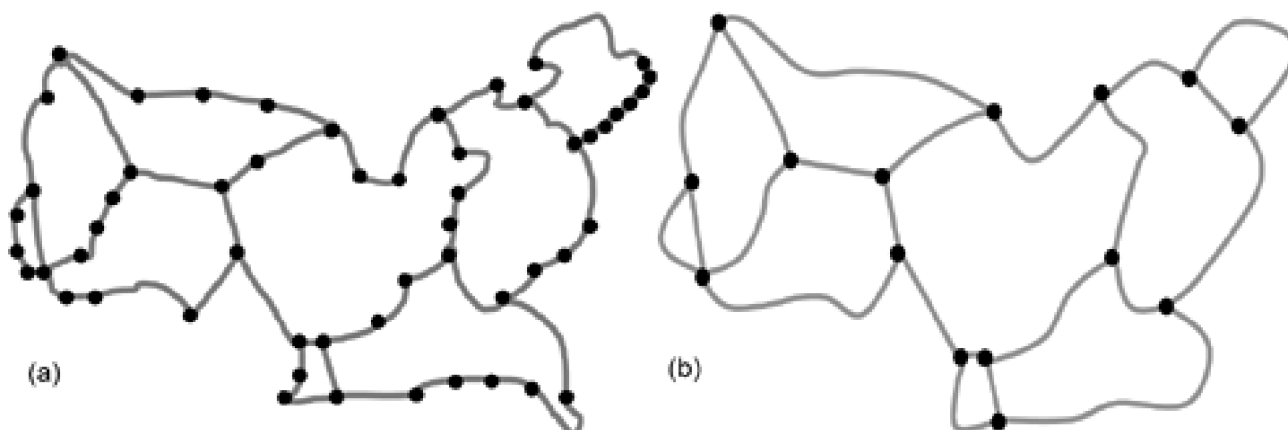
5.8 Chain Optimized Mesh Design for Low Connectivity Graphs

Here we introduce a refinement to the basic model for span-restorable capacity design to increase the capacity efficiency on especially sparse facility graphs.^[11] Chains are natural features of sparse networks and very sparse networks represent an especially challenging case for achieving spare capacity efficiency. Such low average nodal degree is an important reality of several North American Inter-Exchange Carrier (IXC) (long-haul backbone) networks in particular. While European networks often have $\bar{d} > 4$, some North American networks can be extremely sparse, with \bar{d} as low as ~ 2.2 . (Keep in mind the graph requires at least $\bar{d} = 2$ just to be closed.) And it is not easy to increase \bar{d} simply by acquiring more rights-of-way. Right-of-way costs are one of the largest investments the network operator faces so it is not easy to just increase connectivity as the means increase mesh efficiency. The meta-mesh idea helps in these cases.

^[11] This section is an adaptation of work previously published in [GrDo02](#).

The method that follows focuses on the way chain subnetworks are treated under span restoration. In a sparse but still biconnected network it follows that there will be a preponderance of degree-2 locations. Every degree-2 node can be thought of as being part of a chain subnetwork of at least two spans. Often, however, degree-2 nodes will aggregate into more extensive chains, like beads on a string. [Figure 5-24\(a\)](#) gives an example of this tendency as clearly exhibited in at least one real life network [Level3](#). The example has 55 nodes and 62 spans, for an overall $\bar{d} = 2.25$. Inspection shows, however, that the network can also be decomposed into 14 chain subnetworks and seven direct spans. By definition, chains are bounded on each end by a node with $d \geq 3$ which we will refer to as the *anchor nodes* of the chain.

Figure 5-24. Example of a sparse topology (a) comprising 14 chains (55 nodes, 62 spans) at $\bar{d} \sim 2.2$ and its "meta-mesh" abstraction (b) of 15 nodes and 23 spans at $\bar{d} \sim 3.1$.



5.8.1 The Meta-Mesh Concept

The logical structure of chains and direct spans is abstracted in [Figure 5-24\(b\)](#) which we call the *meta-mesh* topology of the original graph. The so-called meta-mesh is not a higher layer network in the usual sense nor is it a subnetwork. Rather it is the topology that arises when

all direct spans and chain subnetworks are viewed equivalently as edges of another graph: the meta-mesh graph. Equivalently, the meta-mesh is the topology obtained when nodes of only degree 3 or higher are considered and no further distinction is made between direct physical spans and chain subnetworks. Both are considered just logical spans of the meta-mesh. In formal terms the meta-mesh topology is a homeomorphism of the full graph.

The significance of the meta-mesh is that it is only at this level of abstraction that efficient mesh sharing of spare capacity *can* arise. Note that while the complete network has 55 nodes, 62 spans and $\bar{d} = 2.25$, the meta-mesh has only 15 nodes and 23 "spans" with $\bar{d} = 3.08$. By its nature, *the meta-mesh graph is always of at least degree 3*. The potential difference in efficiency of a span-restorable mesh on the full network versus the meta-mesh can be appreciated from the $1/(\bar{d}-1)$ lower bound on redundancy derived earlier. The lower bound says that if we solve the SCA (or JCA) problem on the entire network the best we could ever do is achieve a redundancy of $1/(2.25-1) = 80\%$. On the other hand, as an abstract proposition at this stage, a span-restorable design on the meta-mesh graph could approach $1/(3.08-1) = 48\%$ redundancy. These are only lower bounds, not achievable in general, but it gives a demonstration of significant potential for efficiency increases if we could somehow achieve restoration with the efficiency of the meta-mesh, not the full network. The chain bypass technique to follow partly achieves this efficiency. To see how this happens we need to first consider how the span-restorable mesh design models in previous sections will handle chains.

5.8.2 Chain Subnetworks Under Ordinary SCA or JCA

Figure 5-25 portrays a chain of three degree-2 sites and a set of working capacity accumulations (w_{Tot}) resulting from the routing of demands in the network. For the moment, consider only the w_{Tot} values. These w_{Tot} values may be the accumulation of demands crossing the span from shortest-path routing or from a joint capacity design. Under the normal span restoration model, the entire chain must have spare capacity sufficient to support the loopback routing of the largest working capacity in the chain. Strictly, one only needs spare capacity on the largest span to match the second largest working capacity of the chain, but otherwise the conventional model will capacitate chains essentially as if they were sections of BLSR-type rings because the routing to escape from the failure span has no other options under span restoration within the chain. Thus in the example, the worst-case cut is of span 2-3 span at 440 working units and the spare capacity allocation within the chain would be as shown in Figure 5-26.

Figure 5-25. An example chain of three degree-two nodes between two mesh anchor nodes.

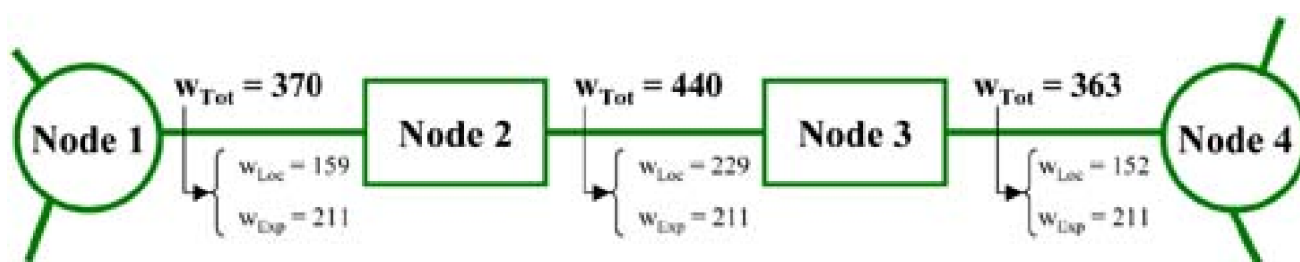
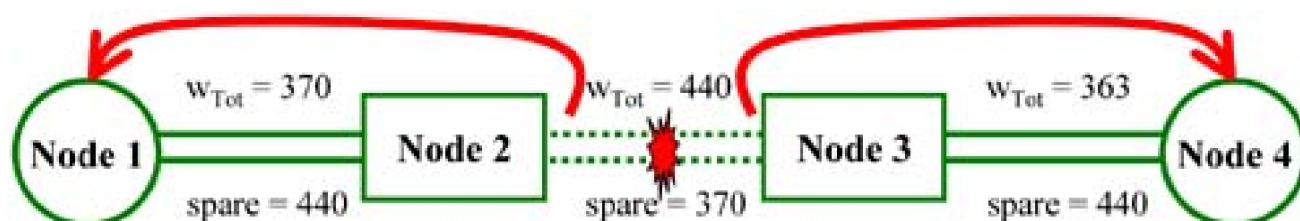


Figure 5-26. The spare on a chain is determined by the size of the largest working total.



5.8.3 Logical Chain Bypass Spans

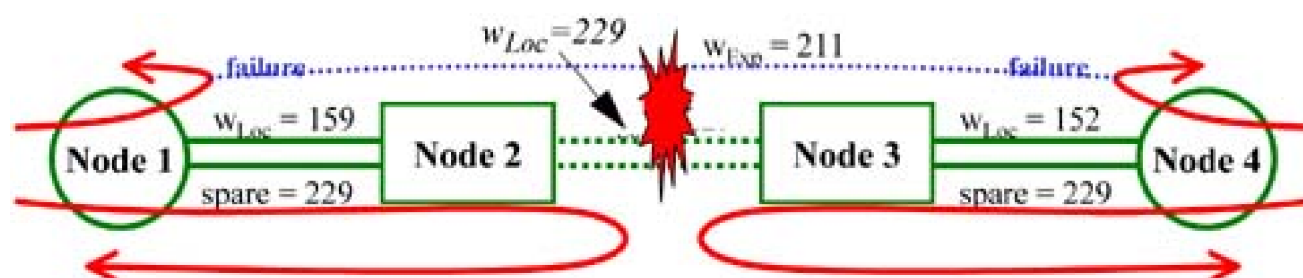
A way to improve upon this situation is to create a logical bypass or express span through the chain that will handle any working flow that completely transits the chain. To see how this works, let us reconsider the w_{Tot} values in [Figure 5-25](#). In general, such accumulations of working flows will contain some demands originating or terminating within the chain, and others passing completely through the chain. For discussion, say the breakdown of the w_{Tot} values in [Figure 5-25](#) is as shown in the brackets below each w_{Tot} value. The w_{Loc} values are *intra-chain* working capacity totals that arise only from demands that originate or terminate at one of the nodes in the chain. If the w_{Loc} and w_{Tot} values are known, any difference remaining must be express flow, denoted w_{Exp} . These are an accumulation of working demands that are flowing entirely through the chain and arise either from demands for which nodes 1 or 4 are the origin or destination, or which otherwise flow entirely through this chain on general paths across the network as a whole.

Now we can observe that, if the breakdown of local and express flow through the chain is as shown in [Figure 5-25](#), then the conventional design is providing loopback spare capacity for the entire cross-section of both local and express flows. Thus, the normal response to any span cut in the chain is to return both local and express flows via loopback to the anchor nodes. From there, the restoration rerouting can take more general routes through the rest of the graph. One can argue therefore, that there is no need to explicitly loop back the express flows to the anchor nodes. Rather, the express component of the working flow crossing the failure span could be returned to the anchor nodes simply by letting those demands fail all the way back to the anchor nodes. The advantage in treating the express flows this way is that there will be no spare capacity required within the chain itself for restoration of any express flows, other than an allocation of spare that may be made to the chain from a globally efficient meta-mesh design standpoint.

5.8.4 Restoration in the Bypassed Chains

[Figure 5-27](#) shows the difference this makes in the example of [Figure 5-25](#), in terms of both capacity and the reconfiguration that is undertaken for restoration. Intra-chain working flows must be matched by loopback type spare capacity so they can escape back to the anchor nodes with the demand composition they had at the particular location of the break. All express flow through the chain automatically "fails back" to the anchor nodes. In effect the anchor nodes become the custodial nodes for the restoration of the express flow over the logical chain bypass span between them. No spare capacity is therefore needed within the chain for the express flows. In the example of [Figure 5-25](#), the spare capacity requirement is thus revised downwards to that in [Figure 5-27](#). Failed-back express flows and looped-back local flows are logically unified back at the anchor nodes and treated as a total amount of failed working capacity on the corresponding span of the meta-mesh. From there on, nodes 1 and 4 cooperate as a conventional pair of custodial nodes for span restoration (often called Sender-Chooser nodes in the context of a DRA), but they operate within the logical meta-mesh of OXC nodes, not the full network.

Figure 5-27. How the spare capacity and restoration of a chain subnetwork is affected by restoring the express demand flow in the meta-mesh, not in the chain subnetwork.



5.8.5 Design Method to Effect the Meta-Mesh Concept

To put these ideas into effect in the design of a span-restorable network there are three main considerations relative to a conventional SCA or JCA design model.

1. Representing the Augmented Logical Topology

One change is that the network topology file is augmented to include a logical bypass span in parallel with each chain subnetwork. If a given chain composition is (by nodes) A-B-C-D-E-F, with total mileage X , then the associated bypass span added to the topology model is a new span with end-nodes A-F and mileage X . If the original complete graph, such as [Figure 5-24\(a\)](#), has a set of spans S^0 and the corresponding meta-mesh graph, such as [Figure 5-24\(b\)](#), has a set of spans S^{mm} , then the augmented graph is $G(N, (S^0 \cup S^{mm}))$, recalling that sets never contain duplicate instances of the same members (in this case the direct spans). The logical bypass spans represent an equidistant routing option for working flows, which does not have the side effect of contributing to the loopback spare capacity requirement in the chain. This also means the express flow will follow the physical route of the chain (and the same fibers, cables, etc.) but is not handled by each OADM site en route of the chain. The express flows will go through simple splices or optical amplification, only being accessed by the OXC at the anchor nodes. The revised formulation will do nothing to explicitly force the solver to use the bypass spans but under global minimization of capacity, the solver will be enabled to reduce total cost by treating express flows in this separate way.

2. Representing the Dual-Failure Scenarios that Arise from Bypasses

Secondly, the mathematical model is extended to convert single physical cuts on spans of each chain into the corresponding *logical* dual-failure scenarios of failure of a physical chain span (between its immediate end-nodes) and simultaneous failure of the associated logical bypass span (between the anchor nodes of the corresponding chain). All other elements of the basic SCA or JCA design models remain as previously presented. To represent these simultaneous logical span failure scenarios, the set of spans (previously just S) is redefined as:

- S_d is the set of direct spans in the network.
- S_b is the set of logical bypass spans in the network.
- S_c is the set of chain spans ($= S \oplus (S_d \cup S_b)$).

With reference to these subsets of S , the spare capacity generating constraint, which is the same under SCA or JCA, (for example [Equation 5.28](#)) is restructured as follows:

Equation 5.56

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p} \quad \forall i, j \in S_d \times S | i \neq j$$

Equation 5.57

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p} + \sum_{p \in P_k} \delta_{k,j}^p \cdot f_{k,p} \quad \forall i, j \in S_c \times S | i \neq j, k = K(i)$$

[Equation 5.56](#) is identical in form to the ordinary spare capacity generating constraint mentioned above, but here it only applies in that form

to failures on direct spans. It ensures there is sufficient spare capacity over all other spans j to accommodate all restoration flow routed over the span for failure of any direct span i . Note "all other spans" in this context includes all direct, chain and logical bypass spans^[12]. This allows the solver to exploit the spare capacity that may be needed for local loopback in a chain under one failure scenario to also be used for restoration of a direct span elsewhere in the network. Equation 5.57 represents the logical dual-span failures that arise when a bypassed chain is cut. It requires that there be enough spare capacity on all other spans j to carry all restoration flows routed over them in response to the *simultaneous* failure of any chain span i and its associated bypass span k . The many-to-one mapping between individual chain spans and their associated logical bypass span k is provided by $K(i)$. For instance, if spans 7, 8, 9, 11, 12, comprise chain 6, then $K(i=7,8,9,11 \text{ or } 12) = 6$. Equation 5.57 also employs appropriately changed sets P_j and P_k of eligible routes for restoration of these combined logical failures. Generation of the revised eligible route sets is the third change needed to implement the meta-mesh designs.

^[12] Nothing in the meta-mesh concept prevents spare capacity being allocated to one of the logical bypass spans itself. Such capacity would be placed by the solver when the restoration requirements of the meta-mesh require more spare capacity on a chain than the chain otherwise has for its own loopback needs. If placed, such "extra" spare capacity is associated with the bypass, not the chain, to reduce termination costs through the chain.

3. Redefining Eligible Route Sets in the Augmented Topology

The following changes are made to the sets of eligible routes for restoration of each span failure:

- For direct spans, P_j is regenerated in the presence of the logical bypass spans now associated with the chain subnetworks. In other words, the entire augmented graph is represented for generating P_j for direct spans.
- For a physical span i that is part of a chain, its eligible route set is now enriched to include routes over the logical bypass spans of other chains, but excludes any routes that would cross its own associated bypass span, $k=K(i)$, since these fail together. In other words, the graph presented for generating P_i is $S_d \cup (S_b - \{k\}) \cup S_c$.
- New route sets are generated for the logical bypass spans themselves. For logical bypass chain k its eligible routes are generated within $S_d \cup S_b \cup (S_c - \{\text{all } i | k \text{ bypasses } i\})$. In other words, eligible restoration routes for a bypass span have to exclude routes over any of its associated chain spans, i.e., spans where $K(i) = k$, since these fail together.

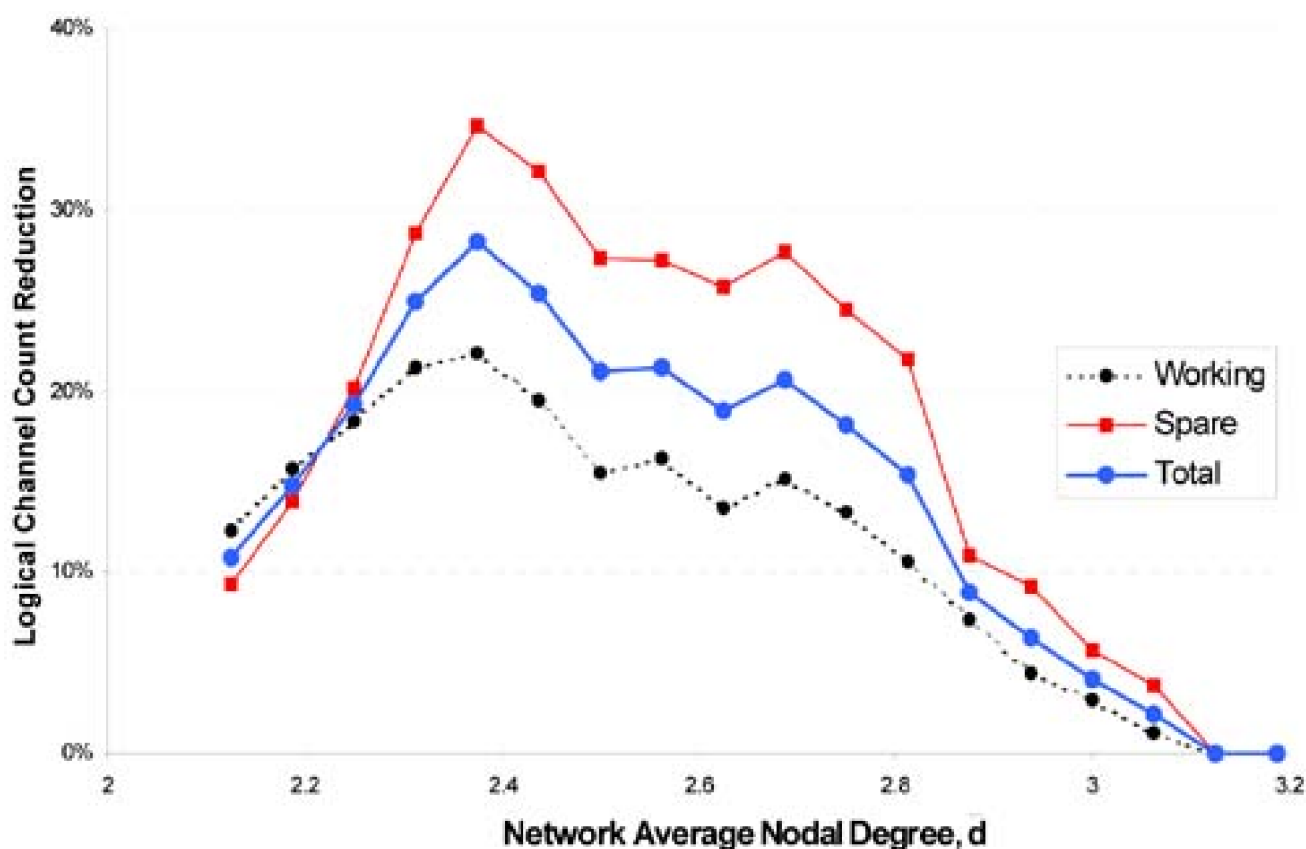
If the chain-optimized design is non-joint, then the demands are shortest-path routed through the augmented graph. The program that does this should, however, explicitly associate working paths going completely through a chain with the w_j quantity of the associated bypass, not the chain spans en route. A simple alternative to assure this outcome with an existing routing program is to slightly reduce the apparent length (or any other respective cost measure) of bypass spans relative to their corresponding chain's end-to-end length. This way whenever O-D relationships allow, a min-cost router will automatically prefer the bypass over the associated chain, but still be forced to route local flow into the chain explicitly over chain spans. On the other hand, if a jointly optimized design is desired (the meta-mesh equivalent to JCA), then the eligible routesets for working path assignment Q^r need to be regenerated to include routes utilizing the bypass spans. They must be regenerated within $G(N, (S^d \cup S^{mm}))$ to encompass a new view of all possible routes including the logical bypass spans now added.

5.8.6 Sample Results

As an example of this improved method for span-restorable design on sparse graphs, we summarize some results from [GrDo02]. Because nodal degree was the prime characteristic of networks against which we wished to observe the benefits, families of test networks are created by a systematic reduction in network degree from a highly connected master graph. Progressively sparser networks were derived by random removal of one span at a time from the master graph, subject to retaining biconnectivity. This provides a reasonably continuous variation of \bar{d} while keeping nodal positions and the end-to-end demand pattern common over all test networks. The same basic scheme for sweeping \bar{d} is used later to compare a variety of different mesh schemes in terms of how capacity typically varies with \bar{d} .

Figure 5-28 gives a sample result showing the breakdown of constituent savings in working, spare, and total logical channel counts of a joint meta-mesh design relative to JCA benchmark designs on test networks at each nodal degree. Figure 5-28, and the corresponding curves of total distance-weighted capacity savings of the meta-mesh designs (not shown), both show an interesting overall trend where (starting from a high \bar{d}) the relative savings first increase as \bar{d} is lowered but then drop off again toward the most extremely sparse cases. The reason is that the method produces its benefit where there are significant express flows through chains. The benefit or contrast against the conventional model is therefore greatest when there are long chains being traversed by significant express flows. In Figure 5-28 and other results these effects reach a joint maximum for the network family tested at $\bar{d} \approx 2.4$. In the vicinity of the peak there is a 10% to 15% reduction in total network capacity and up to 35% reduction in spare channel counts and ~22% in working channel counts. But as \bar{d} goes even farther toward the limit of sparseness, $\bar{d} = 2$, more of the nodes that were previously sources or sinks for express demands over chains are now also *part of* a chain themselves, converting their demands into local (intra-chain) demands. Thus, the continued reduction of \bar{d} reduces the number of express flow relationships which are opportunities for chain optimization. In the logical limit of a Hamiltonian cycle on all nodes, all demands are intra-chain and there can be no express flow related spare capacity savings in the sense pursued here at all. Hence curves such as in Figure 5-28 are all heading to zero in the limit of $\bar{d} = 2$. The relative benefit is also zero at the high \bar{d} range because those networks contain few, if any, chains and the conventional mesh design cannot be improved upon by chain bypass in a more richly connected topology.

Figure 5-28. Breakdown of logical count reduction for working, spare, and total capacity due to chain-bypassed designs relative to JCA reference designs.



An additional benefit to this method in optical networks is that only the meta-mesh nodes require a full OXC functionality. Chain node sites can go on using simpler OADM equipment. The logical bypass flows on chains are also an ideal application for a waveband pass-through feature on the OADMs. If the express flows are conveyed through chains via OADMs with passive waveband pass-through filters, chain span failure will propagate a loss of signal alarm to the optical cross-connects in the anchor nodes. This triggers an otherwise normal mesh restoration reaction that proceeds, for both express and looped back working capacity within the meta-mesh graph. Alternately the chain bypass express flow can also be supported on separate fibers or wavelength channels co-routed with the chain spans but only terminated at the anchor nodes.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

5.9 Span-Restorable Capacity Design with Multiple Service Classes

So far in this chapter we have always considered that upon a failure all affected working capacity is to be restored. This is the implication expressed by the generic "[restorability](#)" constraint that is present in each of the models above:

- restorability:

Equation 5.58

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

This assumption in transport was almost axiomatic in the SONET era. In fact it is the only paradigm that SONET ring-based transport supports: if a demand is routed through the working capacity of a ring, then it is 100% restorable against single-span failures, regardless of the priority of the services borne or the different revenues for services. But with the evolution toward IP data as the main traffic type it is more reasonable to consider a range of different service classes in terms of the protection guarantees they obtain. In some cases, lightpaths or OC-ns provisioned for use as inter-router IP trunks may not need to be protected in the transport layer if the IP service layer relies on routing reconvergence or MPLS backup path switching within the router-peer layer for fault recovery. On the other hand, some ISPs may prefer to refer survivability into the transport layer for faster response and/or simpler service layer administration and management. The relative ease and efficiency with which different survivability options can be supported on a per-path basis, all within the same capacity pool and using the same basic restoration procedure, is an important advantage of mesh networking in general over ring-based transport. In this section we will specifically consider how a span-restorable mesh can be designed to support multiple "quality of protection" (QoP) classes. It is easy to conceive of at least four different QoP classes for individual demands:

- **Gold** (assured restoration): these are working channels that must be restored against any single-span failure.
- **Silver** (best-efforts restoration): these are working channels that should be restored if possible within any unused spare capacity following 100% restoration of any gold class service capacity.
- **Bronze** (non-protected service): these are working channels that are ignored from a restoration viewpoint. They do not receive any restoration efforts but they are also not interrupted or preempted unless they themselves experience a failure.
- **Economy** (preemptible services): these are working channels that are not protected and moreover may be seized (interrupting their service paths) and logically viewed as spare capacity on spans not affected by a current failure if needed to satisfy gold requirements.

[Table 5-4](#) lists three composite multi-QoP schemes that can be further defined based on the four QoP classes. In Scheme 1, traffic is either protected (gold) or unprotected (bronze). In Scheme 2, a best-efforts class (silver) is added, and in Scheme 3, a preemptible traffic (economy) service class is offered at the same time as all three other service offerings.

Our interest in this section is how to adapt the single-QoP capacity design problems, such as SCA, JCA or MJCA, to take into account the differences and interactions between mixtures of simultaneous demand in the multiple QoP classes. [\[13\]](#).

^[13] While under development for the book, selected material from this section was adapted into the page [GrCI02](#). The models and results for this section were implemented in collaboration with M. Clouqueur.

Table 5-4. Three schemes of multiple QoP service classes

Scheme 1	Scheme 2	Scheme 3	service class designation
protected	protected	protected	gold (g)
—	best-efforts	best-efforts	silver (s)
non-protected	non-protected	non-protected	bronze (b)
—	—	preemptible	economy (e)

5.9.1 How Multi-QoP Span Restoration Works

Let us first explain, however, an operational framework that would go hand in hand with the multi-QoP capacity designs. This will show how relatively simple it is to support multi-QoP restoration in an SR network. Many implementations would be possible, but we describe one that we think is relatively simple and elegant. The key ideas are:

1. Each service path is tagged with the service class of the demand it bears.
2. OXCs self-monitor the breakdown of channels incident on them in each span by service class tag of the payload on them, and of any unused capacity. $w_i^g, w_i^s, w_i^b, w_i^e$ denote the working capacity in each class (respectively gold, silver, bronze, economy) in span i .
3. DPP trials (or centralized computation) identify (but do not activate) the maximal set of paths between custodial nodes formed in the (logically merged) pool of spare and economy channels on all other spans $\{rest_paths\}$.
4. Paths in $\{rest_paths\}$ are ranked in decreasing order of the number of economy channel hops in their makeups and secondarily ranked by length or hop count (i.e., the shortest paths made solely from spare capacity are at the top of the list. Paths containing the most economy channel hops are at the bottom.)
5. Upon failure the custodial nodes broadcast the failure span ID and the selection of path numbers from the preplan for that failure which they want formed over the other nodes. The selection is the topmost $min\{(w_i^g + w_i^s), |\{rest_paths\}|\}$ paths from the ranked $\{rest_paths\}$ set.

The tagging of demands as to their service class is easily accommodated in OC-n, digital wrapper, or GFP overheads, or conveyed by an optical service channel (OSC). The OXCs adjacent to each span would use these designations to classify working channels on the span. Being present in the overheads or OSC for the span, this information is self-updating at every OXC and does not need to be held in any network databases or disseminated by any protocols. It is inherently suitable for a dynamic demand environment. If the service class changes or the routing of a path is changed at any time, the custodial cross-connects for a span see the updated working capacity class composition essentially the instant the new path or status change is made. Thus, what has previously been a single w_i quantity for the working capacity to be protected on span i is now composed of up to four constituent types of working capacity ($w_i^g, w_i^s, w_i^b, w_i^e$) which the custodial cross-connects will treat as per the four basic service class definitions above.

Self-planning via DPP is again proposed to continually develop and revise fast protection preplans in each node. But in the multi-QoP context for DPP we use a merged view of spare capacity and economy class channels. The merging of the two types of capacity (as "equivalent to spare") during the DPP trials allows for the greatest efficiency of overall path-set formation for restoration. It remains desirable, however, to preempt as few economy class services as possible in actual restoration. This is why the custodial nodes rank members of $\{rest_paths\}$ as described so as to make the most use of true spare capacity before interrupting economy paths to access their capacity. The information for ranking by the custodial nodes is obtained directly as an outcome from the DRA: the statelets which wind up tracing each feasible path that has been identified normally bear a hop count indicator. For multi-QoP they would bear two hop count values: counting hops of spare and economy channels in their make-up. (Alternately, the custodial nodes can explicitly interrogate the hop structure of each preplanned path in terms of its spare and economy channel composition.) Note, as always in DPP, that these are *potential* restoration paths that have been identified. None of them are physically cross-connected at this point, but each participating node is "armed" by the DPP information it develops so that a simple notification from any failure end-node can crystallize formation of all the

corresponding paths by cross-connection in parallel as each node hears the activation flood^[14]. (The signaling for path activation is not done in series along each path itself as cross-connections are made.) In conducting DPP trials, the custodial node that launches the DRA to effect the discovery phase for feasible paths uniquely labels each individual path formed in the trial, and this number labels the path through all nodes that would be involved in its formation.

[14] In ordinary span restoration, cross-connection of the spare capacity is not possible before failure (even if preplans are deployed in every node) because each spare channel is shared over many possible failure scenarios that may use it in a different restoration path. Only when a failure occurs, it is then known which specific preplanned or on-demand cross-connections to activate regarding use of the given spare channel.

Upon an actual failure, the custodial nodes (one or both can do this) broadcast the failure span ID notification appended with a list of the specific path numbers they want activated from their DPP failure plan. The path selection list is added for the multi-QoP context so that actual activation of protection paths is conservative (i.e., based only on the actual w_i^g and w_i^s failure quantities that arise) and reflects the ranked view of $\{rest_paths\}$ that will minimize the impact on economy services. In contrast to normal DPP using only true spare capacity, it suffices for each other node to immediately make all spare-to-spare cross-connections that it has pre-armed for the given failure. If it is more than needed, no harm is done because these connections are only made between spare channels, which is not service-affecting. Any excess paths are released if not bearing a payload applied by the custodial nodes. Note that nothing is explicitly required to first "remove" economy class paths from the channels that are to be seized for gold or silver restoration. This happens simply by virtue of the signal substitutions effected at the custodial nodes to reroute the higher priority signals into the restoration paths and at the other nodes who, in activating the requested DPP preplans inherently also just break into the economy paths where needed, cross-connecting to actualize the needed restoration paths. If desired, however, any OXC breaking into an economy path is in a position to also apply a special "preempted notification" form of AIS signal into the surviving stubs of such economy paths.

5.9.2 Multi-QoP Capacity Design

A quite general span-restorable multi-QoP capacity design model can be developed using MJCA (Section 5.6) as a starting point. By addressing MJCA from a multi-QoP standpoint we implicitly also show how SCA or JCA models can also be made into multi-QoP variants. In MJCA all working paths are implicitly equivalent to what we have now defined as gold. In other words, the design intent is to give 100% restoration to all w_j and there is only one class of w_j . The changes to support Scheme 1 in Table 5-4 are the easiest to handle: one uses

MJCA as given but with the ordinary working capacity quantities of spans replaced by w_j^g in all but the modularity constraint. In effect this makes only "gold" capacity visible to the restoration process and related spare capacity allocation. Bronze capacity is essentially ignored—it is neither protected, nor is it cannibalized under any circumstances. The only place that bronze working capacity must appear would be in the modularity-generating constraint and in a working capacity-generating constraint equivalent to Equation 5.49, but pertaining only to bronze demand and generating a bronze working capacity. Because all physically present capacity needs to be considered in generating the module decisions, even if that capacity is not protected, Equation 5.52 becomes:

Equation 5.59

$$s_j + w_j^g + w_j^b \leq \sum_{m \in M} n_j^m \cdot z^m \quad \forall j \in S$$

In approaching the other mixtures of QoP classes, we can first make some observations about how the basic model (MJCA) would be altered to reflect the different treatments of working capacity. By this we mean, for instance, that in one constraint system preemptible capacity is equivalent to spare capacity, but not in others, and so on. Table 5-5 is provided to guide and summarize our discussion of these observations, providing a cookbook summary of considerations to develop any of the multi-QoP models in Table 5-4. In Table 5-5 we identify each constraint system in the MJCA model by the role it plays and with 1, 0, or -1 notations indicate how each class of working capacity is to be accounted for in the corresponding multi-QoP design model. For instance, the column for economy capacity in Table 5-5 shows that in any design problem with an "economy class" the corresponding span capacity is:

- a. Omitted from the restorability constraint (e.g., given a weight of 0 in in row 2 of Table 5-5).

- b. Treated as a credit against needed spare capacity in the spare capacity constraint (e.g., given a weight of -1 in row 3).
- c. Included in the modular capacity constraint (e.g., given a weight of 1 in row 4).

In this way, [Table 5-5](#) actually specifies many different multi-QoP design models all of which can be derived from MJCA for any subset of service classes that is present in a given design context. [Table 5-5](#) gives each constraint system in MJCA a name for reference, points to an example of the standard form for that constraint, and follows this with a summary of the sense (1, 0, -1) of how each new class of capacity interacts with the various constraint systems.

Table 5-5. How each class of service capacity interacts with the general minimum capacity design model for span-restorable networks

	Design Consideration	Example Constraint(s)	w_i^g	w_i^s	w_i^b	w_i^c	s_i	Comment
1	Routability and working capacity	Equation 5.48 and 5.49	Generated for each service class				N/A	Have separate demand matrix per QoP class
2	Restorability	Equation 5.50	1	0	0	0	N/A	Only gold is assured of restorability
3	Spare capacity	Equation 5.51	1	0	0	-1	Generated	Preemptible working capacity reduces spare
4	Modularity	Equation 5.52	1	1	1	1	1	All capacity contributes to modular sizing

In row 1 of [Table 5-5](#), we recognize that the routing of working paths here is essentially the same as in MJCA but that now we need an instance of the corresponding pair of constraints to generate the working routes and the respective $w_i^g, w_i^s, w_i^b, w_i^c$ class-resolved working capacity quantities on each span. In row 2, we record the reasoning that in any of the multi-QoP models restorability will only ever be explicitly asserted for w_i^g quantities. Note in this regard that the difference between best-efforts and strictly non-protected services is really only operational; there is no distinction between silver and bronze from a capacity-design standpoint. Neither has any restoration considerations been built into the design for them, but silver will receive operational best-efforts to exploit any available and otherwise unneeded spare capacity for its restoration under the given failure scenarios, whereas bronze does not receive this effort at all. Both types of capacity must be reflected in the total capacity modularization constraint, however. This reasoning also suggests that we can simply merge silver and bronze capacity requirements in the multi-QoP capacity design model.

In row 3 of [Table 5-5](#), we address the changes to the spare capacity generating constraint, indicating that restoration flows corresponding to gold class working flows need to be fully supported by the spare plus preemptible capacity dimensioning but that no explicit considerations of spare capacity are made for silver, bronze or economy class demands crossing the span. Moreover any economy class demands crossing a span represents *preemptible capacity that is completely interchangeable for spare capacity* on the corresponding spans. This is the meaning of the "-1" in Row 3 under w_i^c : any economy class capacity on the span can be considered as an equivalent credit against the explicit spare capacity variable on that span. The last row of [Table 5-5](#) simply recognizes that all forms of working capacity plus spare capacity must be supported under the final modular capacity placement decisions.

As a result of these considerations we can construct the multi-QoP design model corresponding to any of the three composite priority schemes in [Table 5-4](#), without having to completely detail each one. For instance, for Scheme 1, we include w_i^g in all constraints and w_i^b only in the modularity constraint. Scheme 2 is identical from a capacity design standpoint, except with the additional inclusion of w_i^s in the modularity. This leads us up to Scheme 3 where all four priority classes may be present together. To consolidate the points above, and for reference use, let us now make a complete summary statement of this most general multi-QoP model.

5.9.3 Multi-QoP MJCA

In the most general multi-QoP model we will have all four priority classes present, as well as joint optimization of working routes, spare capacity and modularity. All variables and parameters are the same as in MJCA (Section 5.6) with the exception that the prior set of demand requirements D are now represented by four constituent demand sets, one for each priority class. In practice, for the capacity design problem only, this can be reduced to three by merging the best-efforts and non-protected requirements, based on the considerations above. Thus we have the new subsets:

- $D^g, D^{s,b}, D^e$ are the sets of demand requirements for gold, silver and bronze (merged), and economy class services. Individual values of these sets are still referred to as d^r (integer), and index r continues for use in enumerating members of any of these sets.

The sets P_j and Q^r remain unchanged because the eligible restoration routes and eligible choices for working routes remain properties of the topology alone, unaffected by the structuring of demands into different service classes. The multi-QoP capacity-design model thus becomes:

Multi-QoP MJCA

Minimize

Equation 5.60

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m$$

subject to:

- routability (of all demands):

Equation 5.61

$$\sum_{q \in Q^r} g^{r,q} = d^r \quad \forall r \in \{D^g \cup D^{s,b} \cup D^e\}$$

- gold working capacity:

Equation 5.62

$$w_j^g = \sum_{r \in D^g} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S$$

- silver-bronze working capacity:

Equation 5.63

$$w_j^{s,b} = \sum_{r \in D^{s,b}} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S$$

- economy working capacity:

Equation 5.64

$$w_j^e = \sum_{r \in D^e} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S$$

- assured restorability for gold only:

Equation 5.65

$$\sum_{p \in P_i} f_i^p = w_i^g \quad \forall i \in S$$

- spare and preemptible capacity:

Equation 5.66

$$s_j \geq \sum_{p \in P_i} \delta_{ij}^p \cdot f_i^p - w_j^e \quad \forall i, j \in S^2 | i \neq j$$

- modular total capacity:

Equation 5.67

$$s_j + w_j^g + w_j^{s,b} + w_j^e \leq \sum_{m \in M} n_j^m \cdot z^m \quad \forall i \in S$$

Equation 5.68

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i ; \quad g^{r,q} \geq 0 \quad \forall r \in \{D^g \cup D^{s,b} \cup D^e\} \quad \forall q \in Q^r$$

Equation 5.69

$$s_j \geq 0 \quad w_j \geq 0 \quad \text{integer} \quad \forall j \in S ; \quad n_j^m \geq 0 \quad \text{integer} \quad \forall j \in S \quad \forall m \in M$$

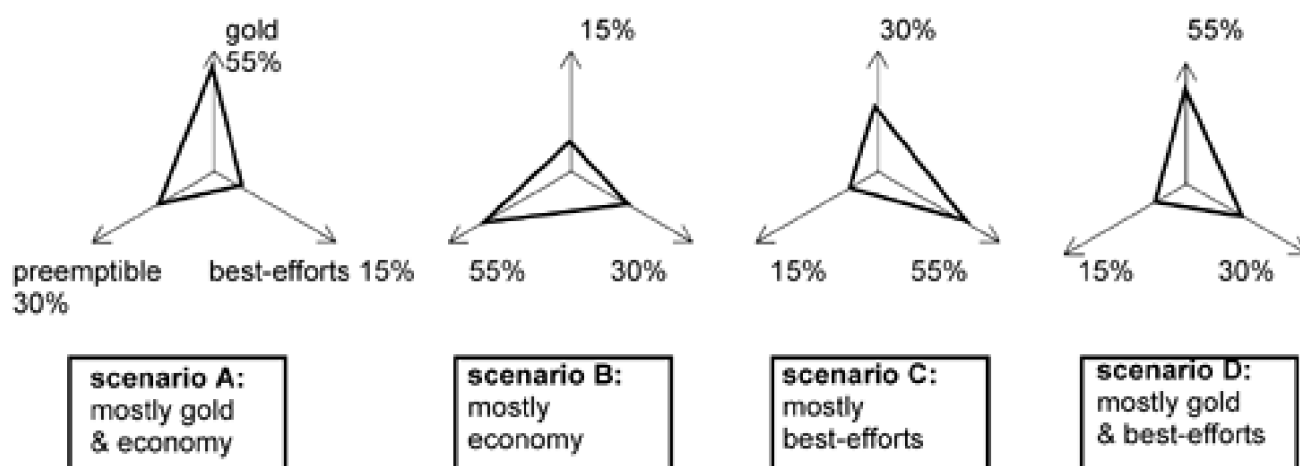
A non-modular multi-QoP model corresponding to JCA is obtained by dropping [Equation 5.67](#) and using the cost-weighted sum of all types of working and the spare capacity as the objective function. A multi-QoP model corresponding to SCA is further obtained by also dropping [Equation 5.61](#) through to [Equation 5.64](#) and generating working capacities through shortest-path routing or some other procedural routing method. Note that once the problem is reduced back to its SCA variant it is identical to an instance of conventional SCA for gold working capacity only, but with some effectively "existing" spare capacity quantities on spans representing preemptible working capacities.

5.9.4 Sample Results on Multi-QoP Designs

There are many performance and network strategy studies that can be conducted with an implementation of multi-QoP MJCA. One of the basic questions of interest is to use the formulation to characterize the levels of best-efforts restorability that are achievable under span-restoration for a given network and volume of gold class demand requirements. Ironically, if the best-efforts restorability levels are intrinsically high, it may be hard to sell customers on the premium price presumably associated with strict (gold) assurances of restorability. Another interesting question is the extent to which gold class services can obtain restoration at the expense of preemptible economy class services instead of through placement of conventional spare capacity.

To provide a sample of results related to multi-QoP design effects, we compared a conventional (single-priority) fully restorable joint but non-modular design to the corresponding designs having the four different profiles of multi-priority demand mix represented in [Figure 5-29](#). The test case demand requirements are constructed by assuming a total of 20 units of demand between each node-pair. For the single QoP benchmark design all 20 are gold class demands on each node pair. In the multi-QoP trials, 55% corresponds to 11 units, 30% is 6 units and 15% is 3 units to make up the four multi-priority scenarios. The results for a sample network of 20 nodes and 40 spans are summarized in [Table 5-6](#). The basic findings were similar for 50- and 60-span test networks. Note that for simplification of the discussion, bronze class capacity (which is without any effects or interactions other than to contribute its own mass into the ultimate span capacity totals) is omitted from these scenarios.

Figure 5-29. Four scenarios of mixed priority classes for studying multi-QoP designs.



The capacity columns are the distance weighted cost of unit capacity totals over all spans. In the conventional reference design, all 20 demand units on each O-D pair were implicitly gold class services. The result was a 100% restorable network with a working capacity cost of 9216 and spare capacity cost of 4943, or ~54% redundancy. When the mix of service types was changed to Scenario A (55% gold, 15%

silver and bronze, and 30% economy), the total cost of working capacity went *up* about 4%, but no actual spare capacity was required at all. Instead, the 55% of demands in gold class were able to entirely realize their restorability requirements through preemption of economy class paths. As a result total capacity dropped almost 48%. The reason the working path mileage rose slightly is a joint optimization effect. In the context where economy capacity can be preempted, gold working paths may deviate more significantly from shortest-path routes than in an ordinary joint design, to enhance the ability to exploit economy class capacities rather than build in spare capacity.

The remaining rows of [Table 5-6](#) can be read accordingly for the other multi-QoP demand scenarios. The most striking overall effect is that in all scenarios there was complete or extensive displacement of spare capacity in favor of preemption of economy capacity. Even in Scenario D, where there was the most gold capacity of any scenario (55%) and the least of economy (15%), the actual spare capacity needed was reduced over 90% (from 6095 to 589) with a related reduction in total capacity ~33% relative to the all-gold conventional design treatment.

Notably in Scenario D, the total working mileage rose the most as well (~7%). Investigation shows that this was almost completely attributable to deviations in the routes of gold and even moreso of economy working paths to create the synergies between gold and economy that allow the avoidance of conventional spare capacity. This is evident in [Table 5-6](#), row 4 (Scenario D) where it can be seen that the gold capacity-mileage total is slightly higher than the minimum of 6226 (the total for silver in Scenario C) whereas economy total mileage rises 42% above its total in Scenario C—this being the other case where economy was only 15% of demands.

In a separate inspection of the results it was found that any given economy class service paths could expect to be interrupted by preemption on about 16% of all single-span failures in Scenario A. The experience of individual economy class paths varied widely, however, from some paths that were almost never being preempted (~2%) to being preempted on ~46% of all network failures. Interestingly in the only other Scenario (D) with 55% gold and only 15% economy class demands, the likelihood of any one economy class path being preempted for gold class restoration needs was only slightly greater: (min = 2.2%, average = 18%, max = 41%). This may be attributable to the placement of explicit spare capacity in Scenario D designs whereas Scenario A designs rely solely on preemption to achieve gold restorability. Related to this, a further inspection showed that on average 75% of the capacity needed for gold class restoration in Scenario D was obtained from economy preemption and the rest was based on conventional spare capacity.

Table 5-6. Test-case results comparing multi-priority span-restorable designs

Demand Scenario (g, b-s, e)%	Total working	Total gold	Total silver	Total economy	Total actual spare	Total capacity
Conventional	11890.80	11890.80	0.00	0.00	6095.33	17986.12
A (55, 15, 30)	11896.37	6450.85	1698.26	3747.26	0.00	11896.37
B (15, 30, 55)	11321.75	1698.26	3396.52	6226.96	0.00	11321.75
C (30, 55, 15)	11708.95	3538.85	6226.96	1943.14	0.00	11708.95
D (55, 30, 15)	12701.22	6540.95	3396.98	2763.28	588.97	13290.19

The level of best-efforts restorability that silver class services would receive was more highly sensitive to the demand scenario. The best-efforts restorability for silver class demands crossing failed spans follows restoration of all gold class capacity on the same span. As explained, the capacity design itself has no direct concern with the restorability of this class of service. Any restorability that does derive for such best-efforts services is therefore purely due to side-effects of the optimal design for gold class requirements. (Technically these are also forcer-related effects: the only reason there is any non-zero best-efforts restorability to be had under an optimal design is that forcers generate an environment of spare capacity that exceeds the needs of some other span failures.) The results for best-efforts restorability were therefore obtained by experimentally testing the resulting designs for the maximum feasible flow through the spare capacity of the surviving graph between the end-points of each span failure. The gold class restoration flow requirement is first subtracted from the total feasible flow and the remainder is assigned to the actual best-efforts (silver class) working capacity on the respective span.

Under Scenario B, with only 15% gold demands and 55% economy, silver class restorability is as high as 97% on average with a minimum of 57% on any one failure. This is so high it may be problematic from a business perspective in that the best-efforts service is essentially as good as the more expensive assured gold class services. In Scenario A, however, with 30% economy, silver managed a 37% average restorability with individual failure restoration levels varying from nil to 100% recovery. However, when the percentage of economy demands drops from 30% to 15%, in Scenario D, then the silver class restorability also drops to an average of 24%, with a range on individual failures from nil to 100%. Interestingly, however, when silver (as well as gold) are both allowed to preempt economy, the average frequency of preemption of economy paths rose only another 2 or 3% on average in both Scenarios A and D.

5.9.5 Approximation Models for Multi-QoP Design

One of the most interesting aspects of mesh network design under a multi-QoP context is the interaction between economy class service routing and the gold and silver classes which can preempt them for restoration. The interaction between gold and economy is conceptually like a mesh counterpart to the practice of putting "extra traffic" on the protection fiber of a SONET ring. In this analogy we can almost think of preemptible services as a kind of opportunistic "extra traffic" over what is otherwise the spare capacity of a span-restorable mesh designed only for the gold class services. We say "almost" because more precisely in a jointly optimized design, the placement of preemptible capacity may differ from the distribution and amount of ordinary spare capacity in an all-gold class design. But, to a first approximation, thinking of the preemptible services as only extra traffic on the spare capacity of an ordinary mesh design suggests a useful simplification to approximate multi-QoP designs, at two different extremes:

- *If the fraction of all demand that is preemptible is relatively small* one would initially ignore the preemptible service demands, generate the routing and spare capacity to serve (and protect) gold demands only, then seek routings of the economy class service paths through the spare capacity of the gold class restorable design. If there are economy class demands that are unserved within the initial spare capacity then the overall design model that we developed could be used, or extra (preemptible) channels turned up on minimum incremental cost paths for the additional economy class services. By minimum incremental cost paths we mean taking routes that require the least addition of new capacity to complete the routing.

Or conversely:

- *If the fraction of all demand that is preemptible is relatively high* one would initially ignore the gold class service requirements and generate a w_i^e capacitation of spans based on shortest-path routing only of the economy class services. The capacity design and routing for gold class service paths then follows but with a modified viewpoint in which all already placed w_i^e capacity is equivalent to existing spare capacity in the graph. This can be done by applying the conventional JCA model to solve for the spare capacity to protect gold class demands only with the simple addition of lower bounds on the logical spare capacity on each span to represent the preemptible working capacity already present on those spans. If little or no actual spare capacity is added by the JCA solution, it is an indication that a fully restorable routing of gold class working demand requirements is feasible using *only* preemptible working capacity and that the approximation is closely equivalent to a true multi-QoP design. If the total spare capacity added by the instance of JCA with lower bounds is judged more significant, then it is an indication that it may be preferable to use the full multi-QoP design model.

5.9.6 Maximum Revenue Multi-QoP Design

In the preceding multi-QoP considerations, the overall viewpoint is one of trying to serve all demands with a minimum total cost for the required capacity. But in a multi-QoP demand environment a related problem is that of determining a policy that maximizes net revenue assuming a currently installed base of capacity. In the single-priority case, net revenue maximization corresponds directly to capacity cost minimization because the total revenue from serving all the available demand is a constant. But under a multi-QoP context each service class not only earns different per-unit revenues but also has different resource requirements. For example a gold class service path will consume its own direct routing resources and will also contribute to the network-wide requirement for spare capacity. On the other hand, a preemptible demand can be thought of as effectively consuming *less* than its own shortest-path route cost by virtue of its contribution to the network spare capacity requirement of other services. The viewpoint is that an economy path is in general partly built over what we would otherwise have called the spare capacity of the network.

The following model assumes the same four service classes and constituent demand matrices as above. For convenience, however, in indexing demand pairs over the set of priority classes we will define the set A , indexed by r over all node pairs. An individual demand

quantity of a specific class on a specific node pair will then be denoted as a d_r^k quantity where $k \in (g, s, b, e)$ gives the service class. Other new parameters and variables that we need are:

- E^g, E^s, E^b, E^e are the earnings per unit demand per unit distance served in gold, silver, bronze and economy class services, respectively.

- L_r = the shortest path distance (or more generally the charging distance employed) for services provided on relation r (regardless of actual routing distance in the solution). If the earnings model is flat-rate (i.e., distance independent) then each O-D pair can be assigned $L_r=1$ and E^g, E^s, E^b, E^e can be considered earnings per unit capacity provided per service class regardless of distance.
- T_j is the total physically installed and available capacity on sparj.
- $n_r^g, n_r^s, n_r^b, n_r^e$ are the integer decision variables as to the actual number of offered demand units to serve in each service class on each relation.

The multi-QoP earnings maximization capacity-design model is then expressed as:

Multi-QoP Revenue

Maximize

Equation 5.70

$$\sum_{k \in \{g, s, b, e\}} \sum_{r \in A} n_r^k \cdot E^k \cdot L_r$$

subject to:

- routability of *selected* demands:

Equation 5.71

$$\sum_{q \in Q} g^{r, q, k} = n_r^k \quad \forall r \in A \quad \forall k \in \{g, s, b, e\}$$

- working capacity in each priority class:

Equation 5.72

$$w_j^k = \sum_{r \in A} \sum_{q \in Q} \zeta_j^{r, q} \cdot g^{r, q, k} \quad \forall j \in S \quad \forall k \in \{g, s, b, e\}$$

- assured restorability for gold:

Equation 5.73

$$\sum_{p \in P_i} f_i^p = w_i^g \quad \forall i \in S$$

- spare and preemptible capacity:

Equation 5.74

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p - w_j^e \quad \forall i, j \in S^2 | i \neq j$$

- existing span capacities:

Equation 5.75

$$s_j + w_j^g + w_j^s + w_j^b + w_j^e \leq T_j \quad \forall j \in S$$

- offered demand limits:

Equation 5.76

$$n_r^k \leq d_r^k \quad \forall r \in A \quad \forall k \in \{g, s, b, e\}$$

Equation 5.77

$$f_i^p \geq 0 \quad \forall i \in S \quad \forall p \in P_i; \quad g^{r,q,k} \geq 0 \quad \forall r \in A \quad \forall q \in Q^r \quad \forall k \in \{g, s, b, e\}$$

Equation 5.78

$$s_j \geq 0 \text{ integer} \quad \forall j \in S \quad ; \quad n_r^k \geq 0 \text{ integer} \quad \forall k \in \{g, s, b, e\} \quad \forall r \in D$$

The model maximizes the total of earnings for all demands served in each class. Limits on the total offered demand in each class apply. The model will choose which specific demands on each node pair to serve to maximize revenue from the available capacity, taking into account interactions between gold and economy on a detailed basis. Whereas silver or bronze demand basically only need to provide more earnings than their direct capacity costs to be included, gold decisions interact with economy. In one case a gold path may derive restorability solely from economy preemption, but another may require explicit spare capacity additions to the design. These are all network-dependent effects. If admitted together in certain selections gold earnings will not be at the cost of added spare capacity, but be

associated with economy capacity which at least has some earnings.

Several simplifying heuristics suggest themselves in approaching this problem. In practice one would expect that E^g could be significantly higher than all other rates. If, based on results above, we further assume that almost all gold capacity could be protected by economy capacity then a first heuristic would be "admit all the gold offered demand possible" up to the point where unused capacity does not suffice to protect it. This is a mesh maximum loading problem in which the gold demands are chosen from so that their individual total earnings is maximized subject to feasibility of restoration of the selected demands in the remaining unused capacity. Reasonably assuming $E^g > E^s > E^b > E^e > 0$ the next step in the heuristic approach would be to maximally load the unused capacity (i.e., the "spare" capacity that is protecting the gold loading pattern) from the economy services demand pool. This is followed in sequence by maximum loading from the silver demand pool in what remains, followed by any further loading possible from the bronze class demand pool.

The higher-level problem of setting the prices E^g, E^s, E^b, E^e themselves to maximize earnings within finite capacity is not directly addressed by this model but the multi-QoP revenue model can be a tool in trying to evolve such pricing policies. At that problem level, elasticity of demand (demand offered versus price) has to become a consideration as well. While nonlinear models may describe the elasticity functions, the effect that a proposed change in prices has can be assessed with the multi-QoP revenue model using the new E^g, E^s, E^b, E^e values and using the elasticity considerations to update the assumed amounts of offered demand in each service class, i.e., $n_r^g, n_r^s, n_r^b, n_r^e$. In this way an implementation of multi-QoP revenue can be used to study how the costs assigned to each service influence the maximum achievable revenue.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

5.10 Incremental Capacity Planning for Span-Restorable Networks

The models for capacity design that we have considered to this point treat the so-called "green fields" situation in that they solve for a complete capacity design to meet the demand matrix in its entirety. No transmission capacity is assumed to already exist. There are many applications for such models, especially in planning studies and fundamental studies comparing basic architectures, technologies or strategies. As the planning horizon shortens, however, the problem becomes one of *incremental* capacity planning. In this context there is an existing network as a starting point. It has already-installed and partly-filled transmission modules, live working signal paths, and a pre-existing arrangement of spare capacity and protection routes. An updated forecast of demand (or a set of exact new requirements) is provided for the coming period and the problem is to deploy additional capacity in the most cost-effective way to augment the existing network to serve this next pattern of growth and/or change in demand.

In serving the next growth increment, the intent is to take advantage of any slack in the existing capacity deployment before adding new capacity. A policy decision related to incremental growth planning is also whether or not the routing of any in-service paths can be altered to accommodate the growth. The traditional view is that this would be unacceptable because existing customers would experience a "hit" on their service. A countervailing view that is probably more viable under the data-centric orientation is that permission to rearrange (with customer notice and coordination) could simply be the basis for a reduced service price. In other words, similar to a multi-QoP orientation, there could be a "rearrangeable service class." The value to the network operator of permission to rearrange existing services in conjunction with incremental growth planning can be quantified by studies using the model that follows.

The following approach to incremental capacity planning needs to be carefully distinguished from the problem of multi-period planning for a series of future demand forecasts. In the latter, the aim is typically to decide upon the timing and magnitude of a series of future capacity placements such that total cost (or present worth of future costs) is minimized, while all forecast demand is served at each epoch. The problem we consider is simpler than this in that only one interval is considered at a time. In addition, with repeated application of an incremental growth model, the largest exposure to deviation of the actual growth from the forecast is only one planning period, whereas in a single multi-period optimization problem the solution can depart from the actual demand that arises over the entire multi-period interval. In other words, the multi-period problem depends on each period forecast being individually accurate whereas a true multi-period optimization model assumes the final, and all intermediate demand forecasts are all equally accurate. In contrast, repeated application of the incremental planning model takes into account the differences between forecast and actual *each time* it builds for the next epoch. It is thus much less able to have a severe departure by the end of several periods from the actual multi-period outcome compared to solution of a single multi-period optimization problem that assumes exact forecasts all the way out to the final period.

In contrast to multi-period optimization, we also have an existing network, a current actual demand pattern served by that network, and a new forecast to satisfy. The current state of deployed network capacity may have arisen from a prior forecast, but it is actually irrelevant what the basis of the current network was when the incremental problem is solved. The network may equally have been bought or inherited as is. What matters is only its current state and the next forecast (or set of incremental demand requests) to be satisfied. If we want to assert non-rearrangeability the current state information needs to include the current *actual* demand pattern served so that changes to meet the next forecast of demand can be addressed without implying rearrangements to existing demands in the network. Let us therefore denote:

- D^0 is the current *actual* demand matrix, indexed by r , with non-negative integer values d_{0r} .
- D is the change in the demand matrix in the next period, indexed by r , with plus or minus integer values h^r i.e., $D^1 = D^0 + D$ where D^1 is the complete demand matrix (or forecast) for the next period. No value of D^1 can be negative, however.

While existing demands that will be continuing through the next period are not rearranged, anywhere that the change value, h^r , is negative represents a release of working capacity between certain node pairs. This capacity will not be physically removed from the network but becomes available for "free" use as either working or spare capacity in the next step. Before running *i*-MJCA for some *m*th planning step, a pre-processing program traces the routes in question to identify the total of such released existing capacity on each span as input to the next planning step. The model thus assumes that the current as-built and as-used network state is characterized for the next planning step by:

- M = The set of modular capacity sizes, indexed m .

- Z^m = The number of capacity units in the m^{th} module size.
- $n_j^{0,m}$ = The number of modules of the m^{th} size existing on span j .
- w_j^0 = The number of *continuing* working paths routed over span j .
- u_j = The number of released working channels from the prior period (non-negative). More specifically, for each $r < 0$ and allowing that the total demand on relation r may be realized over more than one working route then:

Equation 5.79

$$u_j = \sum_{j \in S} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot \eta^{r,q} \quad \text{where} \quad \eta^r = \sum_{q \in Q^r} \eta^{r,q}$$

and $\eta^{r,q}$ is the number of released paths that were on the q^{th} eligible route for relation r .

The decision variables are:

- $n_j^{1,m}$ is the number of modules of the m^{th} size to *add* on span j in the current incremental design.
- $g_1^{r,q}$ is the amount of new working flow allocated to the q^{th} eligible working route for relation r .
- w_j^1 is the number of working channels *added* to span j in the incremental design.
- f_i^p is the revised total amount of restoration flow allocated to the p^{th} eligible route for restoration of span i .
- s_j^1 is the revised *total* number of spare capacity units allocated to span j .

The incremental MJCA (*i*-MJCA) design model is then:

i-MJCA

Minimize

Equation 5.80

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^{1,m}$$

subject to:

- routability of all *new* demands:

Equation 5.81

$$\sum_{q \in Q'} g_i^{r,q} = \eta^r \quad \forall r \in \Delta | \eta^r > 0$$

- new working channels required:

Equation 5.82

$$w_j^1 = \sum_{r \in \Delta} \sum_{q \in Q'} \zeta_j^{r,q} \cdot g_i^{r,q} \quad \forall j \in S$$

- restorability of *all* working capacity:

Equation 5.83

$$\sum_{p \in P_i} f_{i,p} = w_i^0 + w_i^1 \quad \forall i \in S$$

- globally revised spare capacity:

Equation 5.84

$$s_j^1 \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p} \quad \forall i,j \in S^2 | i \neq j$$

- modular total capacity:

Equation 5.85

$$s_j^1 + w_j^0 + w_j^1 - u_j \leq \sum_{m \in M} (n_j^{0,m} + n_j^{1,m}) \cdot z^m \quad \forall j \in S$$

- required span capacity cannot be negative:

Equation 5.86

$$(s_j^1 + w_j^0 + w_j^1 - u_j) \geq 0 \quad \forall j \in S$$

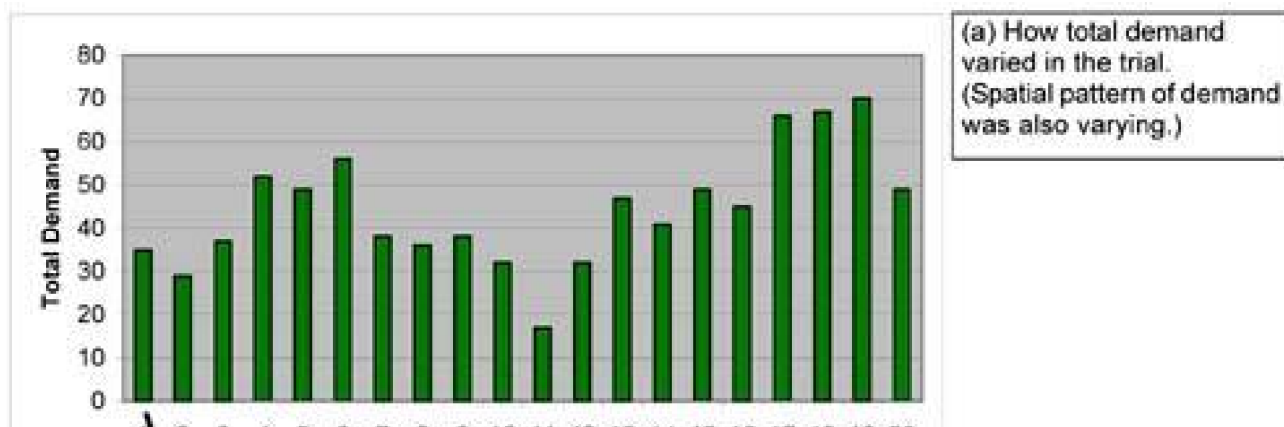
- plus all flow and capacity variables are non-negative, integer.

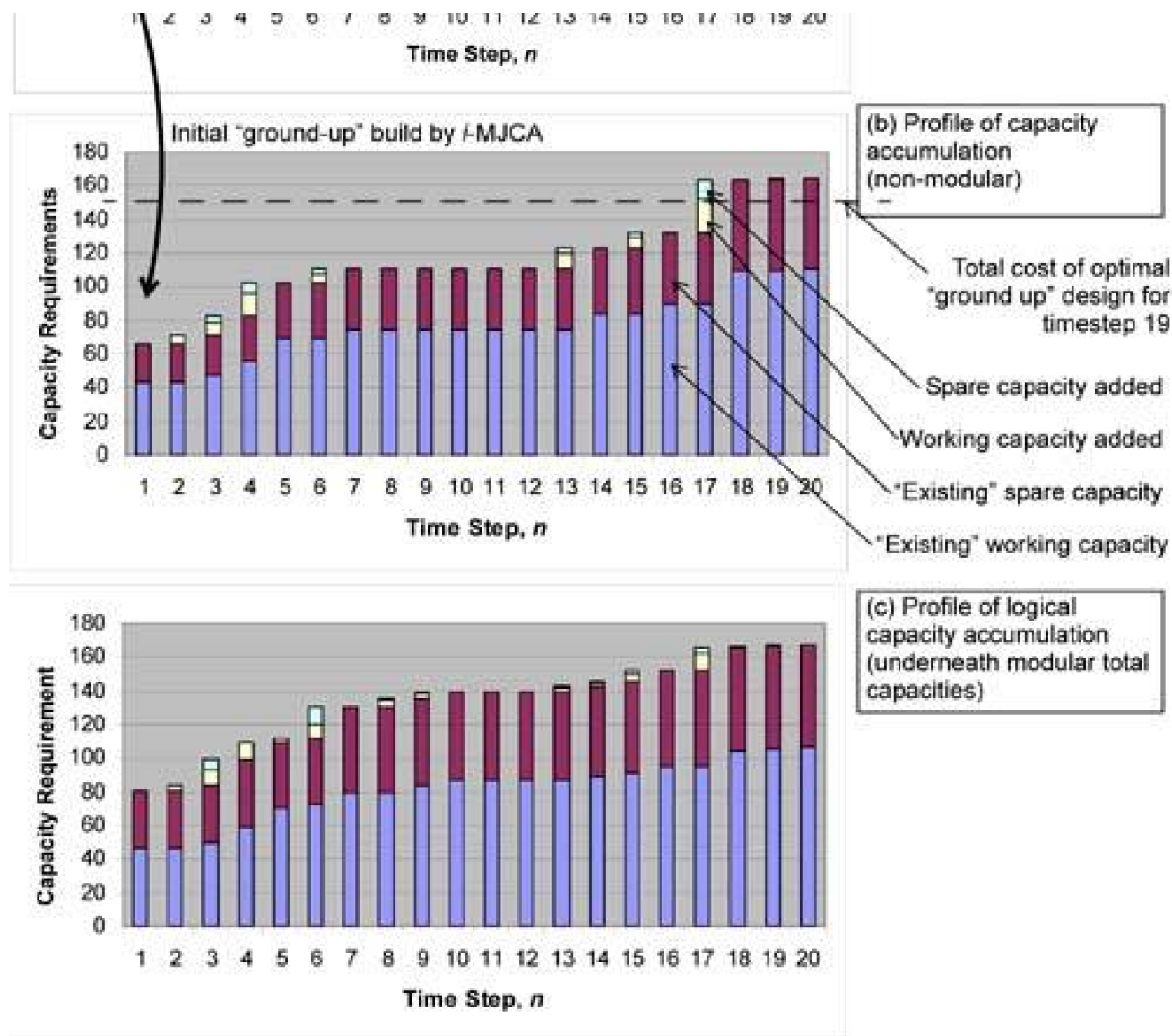
The last consideration ([Equation 5.86](#)) is required so that the "credit" from released capacity is cannot allowed what is logically required in the next step on any span. This could otherwise happen and drive the quantity negative—if, say, a large release occurs on a span whose actual capacity requirement in the next period is small. The salient properties of this model are that:

1. Existing demands are untouched (i.e., no rearrangement of existing working demands occurs as part of the routing and capacity design for the next interval).
2. Revised "ground up" decisions about the amount and use of *spare* capacity (and restoration flow assignments) at each step time. Changes in spare capacity allocations and associated rerouting plans can be globally re-optimized each time to suit the new composite situation because such changes in restoration considerations are entirely in the network background with no effect on working services. Note in this regard that the model implicitly treats any capacity in a module above and beyond working requirements as equivalent to spare capacity.
3. Point 2 further implies that existing spare capacity (including any simply unused module capacity) can be converted to working capacity if needed. The converse does not apply to used working capacity (satisfying the non-rearrangeability requirement) but does apply to released working capacity from the prior step.
4. New capacity can only be added in the form of whole modules.
5. Unused capacity present in existing modules can be allocated to serve growth *or* to become logical spare capacity for restoration.

A question of interest is the extent to which the result of a succession of incremental capacity planning steps approximates the ground-up optimal design if we had perfect knowledge of the actual demand pattern at any given step. Does successive incremental planning approximate the optimal ground-up design or does it diverge significantly with time? [Figure 5-30](#) is a sample result for a small test network (5 nodes, 9 spans) obtained from experiments with the *i*-MJCA model that partly address this question. [Figure 5-30\(a\)](#) shows the initial demand total in time step 1 and the variation afterwards in this experiment. Not shown is the detailed pattern of variation on a point-to-point demand basis, but which was significant—every node pair exchanged 2 Erlangs of lightpath demand with an average holding time of 1.5 time steps. So there was a considerable amount of "churn" in the test cases. The particular time-series captured is one where demands are random in detail as described, but happen to exhibit a slight trend of overall growth between the start and finish times and a noticeable "downturn" in the middle sections—perhaps like an economic cycle. This lets us observe how *i*-MJCA handles changing demand patterns within already-placed capacity at each step and also gives a sample case to see how different the final result of the incremental series of growth steps is from a single optimal design for the step of largest total demand near the end of the series.

Figure 5-30. Successive incremental planning experiment with *i*-MJCA—coasting through the lulls and accumulating total capacity that is not far off from the "ground-up" design.





In [Figure 5-30\(b\)](#) the incremental evolution of the total capacity is shown in the case where modularity is one, i.e., modularity is effectively turned off in the model so we can observe pure effect of how integer working and spare capacity requirements evolve. Here we see how *i*-MJCA is equivalent to "ground-up" MJCA in the first time step. When there is no existing capacity, it generates the initial ground-up design. At step 2 something interesting happens—total demand goes *down*, but a bit of new working capacity is *added*. This means that the demand pattern changed enough that released working was not enough—or not in the right places—to support a routing solution for the new demand pattern, and so new capacity was added. Then at steps 3, 4, and 6, additions of both working and spare channels are made to accommodate the overall growth trend that occurs from period 2 through to 6. From step 6 through to 17 there is no net growth of total demand—a "downturn" occurs in overall demand. Here we see that in effect a stable *protected working capacity envelope* has emerged (by step 7) within which all the changes in demand are accommodated (and protected) with no new capacity again until a major amount of net growth occurs again at step 17. Then working and spare are both "topped up" and run flat again to the end of the experiment. At the end, the total deployed capacity is ~162 (arbitrary units). A separate solution of MJCA gives the ground-up optimal capacity total for the largest total demand period (step 19) as ~150. So the incremental (unit-capacity) planning series has gone 20 steps and departed less than 10% from the optimal design for the largest demand matrix in the time-series. Averaged over all time-steps, the optimal ground up design is 23% better than the incremental design at each step. But this is impressively small considering that during all of the "downturn" periods, the incremental design retains all its deployed capacity while the optimal reference can deploy much less—only that needed for the current demand, not prior historical demand peaks.

[Figure 5-30\(c\)](#) shows the same experiments where real modularity is effected. Capacities of 12, 24, 48, and 96 were available at 2x an economy of scale. (Note, however, that the plot still shows logical working and spare channels employed at each stage, not exact modular capacity deployed.) Here we see more total capacity being deployed because under economy of scale that corresponds to cost optimality. More capacity is being used for both working and spare because of the deviation of routes to exploit economy of scale in module placements. The corresponding plot of total cost (and total modular capacity)—not shown, is as expected, much more stepped. From period 6 through to 14 there is actually no new modular capacity deployed at all. (As mentioned, the rises in [Figure 5-30\(c\)](#) are showing only the evolution of the actually *needed* working and spare channels underneath the earlier deployed modular capacities.) In the modular case, the final network (step 20) cost 1700 units, whereas the highest optimal ground-up modular design in the series cost 1200 units. This

more significant deviation of ~40% is attributed to the amplification of the overbuilding effect incremental steps may have with modularity. That is to say that a module may be needed and triggered at one step, but then with constantly changing demand patterns it is not as likely that this module can be as well employed later (as say a single added channel could be), or the module placed will actually be too small and ultimately cost inefficient in the longer run. The optimal modular case sees the ultimate full requirements which can let it precisely exploit economy of scale effects.

In fact this shows the double-edged nature of modular incremental planning: a small time step helps, in principle, to track demand closely. But on the other hand, to effectively employ large modules (with strong economy of scale), one needs a longer planning view. In a situation where economy of scale in modular capacities is significant, but the growth magnitudes in each interval tend to be small relative to the larger modules, the long-run benefit of placing large modules may be missed, and ultimately inefficient module sizing decisions made. In contrast a global multi-period optimization would recognize this opportunity, but requires a perfect long-range forecast as well. The problem in these circumstances would be that not a long enough view is being taken at each stage to exploit economy-of-scale opportunities that may pay off over the longer run, but not in the immediate planning period. Thus, there is an inescapable dilemma really—what planning interval to adopt? A planning interval that is too short (in terms of the typical amount of growth it absorbs) may result in, say, two successive placements of OC-48 module additions. But a longer planning interval may result in a single placement of a more economic OC-192 in the first instance, leaving even more slack capacity for future periods as well. This illustrates the important linkage between the use of ground up capacity planning to a long-range forecast and the use of shorter-range incremental growth models. The basic issue is, however, not just a limitation of this model but a fundamental one in network planning and business in general: should we optimize for gain in the next immediate period or invest more now for long-range efficiencies? Since the balance between these two goals is fundamentally a philosophical problem it should not be expected that any one O.R. model can encompass both views simultaneously. Instead, we have "ground-up" models for the longer-range view that can realize the eventual efficiencies of the large module investments, and the incremental model that can minimize costs in the short run while meeting operational requirements. One can easily imagine that in the state of the communications industry as this book is written, the incremental model would be strongly in favor. The combination of incremental planning, checked against and informed by the outcomes of separate long-range design studies, is the key to navigating well over both time scales.

5.10.1 What Value for Rearrangeability?

The value of obtaining permission to rearrange existing working paths in incremental capacity planning can be assessed by comparing the routing details in solutions of MJCA to incremental MJCA designs. The MJCA solution is obtained for the total demand matrix as forecast to exist at the end of the next interval, i.e., D^1 . The MJCA design is inherently a ground-up design recognizing no existing capacity or working demands already routed. The i -MJCA solution will specify the additional capacity modules and routes to implement on top of what exists at the entry to the current planning period. The only difference between the MJCA and i -MJCA design solutions at that stage is that the i -MJCA outcome reflects whatever succession of prior incremental builds and routing commitments were made, plus the current incremental requirements, whereas the MJCA solution at the same point represents what would be built, and the routings employed if the entire network were to be rebuilt from scratch at that point. Comparative analysis of the two complete designs can give an indication of:

- Shifts in the routing of existing working flows that would yield greater economy-of-scale benefits. This is indicated if the i -MJCA design shows many smaller capacity modules, on more spans, than the MJCA design.
- Shifts in the routing of existing working flows that would yield inherently greater spare capacity efficiency. This is indicated if the MJCA design has a lower logical redundancy (i.e., ratio of spare to working channel counts) than the i -MJCA accumulation design.
- How well the succession of incremental designs on the chosen period for incremental capacity planning and augmentation is approximating a continuous optimum multi-period solution, had the perfect forecasts been available. Poor tracking of the optimal growth trajectory is indicated if the MJCA design cost is significantly lower than the corresponding cost function applied to evaluate the accumulated as-built incremental design. If the cost differences are due to use of smaller modules than MJCA would use, then the incremental design period may be too short. A series of short-sighted designs is failing to exploit economy of scale. On the other hand, if costs differ significantly, but similar module choices are employed, the incremental period may be too long; forecast error is becoming a major factor. An efficient network is being built by a suitably long-range incremental planning process, but it is being built for a demand pattern that is too different from the actual by the time it is built.

An important property of restorable mesh networks in regards to incremental planning is that the spare capacity plan can be re-optimized at any time. Thus, in one sense there is never any efficiency loss due to spare capacity allocation suffering from incremental mismatch to the demand because there is no impediment to globally re-optimizing the spare capacity to suit the existing actual w_j accumulations at

each period. It is only through the accumulation of "locked in" working routes that the global efficiency may suffer. Any spare or unused channel can be designated to serve as spare, working, or unused in the next period. Without rearrangeability, however, once a channel is designated working, it cannot be reassigned to any other role in the next period.

Some preliminary studies in the author's research group suggest that without significant economy-of-scale effects, integer capacity incremental design accumulations exhibit few working routes that differ from the route for the corresponding O-D pair in a ground-up MJCA design. By itself this would suggest that rearrangeability is not important for incremental planning to approximate a global optimum. This is largely due to it being such a strong principle for working paths to take shortest routes, whether in a design that solved from the ground up (in MJCA) or incrementally built up (via *i*-MJCA).

The open issue is how an environment of significant modularity and economy of scale can alter this outlook, especially in sparse graphs. With strong economy of scale there could still be significant value in being able to rearrange existing paths, to aggregate new and existing flows into larger capacity modules. An early choice of working route might be quite different from the route that is later part of an optimum design that exploits economy of scale by deviating routes to aggregate flows for the employment of large modules (and also to obtain needed spare capacity efficiently from other large module placements). Strong modularity tends to want to attract or "suck up" flows from surrounding routes onto themselves as part of a cost-optimal design. It is in these circumstances where rearrangeability could permit the adoption of new routes for existing flows to take advantage of the optimum large-module design solutions.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

5.11 Bicriteria Design Methods for Span-Restorable Mesh Networks

In this section we show how the bicriteria optimization technique described in [Section 4.16.4](#) can be applied to enhance certain secondary characteristics that we may desire in an SR mesh network design. Combinatorial optimization problems may often have multiple solutions that are equivalent, or nearly equivalent, in terms of objective function values, but which differ in detailed construction. Selecting for preferred characteristics among these different detailed constructions, or trading off a certain amount of the primary goal to enhance the second are the key ideas involved. Examples of such secondary characteristics of interest are:

- The lengths of restoration paths.
- The extent to which working channels can be "rolled to protection" on the spare capacity of the same span to support maintenance actions.
- The restorability level of a "best-efforts" service class.

For each of these, and other secondary attributes of interest, there are two generic questions that apply:

1. Might there be multiple cost-equivalent optimal solutions, some of which would happen to be superior over others in the secondary characteristic? How could we get a design solver to choose one of the secondarily preferred designs rather than an essentially random member of the set of strictly cost-optimal solutions?
2. Is it possible that a relatively small additional expenditure might be able to greatly improve the secondary characteristic, perhaps even to specifiable targets for the secondary criteria?

The bicriteria method allows us to address both these questions for the three characteristics mentioned. The idea from Question 1 will be to lightly bias the solver to choose a preferred configuration for us among strictly cost-equivalent solutions to the main problem. The same technique with heavier bias will let us explore the trade-off, implied in Question 2, between added cost and improved secondary characteristics.

5.11.1 Bicriteria Design for Reducing Restoration Path Lengths

For optical networks it can be desirable to keep optical restoration paths as short as possible for signal quality reasons, especially if this can be done with little or no capacity penalty. Another reason we may prefer short restoration paths is real-time speed of deployment of preplans or development of restoration path-sets on demand. Speed does not necessarily depend on path lengths: In an SHN-type DRA restoration paths are assembled in a highly parallel way with almost no dependency on the length of paths. But in other DRAs or under GMPLS-type implementations, restoration paths are assembled by a series of reserve-and-connect transactions sequentially along each path. In the worst case the restoration path creation time is directly proportional to path hop counts or lengths. Other solutions to mesh restoration or preplan activation are likely to be based on distributed algorithms with characteristics somewhere between perfect parallelism and worst-case serialism. Overall, therefore, it is of interest to study how the average length or number of cross-connections per restoration path could be reduced in design. This was studied in [\[DoGr01a\]](#), which we summarize and discuss in this section.

In preceding methods, we can set the H to strictly cap the longest length of eligible restoration paths in the design, either by distance or hop count. This gives control over the longest restoration path that could be involved in the design. More generally, the step of generating eligible route sets gives us an ability to explicitly approve of any restoration path that might be used by any criteria we wish. But this is only a "veto" type of control: we can eliminate any route we don't approve of, but once the set of eligible routes is finalized, the solver has no particular bias to use the shortest routes that are eligible. In fact, since it is minimizing shared spare capacity totals, it will generally make flow assignments to restoration routes that are maximally sharable, not the shortest for each failure scenario individually.

To develop the design model for trading cost versus path length, let us work with the basic SCA formulation. The extension to JCA or MJCA follows easily from that case. The idea of a bicriteria formulation is that one can express two notions that are of concern for

optimization. In this case, we add a consideration that we want the total of restoration hop-lengths to be as small as possible to the usual goal of minimizing the spare capacity for 100% restorability of all span failures. Specifically, we set up the following objective function:

Minimize

Equation 5.87

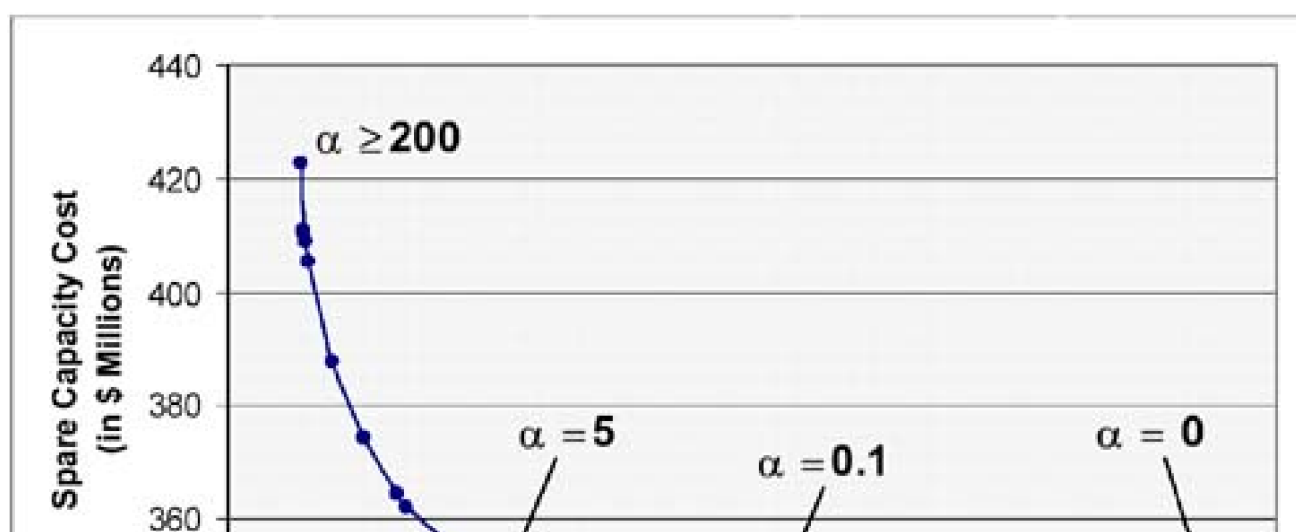
$$\left(\sum_{j \in S} c_j \cdot l_j \cdot s_j + \alpha \cdot \sum_{i \in S} \sum_{j \in S | i \neq j} \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p} \right)$$

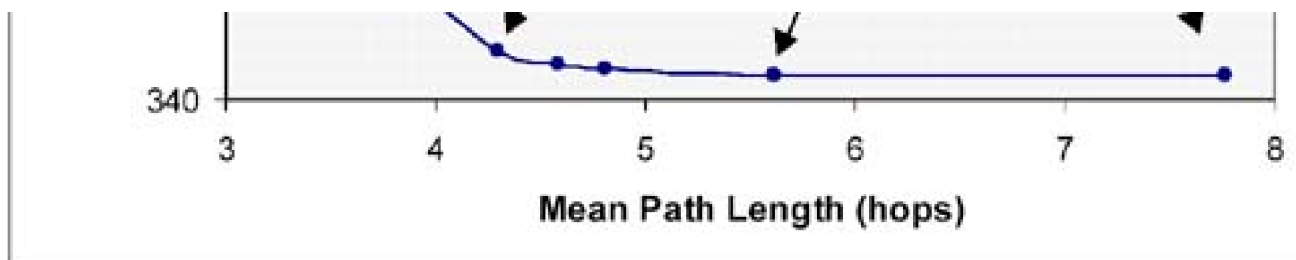
where c_j is the cost of a unit capacity per unit distance, l_j is length of span j , s_j is the amount of spare capacity on span j , and p indexes all distinct eligible restoration routes. $\delta_{i,j}^p$ are the same input parameters as in ordinary SCA which are 1 if the p^{th} eligible route for span i uses span j , zero otherwise. The second term is thus the sum of the flow-weighted $\delta_{i,j}^p$ indicators over all other spans j for all span failures i . This measures the total number of hops used on eligible routes by flow assignments made for all failure scenarios. Note that itself has no particular meaning to the problem but is necessary because the units of measure of the two terms are different. α is simply an arbitrary weighting. In effect, it tells the optimization how important the second term is relative to the first. By varying α over a suitable range we also can sweep out the "Pareto-optimal" combinations of capacity and path-length metric that are feasible. The bicriteria objective function is subject to the ordinary constraints previously introduced for SCA ([Section 5.3.4](#)).

Sample Results

[Figure 5-31](#) is a sample result obtained with this method on a 20 node, 28 span network model with 148 non-zero O-D pairs exchanging an average of 7.43 lightpaths each, via shortest path routes [[DoGr01a](#)]. Span distances range from 100 to ~1900 km and are used to distance-weight the spare capacity cost in the optimizations. [Figure 5-31](#) shows the total cost of spare capacity versus the average number of hops per restoration path in the resulting designs. The curve shows that restoration path lengths can be essentially halved, from nearly eight (7.8) in the baseline design, to somewhat less than 4.3 hops on average (at $\alpha = 5$) with an almost negligible (0.36%) addition of spare capacity. However, to reduce average hop lengths further, from 4.3 to 3.4 say, requires an investment of about 21% more in spare capacity. The rise of the cost toward infinity at the left also tells us that 100% restoration is infeasible in this particular network (no matter how much capacity we added) with fewer than 3.4 hops (on average). This arises from the graph structure itself.

Figure 5-31. Total spare capacity cost versus length of restoration paths (from [DoGr01a](#)).

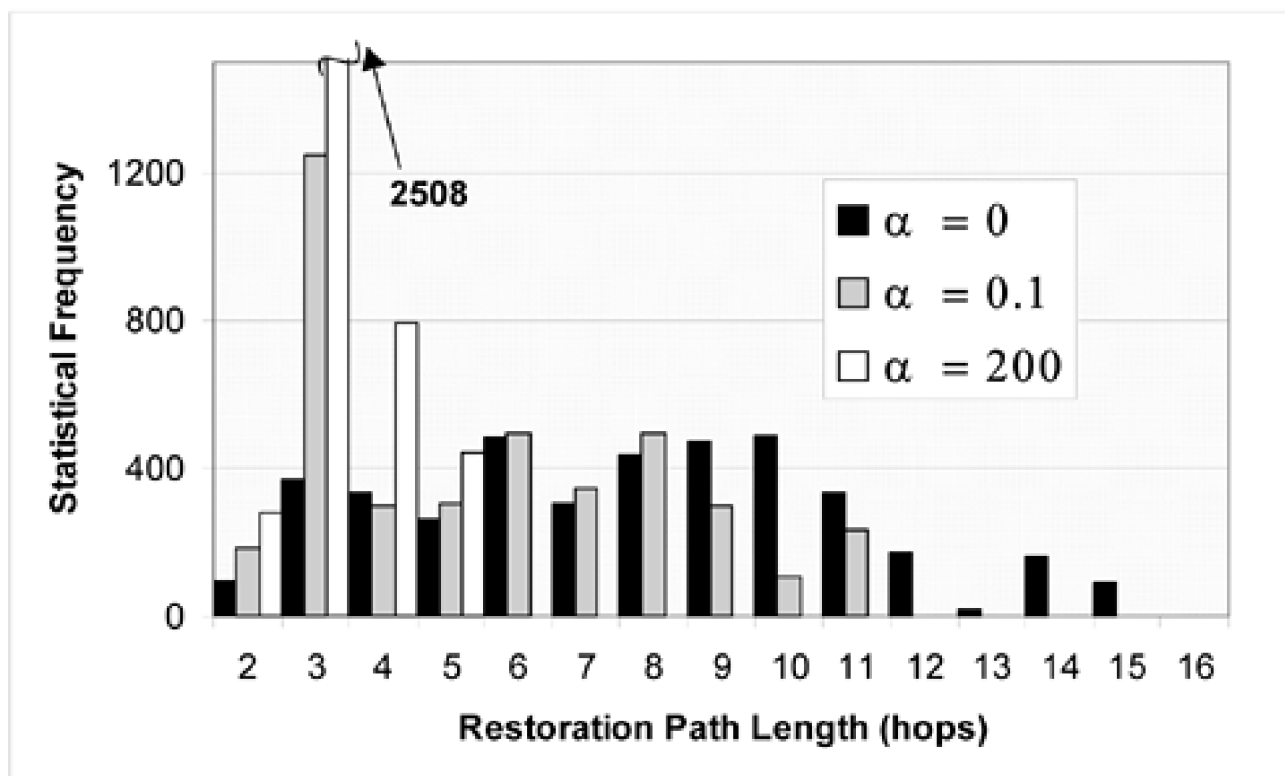




An especially interesting observation is that between $\alpha = 0$ and $\alpha = 0.1$ there is a significant tightening of restoration paths (by 2.1 hops on the average) at *literally* zero-cost. This shows an interesting property of the bicriterion approach in general: by simply "telling" the solver (through this addition to the model) that we care, even the slightest bit, about the side criterion, it is able to select what we like for us from among numerous strictly cost-equivalent solutions under the usual objective function. In this case it is able to select among completely equal-cost solutions to choose one that has an average restoration path of 5.6 hops rather than 7.8. With just the slightest non-zero emphasis on this consideration, the solver is able to recommend significantly shorter routing decisions that already existed within the minimum capacity solution.

Figure 5-32 is a histogram of restoration path length (in hops) showing how instances of the individually longest paths are even more dramatically reduced under the bicriteria pressure to consider path lengths as well as cost. At $\alpha = 0$ (conventional design), individual path lengths range up to 15 hops. When α is just nudged above zero alternatives to these longest routes are easily found. As α increases further, shorter and shorter average and longest routes are adopted, albeit with additional cost for spare capacity. At $\alpha = 200$ we can see that every restoration path is now held to being no more than five hops in length.

Figure 5-32. Statistical frequencies of individual restoration path lengths (from [DoGr01a]).



5.11.2 Bicriterion Design for Maintenance Risk Reduction

Another application of bicriteria methods is to enhance the ability of a span-restorable mesh network to restore against a failure on one span, while some other span is in a maintenance state that is using up some of the network's spare capacity. If the maintenance procedure involves rolling the working capacity on to spare capacity of the same span, or out into the network, then a single physical

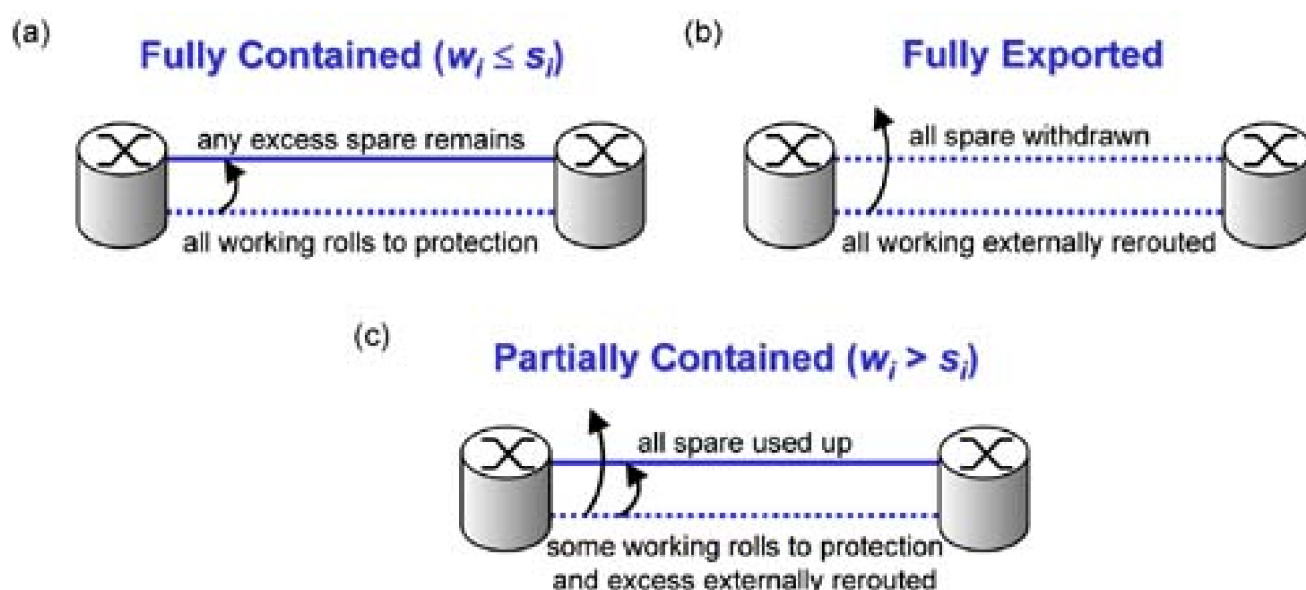
failure coupled with a span maintenance operation elsewhere may create a situation that is like a dual failure. Given the extent and growth rate of some networks, the frequency of maintenance actions for upgrades or repairs may be much higher than the rate of actual failures. This suggests that maintenance states coupled with single failures could be the cause of significant unavailability. In this section we will see how the policy for allocation of spare capacity, to support both maintenance-related uses and failure restoration, can mitigate the possible impact of these maintenance processes.

Functional Models for Effects of Span Maintenance

It is normal practice in ring-based networks, and in linear transmission systems with APS, to speak of a "roll-to-protection" procedure in which working capacity is switched to protection on the same span to facilitate repairs or upgrades on working channels or fibers. In a ring-based network this completely contains any effects from the maintenance to within the given ring, although the magnitude of the risk exposure on those other spans is of 100% loss of restorability should a failure occur while the protection channel is locked out for maintenance use. The same overall concepts apply in a mesh-restorable network but the roll-to-protection model becomes generalized. [Figure 5-33](#) illustrates the different functional models for the effect of span maintenance from a restoration viewpoint. More detail on the issues of maintenance and restorability interactions in span-restorable networks can be found in [\[GrCl01\]](#). This includes the concept of a "risk field" that can be computed arising from the withdrawal of spare capacity on a span under maintenance. It is a way of visualizing and quantifying the extent to which a maintenance action puts a network at theoretical risk of a partially unrestorable failure during the maintenance state. The three models for how maintenance affects the network's readiness to face a failure are as follows:

- If $w_i \leq s_i$ the move to protection is completely contained within span i itself. The network-wide effect is a withdrawal of w_i units of spare capacity from span i . The effective spare capacity of span i in this state is.
- If $w_i > s_i$, the move to protection is only partially "contained." It first uses up all the spare capacity on the maintenance span i , then the remaining working is rerouted over replacement paths through spare capacity on other spans of the network, around the maintenance span. The network effect is withdrawal of all spare capacity from the maintenance span plus a withdrawal of spare capacity on other spans as required to support the creation of maintenance replacement paths for $w_i - s_i = w_i$ units of working capacity.
- In the worst case the maintenance action requires turning down the entire span (all working and spare channels) and fully exporting the working signals to paths through spare capacity on the rest of the graph. This "fully exported" case is functionally identical to complete failure and restoration of the span in terms of its network-wide use and reduction of spare capacity.

Figure 5-33. Basic models for how span maintenance actions can affect network restorability.



The aim in the bicriterion design is to enhance the immunity of the network to these maintenance-related effects on restorability. There are

again two things to investigate: what enhancement can we obtain at zero extra cost, and, if we spend more to enhance this property, what does the trade-off curve look like?

Maintenance-Aware Spare Capacity Allocation (MA-SCA)

One initial idea to enhance the immunity is simply to bias the allocations of spare capacity so that, to the greatest extent possible, the fully contained model applies. This minimizes the extent of the *risk field*, or theoretical exposure to restorability loss elsewhere in the network, by virtue of consuming less spare capacity to support rerouted working flows in the maintenance state. The MA-SCA model is, as before, an ordinary SCA model with the following extended objective function:

Minimize

Equation 5.88

$$\sum_{j \in S} c_j \cdot l_j \cdot s_j + \alpha \cdot \sum_{i \in S} \max(0, w_i - s_i)$$

This states the intent of the new objective, but it cannot be implemented as stated. To do so, we draw on the techniques of [Section 4.16](#) 4 to implement this as:

Minimize

Equation 5.89

$$\sum_{j \in S} c_j \cdot l_j \cdot s_j + \alpha \cdot \sum_{j \in S} \nabla_j$$

in conjunction with the following new constraints:

Equation 5.90

$$\nabla_j \geq w_j - s_j; \quad \nabla_j \geq 0; \quad \forall j \in S$$

The new objective function minimizes the total cost of spare capacity plus a penalty term incurred when diversion of working capacity for a maintenance action cannot be fully contained within the span itself. By increasing α , a planner can allow a controlled increase in spare capacity to minimize the amount of externally diverted working flow in the partially contained roll-to-protection type of maintenance. The new constraint just measures any amount by which w_j exceeds s_j on a span, but keeps the difference from going negative. The idea is to minimize the cost of spare capacity but with a bias to tend to match up the w_j and s_j quantities on each span as much as possible. While this does not eliminate the loss of restorability due to maintenance actions, it can have two effects on the solution. At relatively light a the effect will be solely to bias the solution toward selecting among cost-equivalent solutions for an actual distribution of spare capacities that has a better overall matching in terms of $s_j \approx w_j$. In strictly optimal solutions to the single-criterion SCA problem one fairly often sees spans with $s_j = 0$ and others with $s_j > w_j$. This will give the needed bias to chose among cost-equivalent arrangements with better balance in the per-span working/spare ratios. At higher α the effect will be to invest in additional spare capacity to further enhance the number of

fully contained spans (i.e., those where $s_j \supseteq w_j$).

Sample Results with MA-SCA

Results with this approach were obtained in [DoGr02]. Based on that work Figure 5-34 illustrates how α can be varied to mediate the trade-off between additional spare capacity and the relative level of exposure to restorability loss from maintenance states. Each data point represents a network designed using the MA-SCA model with various α values. When α is zero, the design is optimized for capacity cost only and so acts as the benchmark design. The abscissa of Figure 5-34 normalizes the average risk field magnitude for each span being in a fully exported maintenance state, relative to that at $\alpha = 0$. The magnitude of the risk field is the sum of the theoretical restorability losses (in channel counts) on all other spans should they fail while the given span is under maintenance. Methods to determine this are given in [DoGr02]. The α value that corresponds to each data point depends on the test network but ranged from 0 (at the far right of each curve) through 500 (at the far left).

Figure 5-34. Bicriterion design result for trading risk to restorability from span maintenance against total capacity cost (adapted from [DoGr02]).

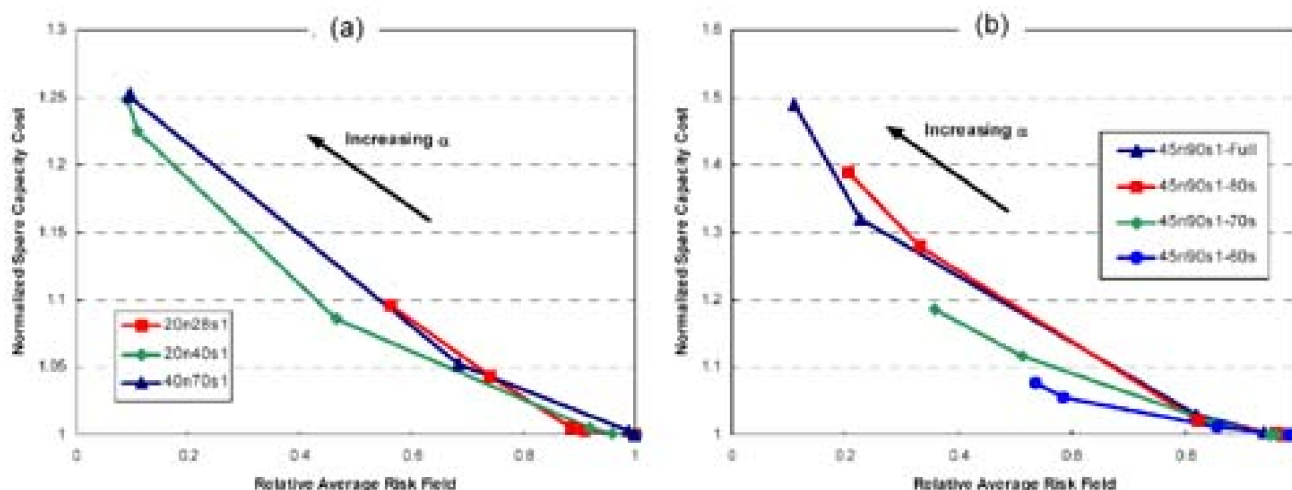


Figure 5-34(a) shows results for a network of 20 nodes and 28 spans (denoted 20n28s1) and two other networks similarly designated as 20n40s and 40n70s. The data shows that in the latter two networks the average risk field can be reduced by about 90% at the cost of about 25% increase in total spare capacity. The range of trade-off for the other test case (20n28s1), which is a much more sparse network, was more limited. To test further for the effect of average nodal degree four additional networks were tested, with results shown in Figure 5-34(b). These test cases are all subnetworks of various nodal degrees, obtained by systematically deleting spans from the highly connected 45n90s1 master network. The result clearly shows how the scope for risk reduction through MA-SCA design is larger for higher degree networks.

A notable use of the capability to trade risk containment off against capacity cost increases could be to generate just enough reduction of risk fields to support a certain fraction of high priority service paths at full maintenance immunity. If say, under all scenarios, the risk intensity on any span is never more than 60% it means that up to 40% of paths over all spans could be in this priority class. The maintenance-aware design approach also has the additional benefit of increasing the dual-failure restorability of the network as a whole and of further expanding the scope for the dynamic working capacity envelope mode of operation (Section 5.2.6) during non-maintenance states.

5.11.3 Bicriterion Design for Best Efforts and Preemptible Services

In Section 5.9 we looked at the capacity design problem in the context where not all services receive the same quality of protection (QoP).

A so-called "gold" service class was assured of restoration against any single-span failure, but "silver" service received only best-efforts and "economy" could even be preempted if its working capacity was needed for restoration of higher service classes. Because the bicriteria approach can let us express more than one optimization goal simultaneously, it is natural to consider a bicriteria approach to the multi-QoP design problem wherein we would seek secondary objectives such as maximizing the best-efforts restorability level, or minimizing the preemption of economy class services. As seen above it may be possible to obtain improvement in the latter side objectives without any increase in overall design cost simply by giving the solver a side criterion by which to effectively choose among otherwise equally good capacity designs. More generally, however, as we put more weight on the secondary criterion the design model will reveal a trade-off curve for total capacity required versus the extent of best-efforts restorability or measures of the preemption of economy services.

To maximize the restorability level of best-efforts service classes, while minimizing total capacity cost, we form a measure of the *unrestorability* of the best-efforts class services. Both elements of the mixed objective are then of a minimization sense. The design model uses all the parameters and variables defined above for multi-QoP formulations. The key new variable is:

- u_i^s is the number of unrestored silver class working capacity units on span i , when span i is the failure span. It is a non-negative integer.

The formulation then becomes:

Max-Best-Efforts

Minimize

Equation 5.91

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m + \alpha \cdot \sum_{i \in S} u_i^s$$

subject to:

- assured restorability for gold:

Equation 5.92

$$\sum_{p \in P_i} f_{i,p} \geq w_i^g \quad \forall i \in S$$

- unrestorability of silver class:

Equation 5.93

$$u_i^s = w_i^s - \left(\sum_{p \in P_i} f_{i,p} - w_i^g \right) \quad \forall i \in S$$

- non-negativity on silver unrestorability:

Equation 5.94

$$u_i^s \geq 0 \quad \forall i \in S$$

plus:

- all other constraint systems of multi-QoP ([Section 5.9.3](#)).

[Equation 5.94](#) expresses the combined goals of minimizing total modular capacity cost but also minimizes the number of unrestored best-efforts class services over all failures. The model is given in the most general context, i.e., that of joint modular design as this is the context that would yield the greatest scope for best-efforts restorability enhancement by routing best-efforts services to exploit modular capacity effects which inevitably provides excess capacity residuals without increasing modular total costs.

[Equation 5.92](#) asserts, as before, that gold class capacity must be fully restorable but it does so now with an inequality, permitting restoration flow assignments that exceed the requirement strictly for gold. In all prior models the corresponding restorability assertion constraint was an equality. The new constraint, [Equation 5.93](#), gives explicit definition to the number of unrestorable silver class demands in each failure scenario. It considers the actual total of restoration flow assignments in response to span failure i , relative to the amount of flow needed for gold class requirements and the number of silver class services present on the span. Note that with the inequality in [Equation 5.92](#) the restoration flow variables are implicitly generic and represent the total flow assignments for both gold and silver in the model. There must be enough restoration flow for gold, but any beyond that can be used for silver. In the prior multi-QoP formulations nothing required or pushed the total restoration flow assignments to eligible routes to exceed the requirement for gold services on each failure. Here, however, the fact that [Equation 5.92](#) is now an inequality, and that the unrestorability variables u_i^s are under minimization pressure in the objective, means that [Equation 5.93](#) will force a larger sum of restoration flow assignments than just needed for gold services. Under the current logical spare capacity it will force the largest feasible total restoration flow or, depending on α , enough flow so that u_i^s is driven to zero at the cost of driving up the spare capacities accordingly. Of course, in the limit $\alpha \rightarrow \infty$ the effect is as if all silver and gold demands were of equal priority for restoration.

At low α , however, [Equation 5.91](#) can force a redistribution of spare capacities without adding any new capacity (associated routings and working capacity may shift too) so that the $\sum P$ totals are increased above their gold-only requirements, thereby enhancing best-efforts restorability without any additional cost. As α is further increased the same constraints will cause addition of new extra spare capacity to further raise $\sum P$ quantities, thereby lowering the unrestorability sum in the objective function. The model can thus be used either to redistribute the minimum cost capacity or to trace out a complete trade-off curve of total cost versus best-efforts restoration levels.

Planning for Uncertainty

Before continuing, another interpretation and use of this model is worth noting: It is to think of the silver class demands as representing an allowance against uncertainty on the actual magnitude of gold class demands. To see this, go back to considering a single service class, all gold, but with the same mathematical structure as regards silver and gold classes (only) in the above. A matrix of gold demand requirements is provided and a secondary set of "additional demands" which are also to be made restorable, but only to the extent possible. Then it is possible to think of the gold demands as the expected or nominally forecast demands, which we want to be certain to be able to restore, and the "other demand" quantities (which we previously knew as silver) are just allowances for uncertainty in the forecast, for which it would be good to also be able to restore as much as possible (in case the forecast is off). Thus, one gets an overall planning model described by the following kind of statement: "We think the demand on O-D pair r will be this much, and the design ensures at least that level of restorability. However, we've also input a contingency that we might be off by up to this much more on estimating the demand (a.k.a the silver amount), and the design maximizes the level of restorability in case we are."

Minimizing Preemption of Economy

A subtlety of the model as written above is that economy class service capacity is inherently preemptible for both gold and silver

restoration. This arises from the inequality in [Equation 5.92](#) and the non-specific nature of the $f_{i,p}^p$ quantities. This means that as given, the maximization of best-efforts restoration would implicitly also be at the expense of preemptible working capacity. If the priority scheme

does not involve preemptible working capacity at all then the w_j^e variables are removed throughout and the model will have the properties of assuring gold restoration and maximizing best-efforts restoration, without any preemption. If, however, the intent of the priority scheme is to have gold class services able to preempt economy class, but *not* have best-efforts services able to do this, then the capacity design model needs to split the restoration flow variables and define a new internal variable as follows:

- $f_{i,p}^g$ is the amount of restoration flow allocated to the p^{th} eligible route for restoration of gold class capacity on span i .
- $f_{i,p}^s$ is the amount of restoration flow allocated to the p^{th} eligible route for restoration of silver class (best-efforts) capacity on span i .
- $f_{i,j}^{gener}$ is the net amount of gold-service restoration flow crossing span j in response to failure of span i after allowing for gold class restoration flow to preempt any economy class capacity on span j (non-negative).

The real purpose for defining the last variable as an explicit quantity is to be able to assert its non-negativity and to allow combination of gold and silver flows in generating spare capacity, while recognizing that gold can preempt, but silver cannot. With this categorization of the restoration flow assignments for each of the top two service classes, we can then express the intent that gold restoration can preempt economy, but while best-efforts restoration levels are maximized, they cannot do so at the expense of economy services, by returning the constraint for gold class restorability to an equality but now specifically based on gold class restoration flow variables only:

- assured restorability for gold:

Equation 5.95

$$\sum_{p \in P_i} f_{i,p}^g = w_i^g \quad \forall i \in S$$

The unrestorability of silver class services is similarly modified to consider the residual capacity left unrestored explicitly in terms of silver class restoration flow assignments.

- unrestorability of silver class:

Equation 5.96

$$u_i^s = w_i^s - \sum_{p \in P_i} f_{i,p}^s \quad \forall i \in S$$

The intended spare capacity generating constraint can then be structured by first defining the intermediate variable $f_{i,j}^{gener}$:

- net gold flow needing spare:

Equation 5.97

$$f_{i,j}^{gnet} = \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p}^g - w_j^e \quad \forall i,j \in S^2 | i \neq j$$

Once we have the net amount of gold restoration flow that requires actual spare capacity (as opposed to being realized through preemption of economy capacity) we can combine this with any additional best-efforts restorability flow for which the trade-off of criteria expressed by the objective function may require additional spare capacity.

- total spare capacity (for silver and gold):

Equation 5.98

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p}^s + f_{i,j}^{gnet} \quad \forall i,j \in S^2 | i \neq j.$$

All other constraints are the same as in multi-QoP. We stress in closing that in this model it would be erroneous, albeit tempting, to simply substitute the expression for $f_{i,j}^{gnet}$ (Equation 5.97) into Equation 5.98. The point is that $f_{i,j}^{gnet}$ must stand alone as an intermediate variable to which non-negativity can be asserted, before it is combined with best-effort class restoration flows. Otherwise, if combined in one expression, the economy class capacity on a span j is again inherently applied against either gold or silver class restoration needs. By explicitly applying it only to gold restoration flows in Equation 5.97, and bounding this variable to be non-negative, we obtain the correct intent for the total spare capacity on span j in Equation 5.98— actual spare capacity has to be dimensioned to meet the net requirements of any best-efforts flow to be supported plus any gold class restoration residual in excess of the preemptible capacity on the span.

A related bicriteria design model focuses on controlling the impact on preemptible class services, due to restoration of overlying gold (and/or best-efforts) service classes. For a discussion of this design approach we will assume that gold and silver service classes may both preempt economy class services to obtain needed capacity for gold restoration and to maximize the best-efforts restoration for silver. Gold is still assured of 100% restorability by designed-in spare capacity or preemptible working capacity. Silver services may use any spare or economy class capacity not needed for gold service restoration. Best-efforts performance in this situation would be generally better than above where best-efforts can only use true spare capacity, where unneeded by gold. But this by no means assures the best-efforts category of 100% restorability since there may still be inadequate spare plus preemptible capacity to reach 100% restoration for the best-efforts services. Of course letting both gold and best-efforts preempt economy capacity is the most disruptive policy from the standpoint of economy services.

The bicriteria design model will again minimize capacity cost but with a side criterion of trying to distribute capacity and choose routings so that the preemption of economy services is kept as low as possible. The principle idea is to quantify the amount of economy class capacity used to offset spare capacity in assigning restoration flows for gold or silver. Because we are not now carrying any aspect of maximizing best-efforts restoration levels, economy class capacity will be in a position to "push back" against the silver class services, protecting itself against preemption by silver class, at the latter's expense. The economy class capacity will not, however, be able to escape preemption by gold class restoration needs. The new internal variable we need to define is:

- $b_{i,j}$, the amount of economy class capacity on span j that becomes preempted in the restoration response to failure of span i .

Min-Preemption

Minimize

Equation 5.99

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m + \alpha \cdot \sum_{i,j \in S^2 | i \neq j} b_{i,j}$$

subject to:

- Routability: [Equation 5.61](#)
- Working capacities: [Equation 5.62](#) to [Equation 5.64](#)
- Restorability for gold: [Equation 5.65](#)
- Economy capacity used:

Equation 5.100

$$b_{i,j} = \sum_{p \in P_i} \delta_{i,j}^p \cdot f_{i,p} - s_j \quad \forall i,j \in S^2 | i \neq j$$

Equation 5.101

$$b_{i,j} \leq w_j^e \quad \forall i,j \in S^2 | i \neq j$$

- Spare and preemptible capacity: [Equation 5.66](#)
- Modular total capacity: [Equation 5.67](#)
- Non-negativity: [Equation 5.68](#) and [Equation 5.69](#)

The bicriteria objective tries simply to minimize the total unit count of economy services preempted over all possible single-span failures. This too may be normalized or given specific weightings in any given detailed study. Because this model solves explicitly for all $b_{i,j}$ they are also available for statistical analysis of the impact of the frequency with which failures on other spans cause preemption of economy class services crossing each span of the network.

Finally, it is conceivable to combine all three considerations: min-capacity cost, best-efforts restorability, and minimized impact on economy services in a three-part multi-criterion objective. The practical problem with doing so is, however, the high dimensionality of the Pareto-optimal surface. More likely one would have a reasonable *a priori* guideline as to acceptable values on one or more of these criterion. A most practical approach might be first to obtain the total design cost of a single-criterion cost-minimized design that simply routes all demands, without any restorability considerations. One would then use this cost value as the basis for a total cost budget, say 125% of the basic figure for routing only. With this set as an upper limit on the total cost of the design one can then study composite

designs along the trade-off curve between preemption and best-efforts maximization, with an assurance of 100% restorability of gold class services.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Chapter 6. Path Restoration and Shared Backup Path Protection

In a path-restorable network or a path-protected network, the reaction to a failure takes place from an end-to-end viewpoint for each service path affected by the failure. Path-oriented survivability schemes have several advantages. One is that they are more amenable to customer-level control and visibility. Span restoration is rather inherently a function of the transport network itself, whereas path-level protection or path reestablishment can be a function that is put in the users control, or under the control of a service layer node such as a router. In shared backup path protection (SBPP) it is also possible for the users to know ahead of time, and even control, exactly where their service will be rerouted in a failure. This is sometimes said to be important to customers. In contrast with span restoration, customers may be assured of restorability, but would not ordinarily know the exact reroute that their service would take for a given failure (although if important, this could be worked out in advance from the protection preplans). Because path restoration spreads the overall rerouting problem more widely over a network as a whole, it is also usually more capacity-efficient than span restoration, although the difference depends on network topology and issues we will cover such as stub-release. Path schemes also provide an inherent form of response against node failure that is not as readily available with span restoration. Path-oriented schemes also only require fault detection at the path end-nodes, and this can be attractive for transparent or translucent optical networks where signal monitoring may not be available at intermediate nodes. On the other hand, path restoration or protection schemes are generally not as fast as their span-oriented counterpart due to the greater distances and numbers of nodes involved in signaling. Availability issues can also arise when working and backup paths are both long. There are also issues of scalability due to backup-sharing databases with SBPP and generally greater overall complexity is inherent with path restoration: SBPP and dynamic path restoration both involve considerations to address the "mutual capacity" issue which is the central complicating difference between span and path restoration.

In thinking about an isolated path failure, such as that which might arise from an interface circuit pack failure, path restoration can be fairly simple: one simply switches over to a predefined backup route, or using GMPLS auto provisioning one can expect a fairly high probability of being able to just "redial" a new service path. Too frequently, however, proposals for path restoration simply extend this single-path notion of recovery to apply for entire cable cuts. But the situation really is quite different when a possibly large number of paths fail simultaneously under a cable cut. Then the construction of a set of simultaneously feasible end-to-end replacement paths within a minimal amount of spare capacity is quite complicated and requires careful coordination if maximal restoration is to be achieved. SBPP and dynamic path restoration (PR) are the two main schemes that address this central issue of coordination. GMPLS auto reprovisioning provides a kind of ad hoc restoration mechanism but does nothing to effect such coordination. We will briefly explain this aspect of GMPLS when used for restoration, and illustrate the hazards it involves, but then devote ourselves to SBPP and PR.

In *path restoration* (PR), globally efficient restoration path-sets are formed in real-time in response to the actual network state and the specific failure that arises. This can be either through a centralized computation, through periodically updated preplans in the network nodes, or through a distributed protocol embedded in the network to approximate the optimal path-set construction. The failure specificity of *path restoration* makes it possible to reuse working capacity on surviving portions of failed paths—a concept known as "stub release." Such reused working capacity can take the place of certain amounts of spare capacity, making path restoration with stub release the most capacity-efficient of all survivability schemes.

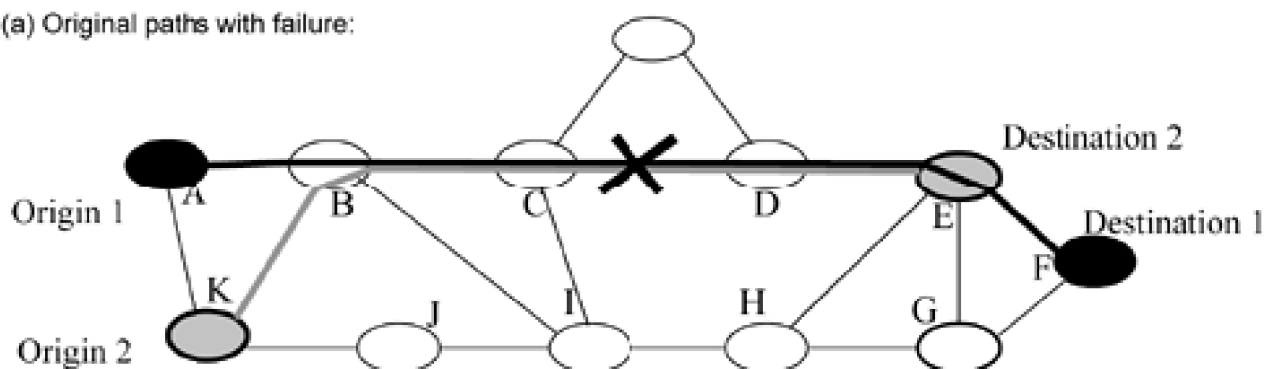
Depending on technology assumptions, however, it may be difficult to know immediately where the failure has occurred. The failure condition may only be known at the end nodes of the paths affected. In this case one can decide in advance upon a set of end-to-end backup paths that are fully disjoint from each working path protected. This gives up some capacity efficiency because one cannot use stub release. It also requires a global view for coordination of backup-path capacity sharing at the time the path is provisioned. But the advantage is that it allows activation of the switchover by the end-nodes without any other knowledge about the failure. For efficiency, spare capacity is shared over backup paths that have disjoint working routes. It is in coordination of this sharing that a global view of network state is required. The resulting scheme is called either *failure independent* path protection or more commonly *shared backup path protection* (SBPP).

6.1 Understanding Path Protection, Path Restoration and Path Segments

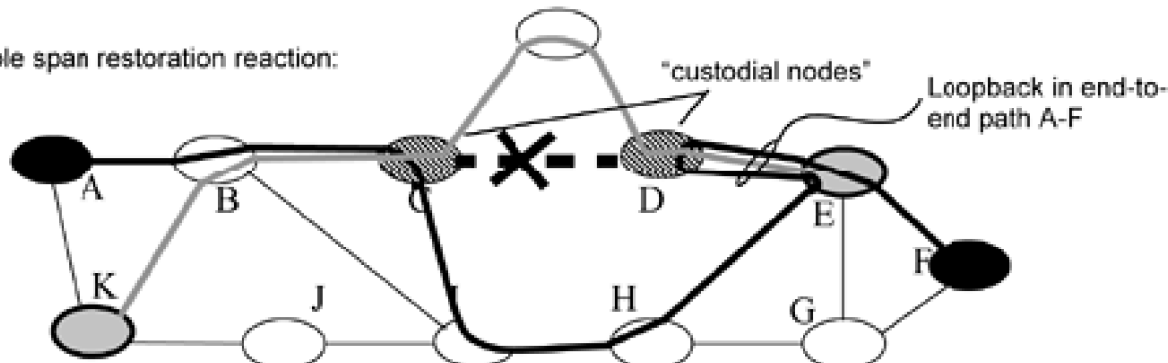
Let us introduce some basic concepts about path-oriented recovery in general by way of a comparison to span restoration. In span restoration, the replacement paths for restoration start and end at the nodes adjacent to the failed span. In path restoration replacement paths are effected via end-to-end rerouting for every working path affected by the failure. The difference is illustrated for a two-path example in [Figure 6-1](#).

Figure 6-1. Comparing span versus path restoration.

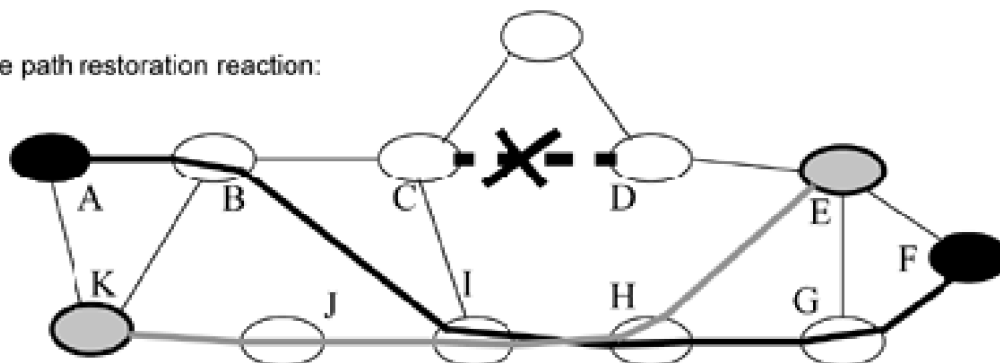
(a) Original paths with failure:



(b) Possible span restoration reaction:



(c) Possible path restoration reaction:



The failure of span C-D affects both paths. Note that our primary concern will continue to be survival against *span* cuts as the main failure scenario. That the failures still occur on spans is not in contradiction to the recovery being via path restoration. [Figure 6-1\(b\)](#) shows a span restoration solution in which a back-haul or loopback arises on span (D-E). In contrast, [Figure 6-1\(c\)](#) shows a possible PR response. By acting farther back from the break the end nodes can take advantage of more overall options for rerouting and are able to avoid the back-haul apparent in [Figure 6-1\(b\)](#).

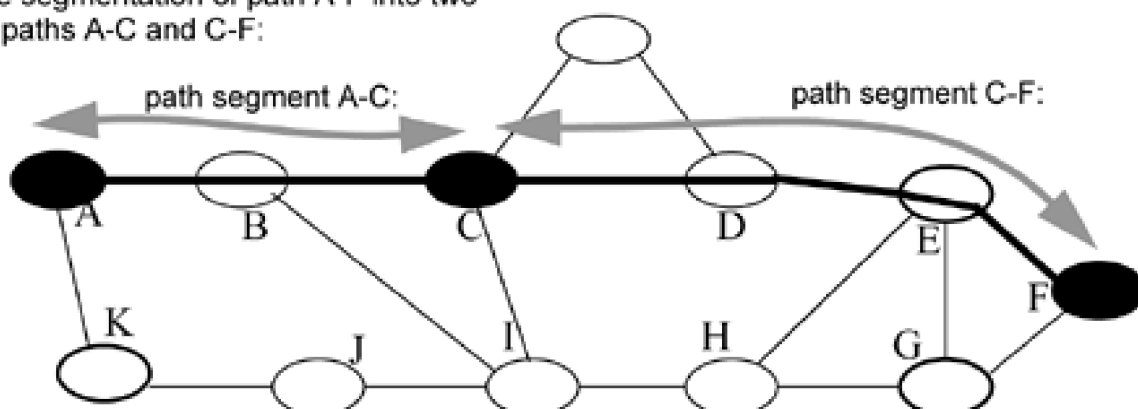
Span restoration is like deploying a set of road detours around a construction site. Drivers following their basic route will come up to the detour, follow the detour, and continue on their original route immediately at the other side of the disruption. As long as the detour is of the same capacity as the disrupted road, we know there will be no congestion. The end-to-end trip is still completed by the driver but if the driver, with a more global view, knew about the road outage before setting out from his origin, he may have chosen a completely different end-to-end route. The latter is the analogy to path restoration. But of course if a large number of drivers independently make the same decisions to take a different route altogether, they may unwittingly converge at unpredictable other locations causing congestion, hence the greater extent of global coordination needed in path-oriented recovery.

Segments—A Special Case of Paths

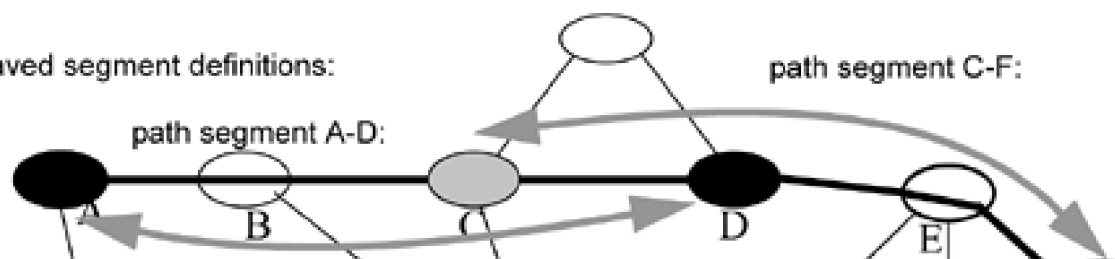
Figure 6-1 also suggests the range of options that may exist in between pure span and pure path restoration, depending on how far back from the failed span the route of the restored path departs from the prefailure routing. The point is that nodes A and F could actually be the origin-destination nodes or just a pair of nodes designated to act as end nodes for protection or restoration of the *path segment* between them. In practice path segments may be defined at entry and egress points of separately administered optical network domains, or a long path may be partitioned into segments simply to improve availability. Figure 6-2(a) illustrates simple path segmentation and Figure 6-2(b) shows the concept of interleaved path segmentation which has been proposed [KrPr00] to avoid creating a single point of failure at a node such as C in the simple segmentation example. Thus, in subsequent sections, we speak for convenience in terms of the end nodes being the O-D nodes, but it needs to be understood that all the same developments and methods apply to any set of path segments defining intermediate protection domains on the path. Simple segmentation simply alters the apparent demand matrix. With interleaved segmentation, the planning of protection paths and capacities is again logically identical to planning for a set of end-to-end paths, but in the allocation of working capacity it needs to be noted that where path segments overlap, such as at C-D in (b) above, capacity for the path is allocated only once and not once for each segment where such segments overlap. Although the logical protection segments overlap, there is only one instance of the path signal itself on all spans on the end-to-end route. Note that span restoration is just the limiting case of segmented path protection. Interleaved segmentation is also handled by the basic methods for non-joint PR or SBPP design on the corresponding segmented version of the demand matrix and appropriate definition of the path segments affected by each failure scenario to reflect the interleaving information, e.g., if span (B-C) fails in (b), the affected "path" is that between nodes A to D, whereas in (a) it is the "path" between A to C. If a jointly optimized model for interleaved path segmentation is desired, an extension is required to the models that follow to recognize that on a span such as C-D, the segmented paths are really using the same unit of working capacity, not two working channels, where the segments overlap.

Figure 6-2. Path segmentation concepts.

(a) Simple segmentation of path A-F into two apparent paths A-C and C-F:



(b) Interleaved segment definitions:

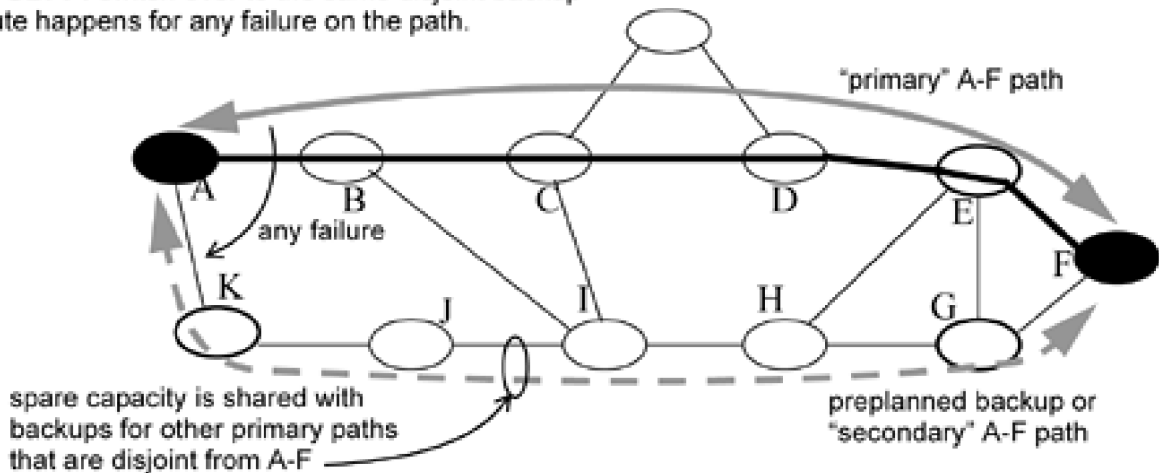




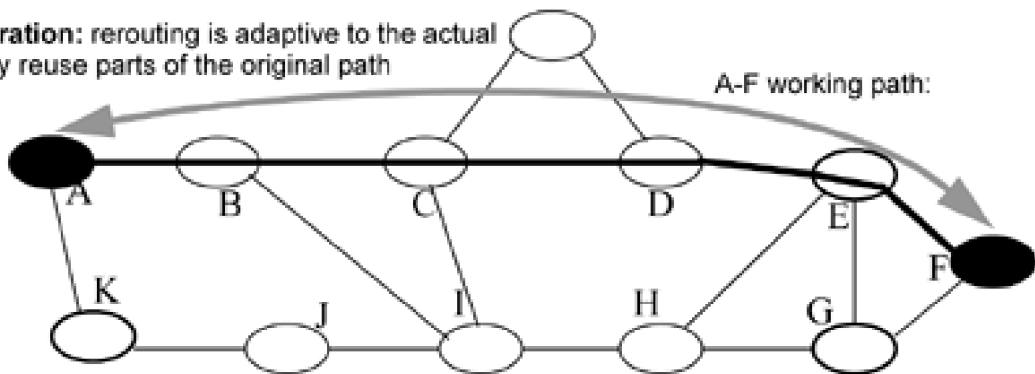
The main differences between PR and SBPP are illustrated in [Figure 6-3](#). In SBPP a single disjoint backup route is preplanned for each working or "primary" route. Spare capacity to form backup paths can be shared for efficiency over several backup paths which correspond to mutually disjoint primary paths (i.e., such paths won't need the backups at the same time). Under path restoration, the rerouting response is failure specific, and adaptive to the actual spare capacity environment and the set of other paths that fail simultaneously. The restoration path for any one node pair, may reuse any part of its prefailure route and capacity.

Figure 6-3. The main differences between SBPP and Generalized Path Restoration.

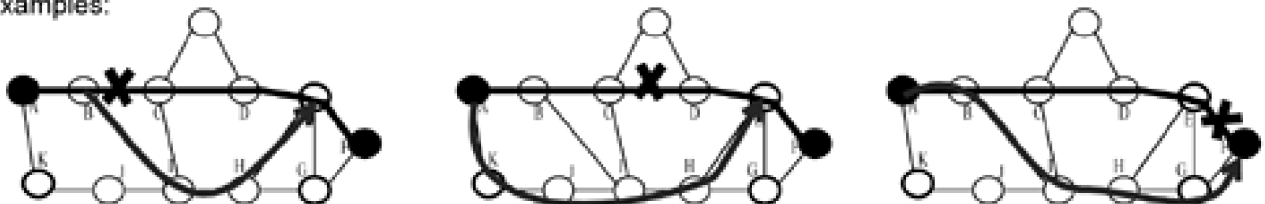
(a) **SBPP:** switch-over to the same disjoint backup route happens for any failure on the path.



(b) **Path restoration:** rerouting is adaptive to the actual failure and may reuse parts of the original path



Examples:



6.1.1 Is Path Restoration Just Span Restoration Without Loopbacks?

A frequent notion is that path restoration is essentially just span restoration where loopbacks, such as in [Figure 6-1\(b\)](#), are detected and removed. The idea is that if you detected and "undid" any loopbacks that arise in span restoration, you would arrive at path restoration mechanism and, hence, realize the benefit of lower spare capacity that comes with a path-restorable design. In [Figure 6-1\(b\)](#) for example,

the idea would be to detect (at node E) that the normal route of the path between A to F is through node E. One could therefore break the loopback at node E and alter the restored routing from ABCIHEDEF to ABCIH-EF, eliminating the EDE loopback segment *and* (the thinking is) thereby also eliminate the spare capacity on span ED needed to support the loopback.

However, such "loopback elimination" does not by itself bring us to the greater efficiency of path restoration. Often the detection and elimination of a loopback routing under span restoration will yield no reduction in spare capacity at all if all other failure scenarios are also considered. The reason is that under design for span restoration, the spare capacity that is used to support loopback in one particular failure scenario is almost always also used in a more essential (i.e., non-loopback) under other failure scenarios. This is true even for optimal designs. As with BLSR rings, which loopback around their entire length, the appearance of loopbacks in span restoration is part and parcel of realizing a simple and fast scheme. Technically, it is not loopback elimination per se that will reduce spare capacity of the overall design but *forcer reduction*. The forcer concept was introduced in [Chapter 5](#) and is discussed further in [Chapter 11](#) and later in this chapter where heuristics for path-restorable design are considered. Once the dimensioning of spare capacity on a span is set by its corresponding forcer, that spare capacity may be harmlessly reused in many other contexts to support loopbacks or other routing constructions in response to non-forcer span failures. One case where the elimination of spare capacity that is placed *only* to support loopback is effective is in multi-hop chain subnetworks. This is the essential source of the greater efficiency of the meta-mesh designs (in [Chapter 5](#)) relative to ordinary span-restorable capacity designs. But in general on-line detection and elimination of loopbacks alone would not bring us to the efficiency of a true path-restorable networks. Thus, the existence of loopback routings in span restoration is not really so inefficient and their elimination is not the main reason for considering path restoration.

6.1.2 Concept of Mutual Capacity in Path Restoration

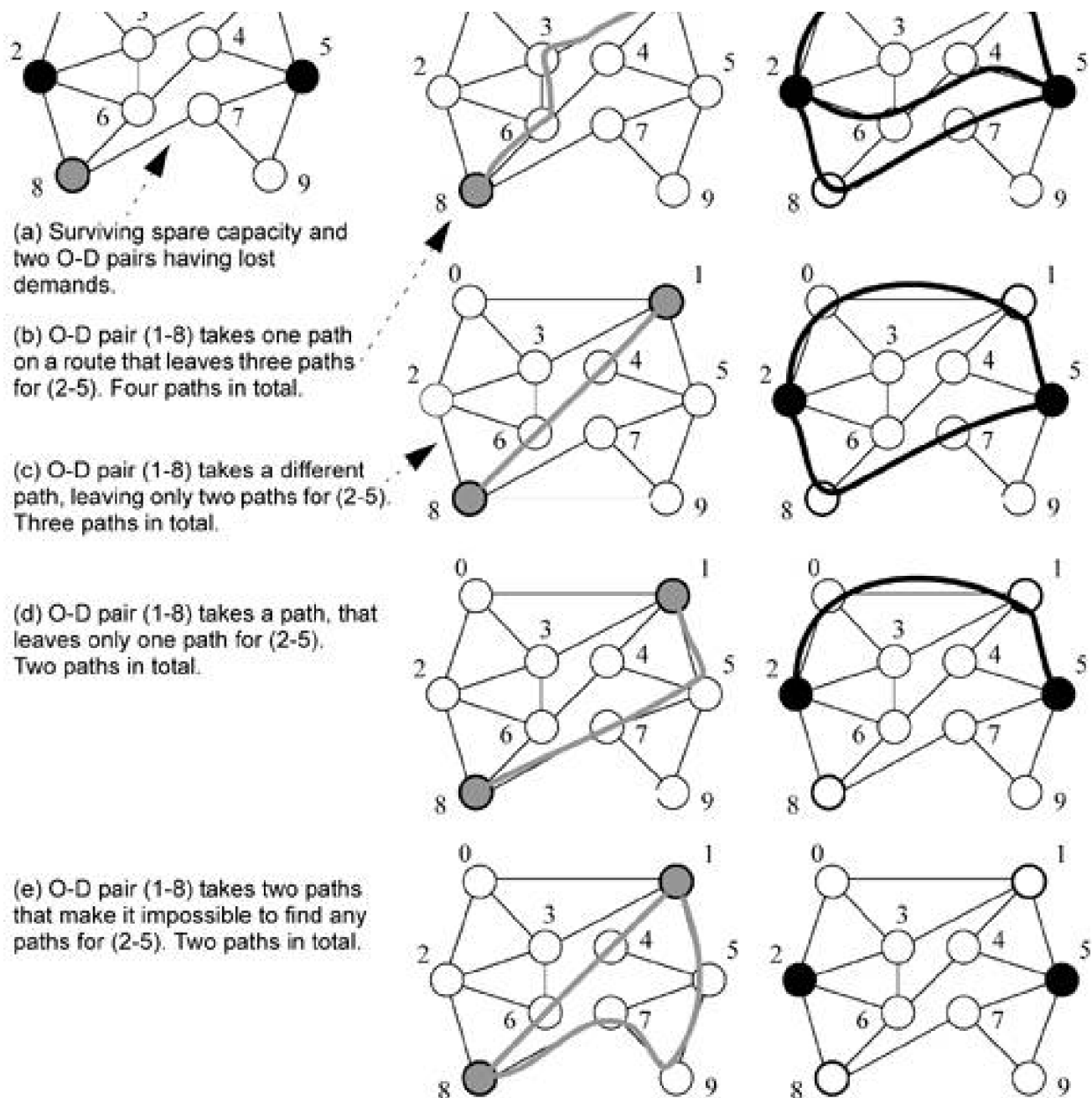
In span restoration the routing problem is to find a replacement path set in which all paths have the same end nodes, i.e., to find k paths between two nodes where k is the lesser of the number of working paths on the failed span, or to find the maximum number of feasible replacement paths. In either case it is a *single commodity* flow problem for which we have already seen solutions of low polynomial order complexity (i.e., ksp or max-flow). By comparison, the path-oriented approach is inherently a multi-commodity simultaneous flow problem. It involves identification of the different origin-destination nodes of each path that are disrupted by the failure scenario and production of multiple subsets of restoration paths within a single pool of spare capacity. The central issue that makes this a much more complex rerouting problem is called "mutual capacity". The basic problem has been recognized before in the context of ordinary routing of working demands, going under such descriptive names as "criticality avoidance" [[HoMo02](#)] or "courteous routing" [[BeBl00](#)]. As it pertains to path restoration, the same basic problem arises, but it pertains to the simultaneous routing restoration flows for multiple distinct end node pairs through an environment of shared (i.e., mutual) finite capacity.

To illustrate, consider a number of simultaneously affected O-D pairs and imagine for argument's sake that they each took a turn in sequence to obtain restoration paths. One O-D pair may have 20 equivalently good route choices that will restore it, but for another O-D pair there may be only one particular route on which it can obtain restoration paths. What if the first O-D pair chooses a route that uses up the spare capacity on the only possible route for the second O-D pair? More generally, how can the exact route choices made by each of a multitude of O-D pairs be coordinated so that the spare capacity that is crucial for restoration of one O-D pair is not blindly used up by another, which may have had different routing options in any case. This is called the *mutual capacity* problem in reference to the central quandary of this situation: to which pairs should we allocate the spare capacity of a span among the many that try to seize it?

[Figure 6-4](#) shows how easily the path-set choice for one O-D pair can interfere with the feasible paths for other O-D pairs in a way that may be detrimental both to the total restoration level and may create severe differences between the restoration levels achieved for different individual O-D pairs. The overall complexity in a real network will be vastly greater than in [Figure 6-4](#), but even in that simple example we can see that if O-D pair (2-5) happens to grab the three paths shown in (b) first, then O-D pair (1-8) is limited to at most one path. But if pair (1-8) first seizes the two paths in [Figure 6-4\(e\)](#), the tables are turned and O-D pair (2-5) is completely shut out and remains *disconnected* as part of the overall restoration response. And many other combinations are possible with varying total and individual O-D pair recovery levels depending on the topology, the exact random order assumed, and the exact capacity distribution. The example also illustrates the concern with relying on a totally ad hoc approach to path restoration such as "GMPLS auto reprovisioning" in which every affected end-node pair will simply conduct its own independent shortest-path reprovisioning attempt. This is an understandably tempting idea because everything needed to do so is provided by OSPF-TE and GMPLS protocols.

Figure 6-4. The concept of "mutual capacity" in path restoration (adapted from [[Iras96](#)]).





From a standpoint of mutual capacity considerations the notion of relying on a "mass redialing" approach for restoration is, however, flawed and risky because the outcome of independent mass redialing is fundamentally unpredictable. No assurances can be given as to the overall recovery level or distribution of recovery levels for any given O-D pair. Conversely, if assurances of 100% restorability were to be given, a large amount of extra capacity would have to be provided in the network. Such an assurance would really only be certain with a reservation of 100% redundancy on a disjoint backup route for every O-D pair. Only then is the blind seizure of a replacement path on the next shortest route strictly assured to be without deleterious mutual capacity interactions.

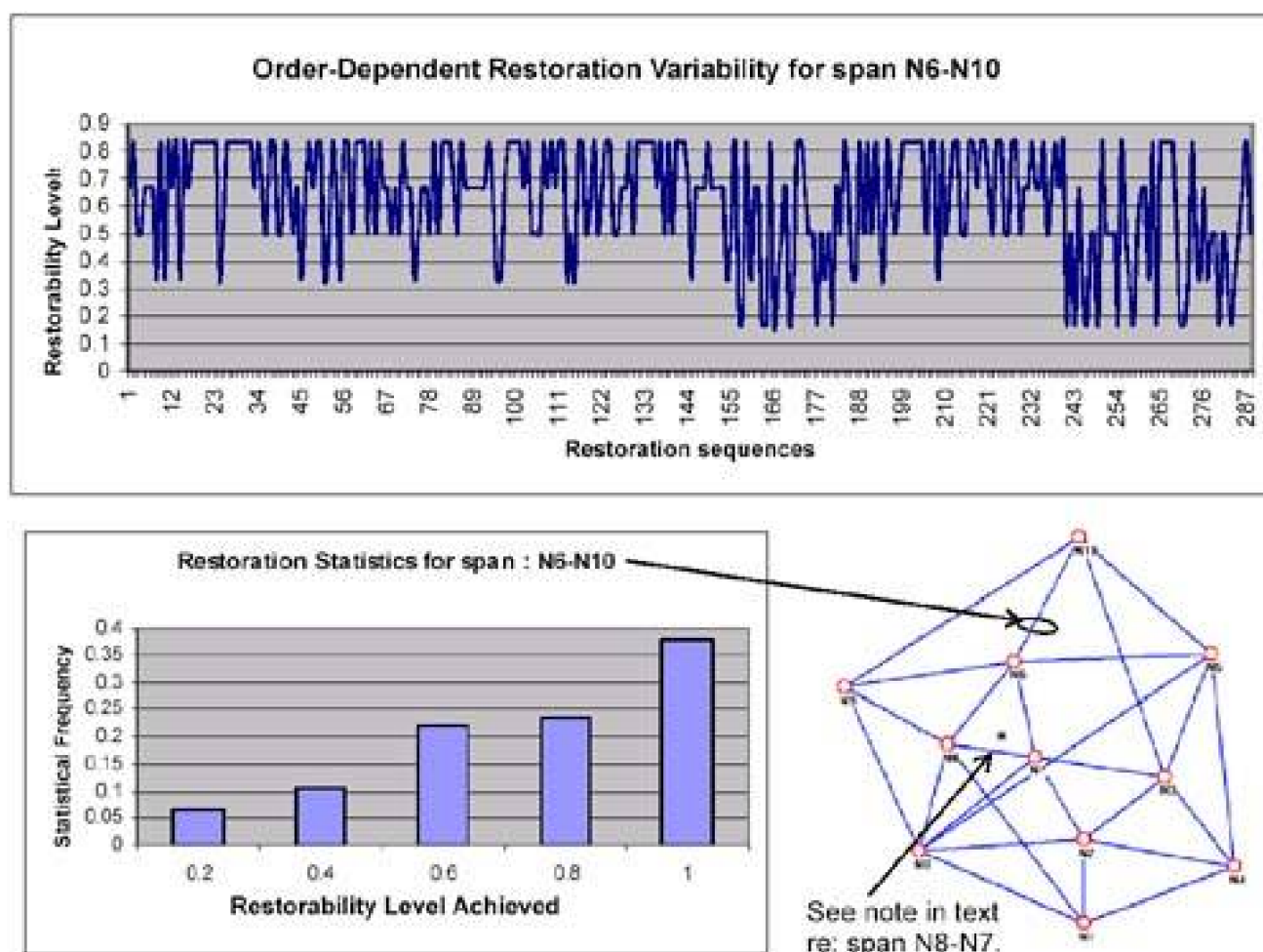
6.1.3 Experiments Simulating GMPLS Auto Reprovisioning

Some evidence in support of the cautionary message voiced above is provided by tests with randomly sequenced independent path reestablishment trials as a model for the GMPLS "mass redialing" approach to restoration. Experiments for inclusion here were conducted by G. Kaigala of the author's research group using the published topology and demand matrix of the COST239 network. Demands were shortest path routed and, using methods that follow in this chapter, a spare capacity allocation was designed that made the network 100% path-restorable against any single-span failure. A separate path restoration routing program validated that the network as designed was 100% restorable under appropriately coordinated multi-commodity restoration path-sets within the available spare capacity.

Capacity allocation under GMPLS-based restoration was then simulated as follows: Upon each span cut, all of the affected demand pairs are identified. We assume that each pair of affected end nodes begins a path reestablishment phase using OSPF-TE data to select the shortest replacement route and CR-LDP to seize a path on the identified route. We do not model the real-time aspects of OSPF-TE database resynchronization nor possible contention among simultaneous CR-LDP capacity reservation and seizure processes. We model these processes as functioning perfectly and we look only at the purely capacity-related outcome of each node pair independently obtaining a path on its second shortest surviving route that has enough capacity. The philosophy is that, at best, the overall dynamics of the mass simultaneous effort at individual path replacements could only be equivalent to the result of assigning each pair an undisturbed chance to obtain its replacement path in a random sequence over all the affected node pairs. After each individual shortest path replacement, the available capacity is updated and the next node pair in the sequence gets its chance, and so on until all affected node pairs have had a chance to find a replacement path.

Figure 6-5 is a sample result showing the overall restorability of demands that were affected by the indicated span cut. The data is for several hundred random orderings of the pair-wise recovery attempts with the argument that the net effect of GMPLS mass reial attempts would be essentially equivalent to such a random trial sequence. The significance is that these are unbiased samples from the space of all possible sequences among affected demand sets. So each of them represents an equally likely roll of the GMPLS mass reial "dice." GMPLS actually has no inherent knowledge of how to sequence them, nor any mechanism to structure such sequencing so that each attempt is not interfered with by the dynamics of other concurrent attempts. This is why we say that the outcome of these experiments is the limiting best-case outcome as it depends purely on basic routing and capacity considerations—which are also repeatable by others. (In contrast, any attempt to also model the signaling dynamics is probably not repeatable because it is so utterly dependent on exact implementation details and simulation timings.) Random timings and delays in signaling interactions with capacity seizure effects, protocol delays, and network timing effects will ultimately determine the *effective* sequence of an actual mass concurrent reattempt event.

Figure 6-5. Experimental results showing the effect of mutual capacity contention on the restorability levels obtained with GMPLS auto reprovisioning for restoration.



The top trace in Figure 6-5 shows the first few hundred results of the recovery-sequence trials in terms of the overall restorability outcomes. The histogram summarizes the data from thousands of such trials indicating that there is less than a 40% chance of 100% restorability of affected demands. 100% restorability is achieved for a good number of individual trials, confirming that there is enough

spare capacity to support that outcome, but it is as if we are picking a lock with a blindfold on: only a few lucky sequences will unknowingly conform to the mutual capacity structure that is present. While the span shown in [Figure 6-5](#) is typical, one of the worst situations in the same network is for the more central span (N8-N7)—highlighted with * in [Figure 6-5](#). The mass GMPLS-type recovery from this failure scenario exhibited only a 24% chance of full restoration and 22% of the time showed restoration levels at 20% or below. In trials on another larger network, one failure scenario was encountered for which the ad hoc restoration process had under 15% chance of ever achieving over 80% restoration, and never exhibited 100% restoration under any of the random-order sequence trials.

As stressed, these trials also speak only to the purely capacity-related issues of such ad hoc path restoration. In reality the time dynamics and the net outcomes could be even less satisfactory if protocol interactions, particularly CR-LDP seizure contentions, crankback dynamics, and OSPF-TE update lags cause path provisioning failures and repeat attempts. Here, each replacement path-finding effort was individually via perfect shortest-path routing with complete global knowledge of the topology and available capacity—data which would only be available to GMPLS nodes when OSPF-TE is completely reconverged. In related studies we also repeated the trials with a uniform allocation of excess capacity to each span to see the extent to which overprovisioning of capacity can "buy one's way" out of this problem. Findings in the same network were that if 150% excess spare capacity is provided on each span (i.e., 1.5x), average restorability of 95% is reached, but with individual pairs in some orderings still receiving zero restoration.

Thus, the main practical issue with "mass redial" as a proposal for path restoration is that one cannot give any precise or repeatable performance assurances—unless capacity is lavishly overprovisioned. It is categorically a "best-efforts only" form of restoration, amenable at best only to a statistical characterization of the overall and individual pair recovery levels. Individual customers could not be assured that they would not remain completely disconnected following the overall set of restoration attempts. It is therefore a source of confusion when some people say "path restoration" but really have in mind only mass greedy reattempting, or conversely, when someone assumes mass redial is the only embodiment of path restoration and based on this asserts that (because of the issues just portrayed) "path restoration is only a best-efforts approach." Both of these statements cause continuing misconceptions and are based on failure to appreciate that there are *true* path restoration protocols, and centrally coordinated control solutions, and related capacity and routing methodologies through which 100% guarantees of restorability *can be assured* and with a theoretical minimum amount of spare capacity. The latter type of scheme treated in the rest of this chapter.

6.1.4 Network Recovery From Node Loss

Because path restoration operates end-to-end for every disrupted path, a PR or SBPP mechanism can inherently react to arbitrary combinations of node and/or span failures. From the *mechanism's* viewpoint, there is really no distinction needed between a span failure, node failure, or multiple failure scenarios. They are all simply processes that "light up" a certain selection of O-D pairs that require simultaneous restoration. The mechanism will transparently kick in and operate in the same basic way whatever the cause of path failure. But this by itself does not imply that networks using PR or SBPP are fully restorable against any node failure as well as span failure. *That* is still a question of the capacity design to also support those failure scenarios.

For several reasons, however, we do not include node failure scenarios in our basic treatment of path restoration capacity design. In practice, node loss is vastly less frequent than cable damage. In addition, under a node failure demands that source or sink at the failure node are not restorable by any means other than physical repair. "Node restoration" is really a misnomer; it is only *transiting* demands that can be restored by network rerouting. Span and node failures are categorically different in that all services *can* be 100% protected by a rerouting response to cable failures, but *cannot be* for node failures. Thus, good site security, backup power, software reliability etc., are fundamentally more important and effective for defense against nodal failures than is reliance on network rerouting. The added spare capacity to support restoration of transiting flows through a failed node can, however, be built into the network design methods that follow by including node failures in the family of failure scenarios. Node failure is represented by asserting the simultaneous failure of all spans incident on the node as the failure scenario. The restorability-asserting constraints in the models to follow must then be defined with respect to the transiting demand at each node only, not all demand that traversed the failed spans.

An interesting general finding about node recovery under path restoration is, however, that it typically takes little or no additional spare capacity to support 100% restorability of transiting flows affected by any single *node* failure in a network designed only to withstand all single *span* failures. In five networks tested in [ShGr03](#), the intrinsic node recovery levels were almost 100% in most cases, none lower than 78%, and all could be made 100% recoverable against node losses with under 10% extra spare capacity, usually just 2 or 3%. This is surprising at first (because a node failure is "like multiple span failures") but it occurs because the "abandonment" of unrestorable paths that source/sink at the failed node significantly relieves the restoration effort. Many individual node failure scenarios actually require *less* spare capacity than is present in a design for all single-span failures. In light of the findings in [ShGr03](#), and what follows here on capacity design and the stub release concept, a simple class of network that would be virtually always node-recoverable without additional design considerations would be one in which stub release applies to the unrestorable source/sink demands upon a node failure, but where the initial spare capacity design is produced for span failure scenarios alone *without* the assumption of stub release.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

6.2 A Framework for Path Restoration Routing and Capacity Design

Let us now look at the rerouting that is required for path restoration (given the spare capacity). Once we appreciate what is involved in routing for path restoration we can look at the corresponding methods for capacity design to support true path restoration. A corresponding treatment for SBPP follows later. Our first step is to define and explain a set of elemental concepts and terminology to support all the subsequent topics in this chapter.

6.2.1 Specifying Failure Scenarios

In a path restoration context there are three aspects to specifying a failure scenario. One is to stipulate the set of end-node pairs that are affected, and the magnitude or number of demand units lost on each pair, and the actual failed span. In a sense as mentioned above, it does not matter to the restoration mechanism what fault actually occurred on the network—the set of end-node pairs that are simultaneously affected as the result is sufficient to define the failure scenario. To be complete, however, the actual source of the failure matters as well because this stipulates which network span(s) are not available for the corresponding restoration effort. Therefore let us define:

- D_z is the set of O-D pairs that have lost one or more units of demand upon failure scenario z . Individual O-D pairs of D_z are indexed by r .
- F is a set of predefined failure scenarios, specified in terms of failed spans, indexed z .
- X_z^r is the number of demand units (individual lightpaths for instance) lost by O-D pair, for failure scenario z .

The rerouting problem takes place in the context of one specific failure scenario at a time. However, the later capacity design problem has to simultaneously address all failure scenarios. Obviously a common set of failure scenarios to consider is all single-span failures $F = \{i:i \in S\}$. The more general case, however, allows F to include all single failures and selected multiple span failure scenarios to include consideration of known SRLG effects or node loss if desired. In what follows we will use z in formulations as the index of a generalized set of failure scenarios which could include all single-span failures, node failures, certain multiple span failures from SRLGs, etc. Later as our discussion and test cases focus specifically on the set of single-span failures, we revert for convenience to indexing failure scenarios by i , the index of the span presumed to have failed.

6.2.2 Specifying Network Structure, Capacity and Eligible Routes

To start, let us also reintroduce the following parameters and variables which keep the same meanings they had for the treatment of span restoration in [Chapter 5](#):

- D is the set of all demand quantities exchanged between O-D pairs, with index r , and element values d^r .
- S is the set of spans of the network.
- s_j is the spare capacity on span j .
- w_j is the working capacity on span j .

- M is a set of modular transmission capacities, index m .
- Z^m is the number of capacity units provided by a module of the m^{th} size.
- C_j^m is the cost of establishing a module of the m^{th} transmission capacity on span j .
- c_j^m is the cost of equipping a channel on span j in a system of the m^{th} modular capacity.

We assume that costs and capacities refer to bidirectional pairs of individual channels or systems in each case. Usually the length of span j is taken into account in generating these values and so span lengths do not appear directly in many formulations. This framework allows for the general but realistic aspect that buildable networks are inevitably modular in transmission capacities. For optical layer network

design C_j^m can be interpreted as the cost of establishing an additional fiber pair on span j and equipping it with basic infrastructure to

support an ultimate capacity of Z^m wavelengths, each of which will have an incremental cost c_j^m to turn up. Note that the per-channel cost may itself depend on the size of the modular system in which it is implemented, e.g., adding a channel in an 8-channel system will have a different cost in general than adding a channel to a 128-channel system. In many comparative studies, however, specific modular assumptions are not used in which case a single capacity cost coefficient c_j can be used to represent the average cost per capacity unit required or provided on span j . Aspects of the routing of working demands are characterized by the following:

- d^r is the number of demand units between end-node pair r .
- Q^r is the set of distinct eligible routes available to satisfy the demand between node pair r , index q .
- $g^{r,q}$ is an amount of working demand flow assigned to the q^{th} working route between node pair r .
- $\xi_j^{r,q}$ is a 1/0 indicator parameter that relates the eligible working routes to the graph topology (which itself never appears directly). The indicator is 1 if the q^{th} working route for demand pair r uses span j .

This structuring of the details for working capacity and routes allows that not all demand on a given O-D pair need follow the same route, i.e., total demand on a relation can be dispersed. In a joint design $g^{r,q}$ will be a variable, but is a parameter in a non-joint design. In a joint design Q^r will contain all eligible routes, but for generality we can also keep Q^r in our non-joint models where each Q^r set is understood to contain simply the one actual route taken for demands on relation r . Unlike span restoration, we always need the working path route information in path restoration, not just the w_j quantities, so that when a given span fails, we can generate the corresponding pair-wise damages, X_z^r , to the affected end-node pairs. In other words, we always need to know which specific paths cross each span, not just

how much working capacity to protect on the span. The process of generating X_z^r for each failure scenario can, however, be done with a preprocessing step before running a design solver.

Next, for each O-D pair, we need a description of the routes that could be used for restoration of demands for that relation. There are two systems of specifying routes that have generally been used. In the first, the failure-specific nature of the restoration response is explicitly present in that there is a set of eligible routes for each relation, for each specific failure scenario:

- P_z^r is the set of eligible restoration routes available to O-D pair r for its restoration under failure scenario z .
- $\delta_{z,j}^{r,p}$ is a boolean parameter that is 1 if span j is in the p^{th} eligible route for restoration of O-D pair r in the event of failure scenario z .
- $f_z^{r,p}$

- is an assignment of restoration flow to the p^{th} route available for restoration of O-D pair r for failure scenario z .

This is conceptually simplifying but it requires a more elaborate preprocessing program and expands the size of the problem files that need to be handed to a solver. A more efficient approach is to specify one master set of eligible restoration routes, generated on the whole unfailed topology, and allow the capacity design model to effectively exclude routes that become ineligible by virtue of containing a failed span in each failure scenario. With this approach we will have the following system of notation:

- P^r is the master set of routes eligible for use in restoration of O-D pair r .
- $\delta_j^{r,p}$ is 1 if span j is in the p^{th} eligible route for O-D pair r .
- $f_z^{r,p}$ is unchanged. It is inherently always a failure-specific quantity.

In the latter system of representing routes, the master route set is effectively sifted for the subset of routes that remain feasible under each failure scenario. That is:

Equation 6.1

$$P_z^r = P^r - \{p | (\delta_j^{r,p} = 1 \cap j \in F(z))\}$$

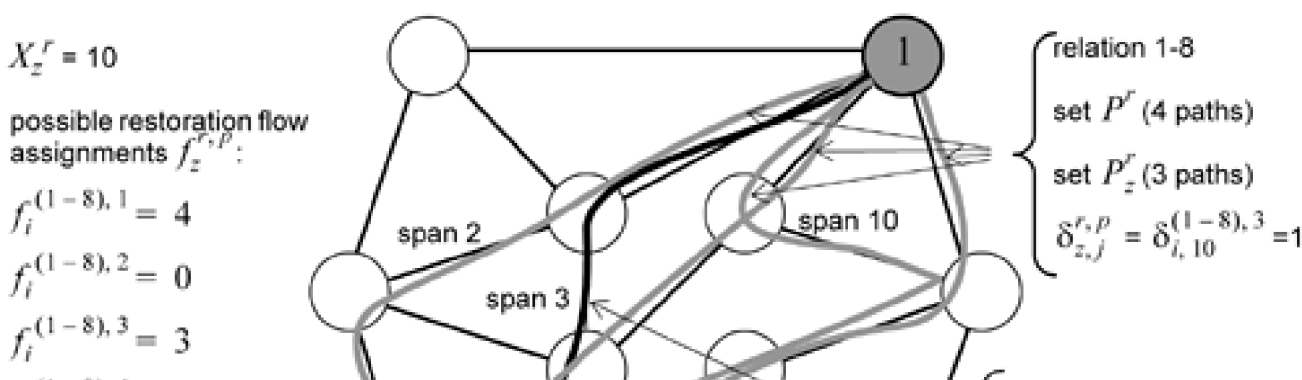
where $\{ \}$ denotes a set, p indexes routes in a route set and $F(z)$ is the set of span failures that constitute the z^{th} failure scenario. Equation 6.1 is a more compact way of specifying all the failure-specific eligible routesets that can be handled directly by a high level tool such as AMPL for setting up the problem tableaux.

Figure 6-6 helps summarize the overall context and relationships between entities mentioned above in the path restoration environment. Figure 6-6 allows us to make some example statements about the path restoration framework: Node pair or "relation" = (1-8) is shown exchanging demands over one working route ($q = 1$). (Other working flow for relation (1-8) may take different routes $q = 2, 3, \dots$) but is not affected by the failure scenario shown and so, for simplicity, is not portrayed.) Ten demand units for this relation cross span i and are affected by failure scenario z which comprises the single failure of span i . The damage to relation (1-8) is $X_z^{(1-8)} = 10$. There are four eligible restoration routes for relation (1-8) in its master route set but only three of these remain feasible under failure scenario z . Span 10

is part of the third one of these routes as indicated by $\delta_{i,10}^{(1-8),3} = 1$. Span 3 is part of the first working route (the only one shown)

as indicated by $\delta_{i,3}^{(1-8),1} = 1$. A possible assignment of restoration flows to surviving eligible routes is (4,0,3) to routes (1...4) respectively.

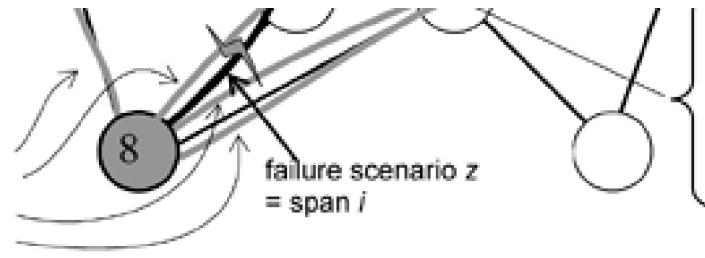
Figure 6-6. Context and example terminology for path restoration routing and capacity design.



$$f_i^{(1-\delta), q} = 4$$

eligible routes in P^r :

- $p = 1$
- $p = 2$
- $p = 3$
- $p = 4$



working route $q = 1$

working flow, $g^{r,q} = 10$

$$\zeta_j^{r,q} = \zeta_3^{(1-8), 1} = 1$$

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.3 The Path Restoration Rerouting Problem

Before treating capacity design, it is useful to first consider just the routing problem associated with path restoration in an environment of given capacities. This is a specific variant of multi-commodity maximum flow routing in which there are maximums on the flow for each commodity and a capacitated graph of spare capacities. An initial statement of the path restoration *routing* problem is, for a given failure scenario z :

PRR-1

Maximize

Equation 6.2

$$\sum_{r \in D_z} \sum_{p \in P'_z} f_z^{r,p}$$

subject to:

- limited requirements on each commodity:

Equation 6.3

$$\sum_{p \in P'_z} f_z^{r,p} \leq X'_z \quad \forall r \in D_z$$

- limited capacity on network spans:

Equation 6.4

$$\sum_{r \in D_z} \sum_{p \in P'_z} \delta_{z,j}^{r,p} \cdot f_z^{r,p} \leq s_j \quad \forall j \in S$$

Equation 6.5

$$f_z^{r,p} \geq 0, \text{ integer} \quad \forall (r, p)$$

The objective is to maximize the total number of restoration paths provided for the scenario as a whole. This is effected by finding assignments of restoration flow to the surviving eligible routes for restoration of each pair of affected end-nodes with upper bounds so that no O-D pair getting more restoration than it needs ([Equation 6.3](#)) and that all simultaneous flow assignments not exceed the spare capacity of any span ([Equation 6.4](#)). Note that the eligible route set for each relation considered in [Equation 6.3](#) and [Equation 6.4](#) is

P_z^f which is the failure specific subset of surviving eligible routes, filtered from the master route set P^f by [Equation 6.1](#).

In PRR-1 the aim is to achieve as high an overall restoration ratio as possible but it has no explicit concern about fairness to each O-D pair individually in the result. This is not an issue if the capacity design supports 100% restorability for the failure scenario z. The outcome is then inherently fair because it is complete: every affected O-D pair will reach 100% restorability (or the design target level) for each pair. In other words, with adequate capacity, [Equation 6.3](#) will produce the binding constraints on the objective, rather than [Equation 6.4](#). Thus PRR-1 rerouting coupled with an appropriate capacity will result in fairness without any explicit measures to that effect. This will most often be the intent of the capacity design process regarding all single-span failure scenarios. But more generally, if we do not presume 100% restorability (say, in multiple failures or a node loss situation) then we can expect to exhaust the available spare capacity. In this case notions of fairness or assured minimums come into play in partial recovery. A simple assurance of not leaving any O-D pair wholly disconnected can be effected (if feasible under the spare capacity present ([Equation 6.4](#))) by adding an additional constraint family to PRR-1:

Equation 6.6

$$\sum_{p \in P_z^f} f_z^{r,p} \geq 1 \quad \forall r \in D_z.$$

This will assure that all affected O-D pairs enjoy at least one restoration path so they are not disconnected, if this is feasible. To go further in asserting fairness on the individual recovery ratios, however, we can change the formulation to:

PRR-2

Maximize

Equation 6.7

$$\lambda$$

subject to:

Equation 6.8

$$\sum_{p \in P_z^f} f_z^{r,p} \geq \lambda \cdot X_z^r \quad \forall r \in D_z$$

Equation 6.9

$$0 \leq \lambda \leq 1$$

- plus [Equation 6.4](#) and [Equation 6.5](#).

PRR-2 will bring all affected O-D pairs to the highest minimum fractional level of restoration, λ , that is feasible. This does not mean all relations will receive the *same* level of restoration, but that a higher level for some node pairs might be sacrificed to bring up the lowest of the restoration levels experienced on other relations. Fairness is provided only in the sense that the compromises will be made so that lowest restoration level on any node pair will be as high as it can be. The sense of fairness is put above the extent of the total bulk recovery level of the failure scenario as a whole.

Neither PRR-1 nor PRR-2 seems wholly satisfactory in cases of incomplete restoration. It seems more reasonable in practice that if all affected O-D pairs cannot be 100% restored, then we would hope for a high overall recovery level with as much fairness as possible in the distribution of recovery levels. This will allow fortuitous O-D pairs to receive a higher recovery level than others where it is not to the significant detriment of any other pair. This leads to a bicriterion expression of the path restoration rerouting problem, as follows:

PRR-3

Maximize

Equation 6.10

$$\left((1 - \alpha) \cdot \sum_{r \in D_z} \sum_{p \in P_z^r} f_z^{r,p} + \alpha \cdot \sum_{r \in D_z} \lambda_r \right)$$

subject to:

Equation 6.11

$$\sum_{p \in P_z^r} f_z^{r,p} \geq \lambda \cdot X_z^r \quad \forall r \in D_z$$

Equation 6.12

$$\lambda_r \leq 1 \quad \forall r \in D_z$$

optional:

Equation 6.13

$$\lambda_r \geq \lambda_{min} \quad \forall r \in D_z$$

- plus [Equation 6.4](#) and [Equation 6.5](#).

PRR-3 introduces a fractional recovery variable l_r for each O-D pair affected in the failure scenario and it (optionally) lower-bounds each individual recovery ratio to be above some minimum requirements $l_{min} \geq 0$. This is the only direct assertion of a notion of strict fairness in the sense of attempting to assure a minimum for each O-D pair equally. Depending on the l_{min} that is attempted the problem could be infeasible in the given capacity. Assuming l_{min} can be satisfied, however, the objective is no longer concerned with restoring all O-D pairs to the highest worst-case level, which (in PRR-2) could be at the expense of the total amount of demand restored. Instead, we have a bicriterion expression of the desirability of maximizing a combined measure of both the bulk recovery level on the scenario, tempered by the desirability of also raising as many individual O-D pair recovery levels to high fractional levels. Note now that the restoration ratio for relation is associated with own variable l_r , the restorability constraints ([Equation 6.11](#)) become equalities (whereas in PRR-2, they are \geq inequalities).

To see how PRR-3 works, consider a conceptual case of a failure scenario i where only two O-D pairs $r = 1, 2$ are affected, and for argument's sake let one pair ($r = 1$) exchange 1000 demand units and the other 10. Then, if $\alpha = 0$, the solution is totally biased toward bulk recovery maximization which could correspond to realizing 800 units of restoration flow for $r = 1$ and zero for $r = 2$ which illustrates what is objectionable with simple maximization of the overall restoration ratio as in PRR-2. At the other extreme, at $\alpha = 1$, it is the sum of fractional recovery levels that matters. It is easy to see the plausible change in the solution: full recovery of the 10 units of affected demand for $r = 2$ is probably quite easy to "squeeze into" the environment of much greater capacities generally available to deal with the $r = 1$ flow. Then, even if (through mutual capacity effects) the realization of 10 paths for $r = 2$ was at the expense of, say, 20 paths for $r = 1$, the objective value becomes $1 + (800 - 20) / 1000 = 1.78$. Thus, maximizing the sum of the individual recovery ratios makes an intuitively more reasonable compromise between satisfying the 1000 unit flow and the 10 unit flow. In contrast, the same measure of the result under $\alpha = 0$ would have been $800 / 1000 = 0.8$. For each intermediate value of α there is a different trade-off between the goal of bulk recovery and high fractional recovery for individual O-D pairs. Some particular α could be adopted as effecting the most acceptable policy for overall recovery plans.

A slight variant on PRR-2 can be useful in cases where $l = 1$ is not achievable, to diagnose the individual relations that were not fully routable by giving each relation an individual l_r variable, just as in PRR-3, but in addition to the overall l in PRR-2. In later heuristics for path-restorable design ([Section 6.14](#) and onward) we will use the resulting $l_r < 1$ information to identify the specific unrestorable demand pairs, r (under a given failure scenario) for which repairs of a spare capacity design are needed.

PRR-2a

Maximize

Equation 6.14

$$\lambda$$

subject to:

Equation 6.15

$$\sum_{p \in P_z} f_z^{r,p} = \lambda_r \cdot X_z^r \quad \forall r \in D_z$$

Equation 6.16

$$\lambda_r \geq \lambda \quad \forall r \in D_z$$

Equation 6.17

$$0 \leq \lambda \leq 1; 0 \leq \lambda_r \leq 1 \quad \forall r \in D_z$$

- plus [Equation 6.4](#) and [Equation 6.5](#).

PRR-2a has the same net effect as PRR-2 in terms of trying to bring every affected relation to 100% restorability or to the highest minimum level that is possible. In cases where $I < 1$ (the network is not restorable as a whole) many individual relations may still be fully restorable however (and we have $I_r = 1$ for them). But the cases of $I_r < 1$ allow PRR2a to give us an explicit identification of the relations for which restorability is lacking.

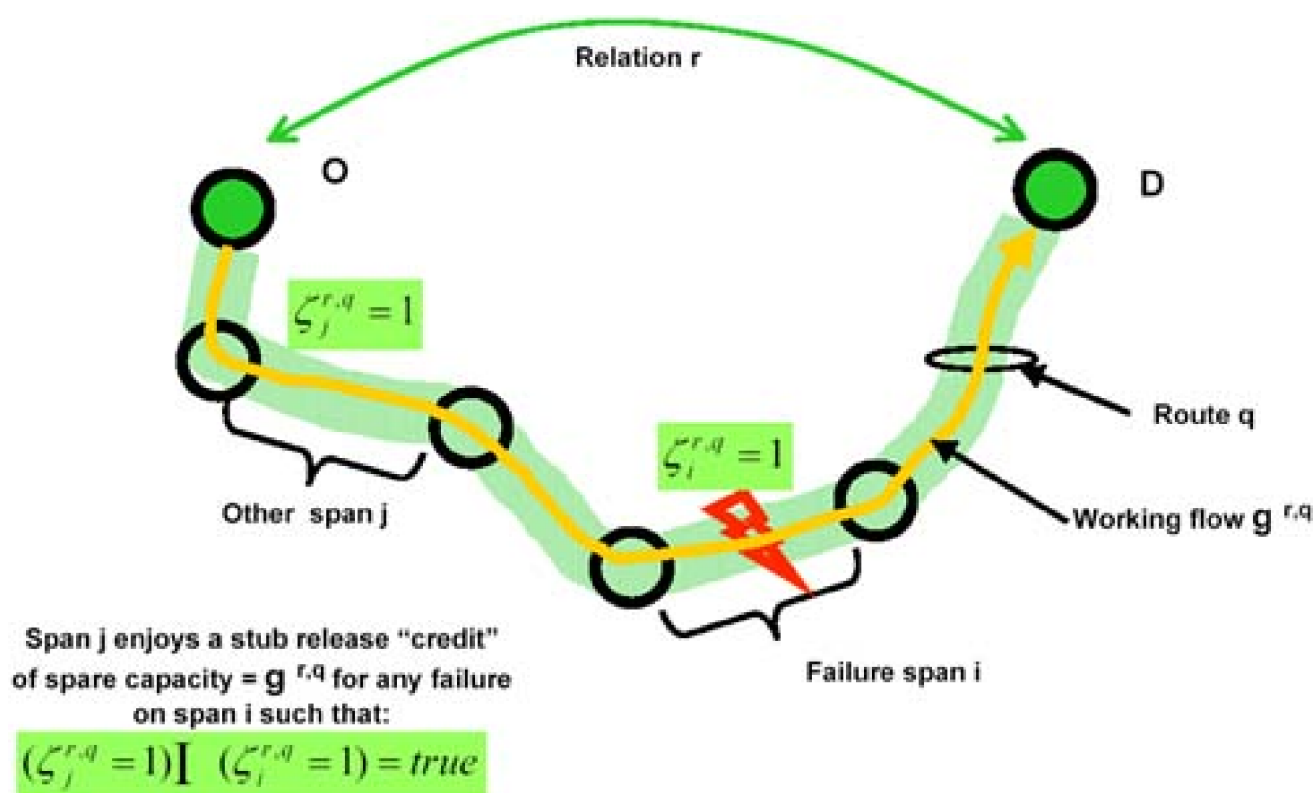
[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.4 Concepts of Stub Release and Stub Reuse in Path Restoration

Having previously introduced the concepts of modularity and joint design, the main additional concept we need to develop the capacity design model for path restoration is that of *stub release*. In a path-restorable network it is possible to release the surviving upstream and downstream portions of a failed working path and make the freed capacity available to the restoration process. This option is called *stub release*. The concept and a mathematical criterion for identifying stub release capacity is illustrated in [figure 6-7](#). If a path fails on span i , but its pre-failure route crosses another span j , then the working capacity that the failed path used on span j can be viewed as being up for reallocation to some new use. An obvious use is to consider it as equivalent to spare capacity for the current restoration problem. The alternative is to leave the stub capacity in place as unused working capacity, reserved for the return of the normal signal path after physical repair of the failure. Such a return to normal routing after repair is usually called the "reversion" process.

Figure 6-7. Concept of "stub release" in path restoration.



Stub release is the main sense in which path restoration provides a failure-specific response. For each failure scenario there is a different environment of stub-release capacity to be exploited. The question of stub release does not arise with span restoration because the reconfiguration that occurs in the latter is around the failed span itself. In other words, the stub capacity leading up to and away from the break (in each direction) is inherently always reused as part of the end-to-end solution. While stub release makes restoration more capacity efficient, there are concerns that stub release causes operational complications.

One concern is about the real-time signaling to effect stub release if it is difficult to pinpoint the failure location, particularly if transparent OXCs are assumed. In that case, how do we rapidly identify the break and, hence, the surviving portions of the broken paths? In a completely optical network, rapid fault location may be difficult so schemes like SBPP—which do not involve stub release—may be the only available option. However, in an opaque network the OXC nodes adjacent to the failure see the failure immediately and substitute an Alarm Indication Signal into the continuing signal path. The presence of AIS signal ([Section 3.4.1](#)) on the stub can itself be the indication that the capacity is available for the restoration effort. In that context stub release is an automatic outcome of the failure itself and does not require any explicit signaling to effect stub release. This model is especially suitable if digital wrapper is implemented on lightpaths and/or

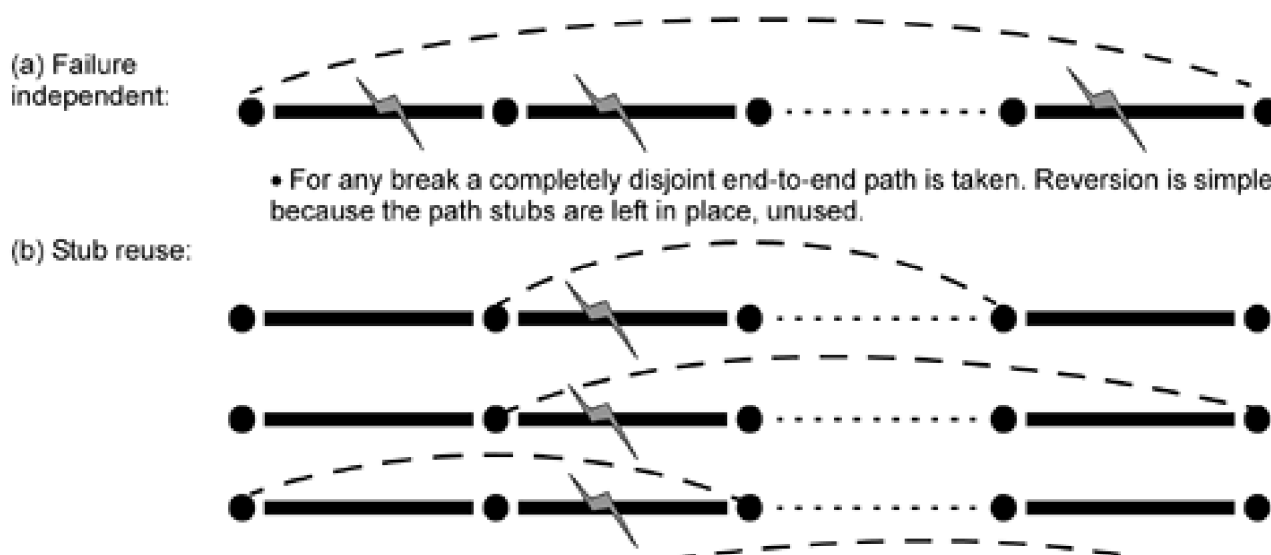
LMP is implemented to monitor each span. If some nodes are fully transparent OXC, then fault sectionalization for stub release simply occurs at the ends of each transparent segment of signal path, between OXCs that are monitoring for signal loss. In that case a set of path *segments* define the failure scenarios instead of single spans. But stub release can still occur automatically from path end nodes up to the boundary of the transparent island in which the failure occurred, even if not localized immediately to a single span.

The second perceived complication with stub release occurs in reversion following physical repair. The issue is basically how to unravel the restored state and put everything back to the prefailure state. Unlike the AIS signaling to effect stub release, reversion is at least not a problem that has to be solved urgently in real-time. Time can be taken to confirm the restored state of actual paths and explicitly compute a series of centrally commanded steps for an appropriate reversion sequence. An aim in solving the reversion problem would be to put no more than one additional "hit" on each service path (the same as under SBPP). Another goal would be to identify any paths in the restored state that are essentially as efficiently routed now as in the prefailure state and leave such paths as is. In general, however, reversion can always be handled one path at a time by identifying and cross-connecting a new path on the shortest route to revert to, putting up a head-end bridge at both ends of the path, and triggering the tail-end switch. At that point all capacity used in the restoration path is released and reversion for one path is complete.

An implication of path restoration *without* stub release is that restoration paths are prohibited from even retaining their own signal path up to and away from the actual break. Additionally, no other restoration path can use such stub capacity. SBPP has these properties but path restoration without stub release is not identical to SBPP because in all cases restoration paths in PR can still make use of explicit *spare* capacity on *any* span of the network, including spans of its own stub route. On the other hand, stub release as currently described allows that restoration paths for relation *X* may wind up using stub release capacity from a failed path on relation *Y*. It is this post-failure cross-entanglement of paths that seems to be the main concern about reversion complexity. This suggests a policy in between full stub release and no stub release. Stub *reuse* would allow a restoration path to reuse any part of its own prefailure working path, but stub capacity would not be available in the more general sense of full stub release. This would greatly simplify reversion by eliminating cross entanglements while still allowing the efficiency of a failure-specific restoration response on each path. During reversion any reused path portions are just left in place and only that segment of the restoration path that departed from the original working path is switched back to its prefailure route.

[Figure 6-8](#) portrays the differences between stub reuse, release and no use of stub release. Heavy black lines show prefailure paths with the light dashed opening just suggesting an arbitrary number of additional spans in the working path. Failed spans are shown with the bolt symbol. The medium dashed lines show examples of the logical relationship between prefailure and restored states. In failure-independent schemes such as SBPP, a completely separate end-to-end route is taken regardless of the location of the failure on the path. The restoration path is forced completely off of the prefailure route. Path restoration without stub release will act this way with regards to its own stub capacity, but may nonetheless still appear to reuse its stub route in part if it is using spare capacity. Any use of true spare capacity does not complicate the reversion process because spare capacity starts (prefailure) and ends (after reversion) in an unused idle state. Under stub *reuse* the path restoration process is at liberty to decide what *segment* of the failed path to replace. This is like a form of SLSP (segmented protection) but where the segment to be replaced can vary from span restoration to fully disjoint path replacement in a failure specific way for best overall capacity efficiency. With full stub release in [Figure 6-8\(c\)](#) we see how it is possible for two restoration paths arising from the same failure to wind up using each other's prefailure working capacity. It is this that most complicates the reversion process.

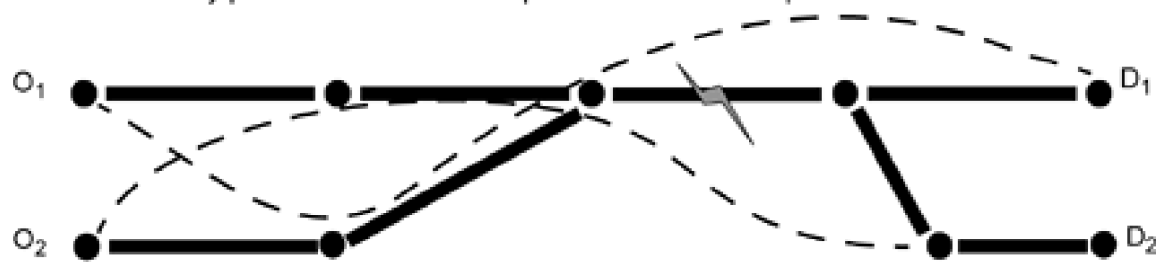
Figure 6-8. Concepts of failure-independent path restoration and failure-specific stub reuse and stub release. (Solid = prefailure path(s), dashed = path in restored state.)





• Any surviving part of the prefailure path may be retained before or after the break by paths on the same O-D pair. Reversion is simple.

(c) Stub release:



• Surviving stub capacity is completely released for use as spare capacity in restoration. Example shows two restoration paths exploiting each other's stub release leaving an "entangled" state for reversion. Reversion is more complex.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.5 Lower Bounds on Redundancy

As a further preliminary to understand path-restorable networks, we can develop a lower bound on the redundancy that corresponds to the $1/(d-1)$ bound for a span-restorable network.^[1] The result reveals an underlying principle for working path routing in path-restorable networks, that of demand dispersion. Let the standard definitions above apply, plus:

^[1]This development is based on part of the Ph.D. work by R.R. Iraschko [\[ras96\]](#).

- $\bar{d} = 2S/N$ is the average nodal degree of nodes in a network
- $|D_i|$ is the total number of demand pairs affected by span cut i
- $CapS$ is the total spare capacity in a network
- $CapW$ is the total working capacity in a network

The average working capacity per span of the network is, therefore:

Equation 6.18

$$\bar{w} = \frac{1}{S} \cdot \sum_{i \in S} w_i.$$

and the average total amount of demand per O-D pair (or "relation") severed by a span cut is:

Equation 6.19

$$\bar{w}_r = \sum_{i \in S} w_i / \sum_{i \in S} |D_i|.$$

For purposes of a lower bounding type of assessment only let us assume:

- a. Restoration is limited by the spare capacity incident on the end-nodes of a failure.
- b. Each span carries the same capacity w .
- c. Each relation has the same demand w_r affected by a span cut.
- d. The failed span is elsewhere in the network, not incident to the end node under consideration, but has failed w_r paths for which the node is an end-node.

Under these conditions, egress for restoration rerouting of failed capacity from the end nodes of an affected relation is possible only if the total spare capacity on the spans terminated at these nodes is enough to accommodate the total demand lost on that relation, w_r .

We have already seen that in a span-restorable network the combined spare capacity of the $\bar{d} - 1$ surviving spans terminated at a node must exceed the working capacity of any incident cut span. Transferring this notion to end nodes under path restoration, we recognize that the source of physical failure will *not* on average be adjacent to the end nodes of the failed paths. This suggests all \bar{d} spans can bear the needed spare capacity for restoration egress. However, in a complete design, every node will still have to bear the spare capacity required under the scenario where a span failure is incident upon it. Clearly the latter cases will tend to be more binding constraints, so even under path restoration we do not escape a requirement that the sum of the spare capacity on the $\bar{d} - 1$ unaffected spans be adequate for restoration of each other incident span. A greater difference from span restoration is, however, that many end-node pairs are affected and each end-node pair is concerned only with restoration of that portion of w that pertains to them, i.e., w_r . In this case each span at a path end-node must have an average of $w_r(\bar{d} - 1)$ spare capacity. The total spare capacity in the network is then $\bar{w}_r \cdot |S|(\bar{d} - 1)$. Thus, given that the total working capacity is $S \cdot \bar{w}$, a lower limit on the redundancy of a path-restorable network is:

Equation 6.20

$$\sum_s / \sum w \geq \frac{\bar{w}_r \cdot S}{(\bar{d} - 1) \cdot (S \cdot \bar{w})} = \frac{\bar{w}_r}{(\bar{d} - 1) \cdot \bar{w}} \quad (\text{no stub release}).$$

Note that the span-restorable lower bound is obtained from [Equation 6.20](#) as the special case of a single relation being affected by the span cut, in which $\bar{w} = \bar{w}_r$. This expression also tells us that the key difference (at least in these bounding conditions) between span and path-restorable networks is in the way \bar{w}_r / \bar{w} behaves. This is where the principle of dispersive (but conservative) routing comes in because the ratio reflects how many paths the average relation has on a span relative to the span's working total. Driving down \bar{w}_r or driving \bar{w} up would both numerically improve redundancy. Indirect routing schemes will increase the value of \bar{w} and reduce \bar{w}_r , while shortest path routing will minimize the value of \bar{w} . However, as a bound on redundancy it is not meaningful that we would deliberately use dispersive routing just to increase \bar{w} . We would be reducing redundancy but increasing total capacity. Notionally, therefore, we want to read [Equation 6.20](#) with a frame of mind where \bar{w} is essentially fixed near the value corresponding to shortest path routing, as in span restoration, and concentrate on how routing can influence \bar{w}_r . (Recall the concept of *standard* redundancy: [Section 1.5.3](#).) In this light the prescription is for aggressive dispersion of the demands on each node pair wherever multiple nearly-equal shortest routes are available, but not where it involves any route significantly longer than the shortest route. The aim is to spread the flow on each relation over many diverse routes so that a smaller amount of demand is lost by many relations on each failure, rather than a large amount being lost by a few relations. While this is an interesting insight, it must also be observed that [Equation 6.20](#) is not a very tight bound, because \bar{w}_r / \bar{w} itself may be arbitrarily low.

On the other hand, the functional form of [Equation 6.20](#) is quite descriptive of experimental results with path-restorable designs if we recognize that while \bar{d} appears explicitly only once in [Equation 6.20](#), there is a secondary dependence of w_r itself on \bar{d} which is not explicitly portrayed. This second effect is that as \bar{d} increases the number of spans present to bear the same total demand goes up linearly. \bar{w} will therefore tend to decrease inversely as \bar{d} increases, reflecting more network spans as a whole to bear the same total demand. On the other hand \bar{w}_r will tend to be less responsive to increases in \bar{d} because most additional spans added to the network will not have any affect on the shortest paths for any one relation r . In other words, from the standpoint of a specific relation, most spans added to the network will not affect its own demands, but every added span affects \bar{w} of the network as a whole. Thus, we infer further that the term \bar{w}_r / \bar{w} in [Equation 6.20](#) is itself $\sim O(1/\bar{d})$. We expect overall therefore that path-restorable network redundancy will respond to increases in network average nodal degree more strongly, approximately as $\sim O([1/\bar{d}]^2)$, than in a span-restorable network—where we expect the redundancy to improve as $\sim O(1/\bar{d})$. In the study results of [DoGr01](#), this effect is quite evident in the comparative curves for span and path-restorable designs plotted against nodal degree.

6.6 Master Formulation for Path Restoration Capacity Design

Having previously introduced the concepts of modularity and joint design and added the concept of stub release, we can now present a formulation for path-restorable capacity design that encompasses all such considerations. This is the path restoration equivalent of the MJCA model for span restoration. From this master formulation it is easy to identify the unneeded parts to obtain several simpler special case formulations. The master formulation is:

path-MJCA

min

Equation 6.21

$$\sum_{j \in S} \sum_{m \in M} c_j^m \cdot n_j^m$$

subject to:

1. All demands are routed for each O-D pair:

Equation 6.22

$$\sum_{q \in Q} g^{r,q} = d^r \quad \forall r \in D$$

2. Working capacity is sufficient to support the routing of working paths:

Equation 6.23

$$w_j = \sum_{r \in D} \sum_{q \in Q} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S$$

- 3.

The total demand lost by relation r from the z^{th} failure scenario:

Equation 6.24

$$\sum_{a \in Q} \left(\bigcup_{i \in F(z)} \zeta_i^{r,q} \right) \cdot g^{r,q} = X_z^r \quad \forall r \in D \quad \forall z \in F$$

T - E

4. Restoration flow meets target restoration levels for each demand pair r :

Equation 6.25

$$\sum_{p \in P'_z} f_z^{r,p} = X'_z \quad \forall r \in D \quad \forall z \in F$$

5. Spare capacity is sufficient to support restoration flows:

Equation 6.26

$$\sum_{r \in D} \sum_{p \in P'_z} \delta_{z,j}^{r,p} \cdot f_z^{r,p} - s_{z,j}^0 \leq s_j \quad \forall j \in S \quad \forall z \in F$$

6. Definition of $s_{z,j}^0$ for stub release case:

Equation 6.27

$$s_{z,j}^0 = \sum_{r \in D} \sum_{q \in Q'} \left(\bigcup_{i \in F(z)} \zeta_i^{r,q} \right) \cdot \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S \setminus j \in F(z) \quad \forall z \in F$$

7. Modular capacity:

Equation 6.28

$$s_j + w_j \leq \sum_{m \in M} n_j^m \cdot z^m \quad \forall j \in S$$

8. The modular capacity decision variables, n_j^m , restoration flows, $f_z^{r,p}$, working flows, $g^{r,q}$, spare capacities, s_j , and working capacities, w_j , are non-negative integers.

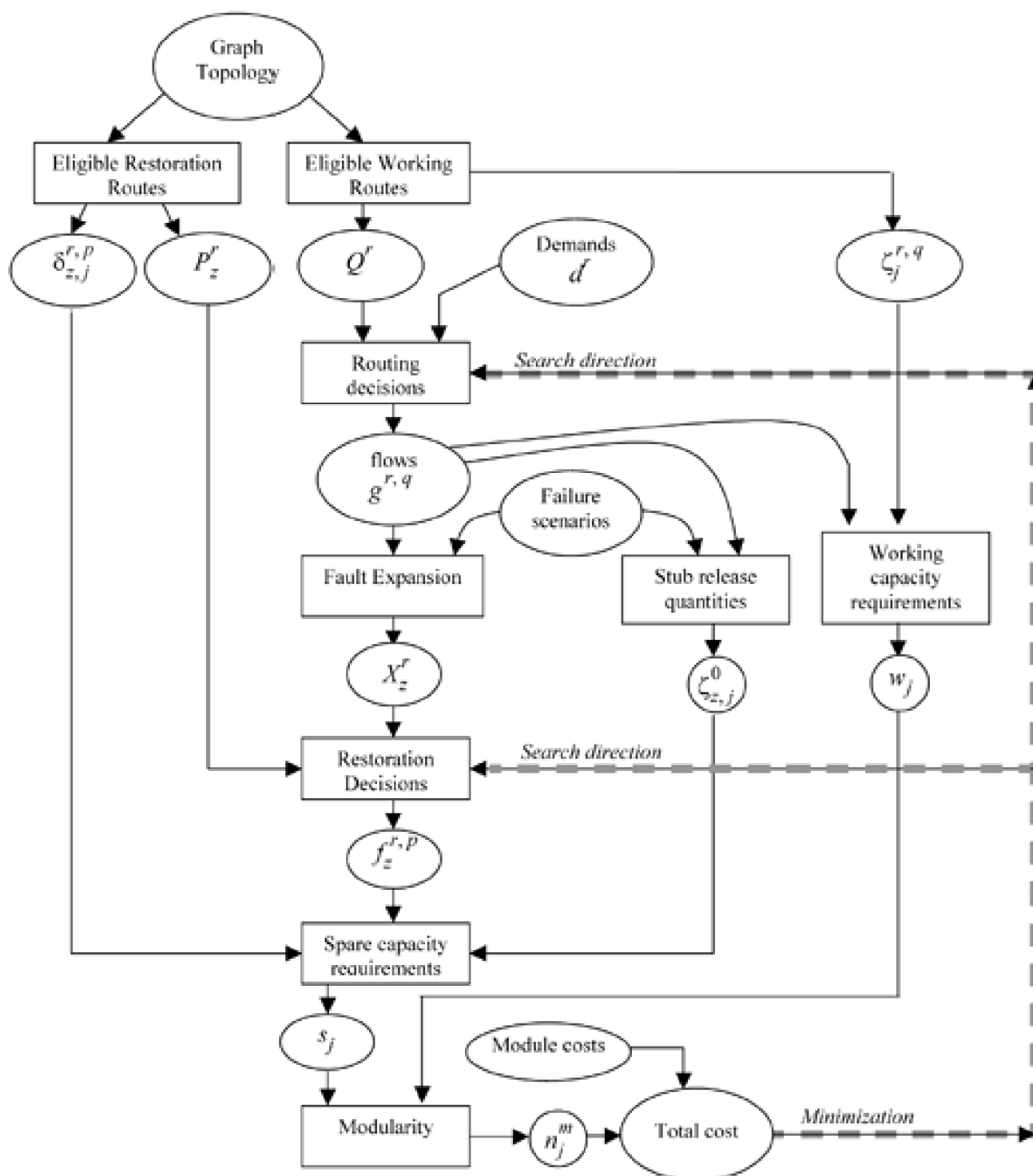
Note that by substitution of [Equation 6.25](#) into [Equation 6.24](#), and [Equation 6.27](#) into [Equation 6.26](#), the intermediate variables X'_z and $s_{z,j}^0$ (respectively) can be eliminated. Similarly, w_j can even be eliminated as an explicit variable, by substituting [Equation 6.23](#) into [Equation 6.28](#). For ease of conceptual understanding we will carry them forward. The properties of this most general of design models for path restoration are:

- Jointly optimized routing of working flows
- Demand bundle splitting
- Capacity is modular

- Stub release is included
- Generalized failure scenarios that may include multi-span failures

Let us comment on how each of these attributes is reflected or implemented in *path-MJCA* and some of their implications and point out how the corresponding feature could be deleted or refined to represent different design intents. Because *path-MJCA* serves as a master model from which many other models can be derived by extension or reduction, we also provide [Figure 6-9](#) to illustrate the overall structure of this class of problem.

Figure 6-9. Structure of the *path-MJCA* master problem.



Jointly Optimized Routing of Working Flows

[Equation 6.22](#) allows the formulation to decide upon the assignment of working demand flow to the set of eligible routes for routing of each demand pair. [Equation 6.23](#) generates the corresponding working capacity on spans. If joint optimization is not the intent, both of these constraint systems are dropped and w_j quantities can be provided as an input from a prior routing process such as shortest path routing. Note, however, that even in the non-joint case, the actual route taken by each demand still needs to be provided so that [Equation 6.26](#) can identify the set of end node pairs affected by each specific span failure. For this reason a convenient technique for a non-joint solution is to use the joint model but only represent a single route (usually the shortest path route) in each Q^r set.

Demand Bundle Splitting

By bundle splitting we mean that the general model permits the total demand on each relation to be spread over several different working routes. Nothing implies the splitting of individual demand units but the structure of [Equation 6.22](#) and [Equation 6.23](#) allows that a total demand of 10 units, say, could be routed on as many as ten distinct working routes between O and D, or all on one route. To design under a restriction that only one route may be chosen for all demand flow on each relation, and that such routes are pre-determined, one would use the same technique just mentioned above, of retaining [Equation 6.22](#) and [Equation 6.23](#) in form but providing Q^r sets that contain only the one intended route for each O-D pair. On the other hand, if the intent is for all demand for a relation to follow the same route, but to allow that route choice to be jointly optimized by the formulation, then full Q^r sets of eligible working routes are provided and [Equation 6.22](#) and [Equation 6.23](#) change, respectively, to become:

- One route is chosen for each O-D pair:

Equation 6.29

$$\sum_{q \in Q^r} \varphi^{r,q} = 1 \quad \forall r \in D$$

- Working capacity is sufficient to support the routing of working paths:

Equation 6.30

$$w_j = \sum_{r \in D} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot d^r \cdot \varphi^{r,q} \quad \forall j \in S$$

where $\varphi^{r,q}$ is a binary decision variable equal to one if the q^{th} eligible route for relation r is chosen, and zero otherwise. In this case the working flow assignment variables $g^{r,q}$ are no longer needed because the full d^r working flow for relation r will always be assigned to the one chosen route. Similarly, in either the joint or non-joint demand bundled cases, d^r replaces $g^{r,q}$ in [Equation 6.24](#) and [Equation 6.26](#).

Modular Capacity

Modularity of capacity is represented by stating the objective function in terms of the cost sum of all module provisioning decisions,

$$H_j^m$$

, and by [Equation 6.28](#) which links all other decisions about routing, restoration and logical capacities under a common modular capacity statement for each span. If a non-modular but integer-unit capacity design model is intended, the objective function changes to become the simple cost-weighted sum of integer total capacity (both working and spare, if a joint model) or simply spare capacity, if a non-joint model is intended, and [Equation 6.28](#) is deleted. A further side effect of dropping [Equation 6.28](#) is that in the case of a non-joint non-modular design model (e.g., the "path" equivalent to SCA), it is also not necessary to explicitly form the w_j variables, so the system of equalities in [Equation 6.23](#) can also be deleted.

Stub Release

Stub release is expressed by definition of the failure-specific variables $s_{z,j}^0$ in [Equation 6.27](#). The right hand side of [Equation 6.27](#) works by trapping working capacity on a span j that is upstream or downstream on the path of a failed span i and representing it as a credit (in [Equation 6.26](#)) against any spare capacity needed on the span j in the current failure scenario. Because s_j is elsewhere bounded to being non-negative, and because the form of [Equation 6.26](#) requires that s_j is the maximum of all the scenario-specific requirements for restoration flow, stub release can never lead to negativity in the s_j values. If stub release is not intended, [Equation 6.27](#) is not implemented

and $s_{z,j}^0$ variables are deleted from [Equation 6.26](#). Without stub release, restoration paths are often implicitly forced to adopt routes that are fully disjoint from their working paths because they cannot reuse or retain the surviving portion of their own pre-failure paths. They may however use explicit spare capacity if available on the same route as their own stub paths. If the end-to-end rerouting of the given path leaves one or more non-failed spans of its pre-failure route unused, however, the corresponding working capacity units on those spans cannot be reallocated to the restoration routing of any other affected path, whether on the same O-D pair or not.

Stub Reuse

As discussed in [Section 6.4](#), stub release capacity becomes available for the restoration rerouting of any demand affected in the same failure scenario, and it is this generality that leads to the possible "entanglement" illustrated in [Figure 6-8\(c\)](#). The difference with stub reuse is that demands on a given O-D pair would be allowed only to use their own stub paths or, equivalently, only stub release capacity arising from failure of other working paths on the same route for that O-D pair. To model stub reuse, the dimensionality of the prior expressions for the stub release quantities ([Equation 6.27](#)) and the spare capacity requirements ([Equation 6.26](#)) have to be increased to be relation-specific as well as failure-specific:

- Definition of $s_{z,j,r}^0$ for stub reuse:

Equation 6.31

$$s_{z,j,r}^0 = \sum_{q \in Q^r} \left(\bigcup_{i \in F(z)} \zeta_i^{r,q} \right) \cdot \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S | j \in F(z) \quad \forall z \in F \quad \forall r \in D_z$$

- Spare capacity to support restoration flows with stub reuse allowed:

Equation 6.32

$$\sum_{r \in D_z} \left[\sum_{p \in P_z^r} \delta_{z,j}^{r,p} \cdot f_z^{r,p} - s_{z,j,r}^0 \right] \leq s_j \quad \forall j \in S \quad \forall z \in F$$

The main difference is that now there is a stub reuse quantity for each relation, on each span, under each failure scenario. Under stub release there is only one bulk total stub release quantity on each span under each failure. Moreover, the stub reuse "credit" values can now only be applied to reduce the amount of spare capacity needed for restoration of demands for the same relation. This is why, in [Equation 6.32](#), the subtraction in inside brackets—making the difference in required spare capacity to stub reuse capacity specific to each relation—before summing over all affected relations, whereas in [Equation 6.26](#), the stub release "credit" applies interchangeably over any demand.

Stub reuse accordingly presents a significant expansion of the number of problem variables. One way to avoid this added complexity but approximate the design with stub reuse characteristics would be to produce the capacity design for the no stub release case but then operate within it in a stub reuse manner to keep reversion simple. The effect is that the network design has a surplus of spare capacity relative to the strict optimum for stub reuse. However, the excess is upper bounded by the difference between stub release and non-stub release designs. To more closely approximate stub reuse capacity designs one could start with the no stub release design and perform a capacity tightening operation on it. For each failure scenario, PRR-1 would be solved for the affected demands without access to any stub capacities. Then each restoration path is traced. For each restoration path that crosses the same span as its corresponding prefailure path, the spare capacity usage on that span, under the current failure scenario, is recorded as being one less than it actually was under PRR-1 using only spare capacity. If, after all failure scenarios are checked, the maximum record of spare capacity usage on any span is less than its non-stub release design quantity, the difference is attributable to stub reuse opportunities, and may be removed.

Generalized Failure Scenarios

This aspect of the general model refers to the fact that each failure scenario $F(z)$ can contain any number of simultaneously failed spans, not only single-span failures. This is useful to take specific SRLGs into consideration in the design but often may not be needed in comparative research or planning studies where it is adequate to compare alternatives on the basis of all single-span failure scenarios (or in circumstances where any specific set of SRLG assumptions would be arbitrary). The generalization of the failure scenarios influences the model from [Equation 6.24](#) through to [Equation 6.27](#), [Equation 6.24](#) generates the damage magnitude to each O-D pair for each

$$\zeta_i^{r,q}$$

generalized failure scenario. The set union operator is applied to the 1/0 indicator parameters that describe eligible working routes, so that for each demand pair, all spans of the failure scenario are considered in determining whether any working flow assigned to eligible route q is failed or not, but without double-counting such flow in assessing the total damage to relation. If, for instance, spans k and m are in series on eligible route q , and are both failed in scenario z then

Equation 6.33

$$\left(\bigcup_{j \in F(z)} \zeta_j^{r,q} \right) = \zeta_m^{r,q} \cup \zeta_k^{r,q} = 1 \cup 1 = 1$$

and so working flow $g^{r,q}$ will be recognized as failed. For the same reason the union operator also appears in [equation 6.27](#) to detect, but avoid over counting, the amount of surviving stub release capacity on each span when a given path may have been hit in more than one place by a multi-failure scenario. Here i indexes through failure spans of the scenario $F(z)$ detecting eligible routes q that have failed

$(\bigcup_i \zeta_i^{r,q})$. The boolean product of this result with $\zeta_i^{r,q}$ further tests whether a surviving span j (which must not be in the failure

scenario itself) also resides on the path q . In that case the working flow $g^{r,q}$ represents stub release capacity on span j under failure scenario z . [Equation 6.25](#) and [Equation 6.26](#) are only affected (relative to a single-failure model), in that restoration flows, stub release capacities, and the indicator parameters that define eligible restoration routes are all indexed by the general failure scenario index z , rather than by span index i , representing the set of single-span failures. The cardinality of F would always be greater than S for a set of generalized single and multiple failure scenarios and there are correspondingly more constraints of the form of [Equation 6.25](#) and [Equation 6.26](#).

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

6.7 Simplest Model for Path Restoration Capacity Design

Let us now look at the simplest basic form of path restoration capacity design model, derivable by applying all the simplifications mentioned above. This model corresponds to a path-restorable network with unit capacity provisioning in which working paths are assigned by shortest path routing (or some other independent routing process) before the optimization, which is for spare capacity only. As written, it also considers only single-span failure scenarios and restoration without stub release. This model is of special interest in practice for comparing path restoration to failure-independent path protection schemes which follow. This model is the simplest form of *failure-specific* path restoration:

path-SCA

min

Equation 6.34

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

1. The total demand lost by relation r equals the total restoration flow:

Equation 6.35

$$\sum_{q \in Q^r} \zeta_i^{r,q} \cdot g^{r,q} = \sum_{p \in P_i^r} f_i^{r,p} \quad \forall r \in D \quad \forall i \in S$$

2. Spare capacity is sufficient to support restoration flows:

Equation 6.36

$$\sum_{r \in D} \sum_{p \in P_i^r} \delta_{i,j}^{r,p} \cdot f_i^{r,p} \leq s_j \quad \forall j \in S | j \neq i \quad \forall i \in S$$

3. The restoration flows, $f_i^{r,p}$ and spare capacity s_j variables are integer.

In this model the working flows, $g^{r,q}$, are provided as input parameters and it is a set of multi-commodity restoration flow assignments

$f_i^{r,p}$

that is being solved so as to minimize the total spare capacity for restorability against all single-span failures. As such this model is the path restoration counterpart to SCA for span restoration. The two constraint systems express "restorability" and "spare capacity adequacy" in the same logical interrelationship between these considerations in SCA except that here the restoration flows are multi-commodity in nature, on the end-node pairs represented by r , rather being between the custodial nodes of the span failure. Not surprisingly the span-restorable design formulation is contained as a special case in path SCA. Span-restorable design is effected when

the set of demands corresponds to apparent "demands" between adjacent nodes only, i.e., $D = S$ and $X_i^r = w_j$ for every relation r .

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

6.8 Comparative Study of Span and Path-Restorable Designs

In the dissertation by Iraschko [Iras96], span and path-restorable capacity design formulations were implemented and comparative designs produced. In this section we summarize and discuss those results. Five test networks and demand matrices were used. Figure 6-10 shows the topologies used and Table 6-1 summarizes other characteristics. Network 1 was used with a uniform point-to-point demand of two demand units between all node pairs. Network 2 is the widely employed New Jersey LATA metropolitan area model which was initially published with a demand matrix, used here, in [Bell93]. Network 3 is another metropolitan area model and demand data set based on a planning model for Calgary, Alberta. Networks 4 and 5 and their demands are other models representative of long haul networks. In Table 6-1, $|D|$ denotes the total number of demand pairs exchanging a non-zero demand quantity. Total demand is the total number of unit-capacity working paths in the network.

Figure 6-10. Test networks on which span and path-restorable designs are compared [Iras96].

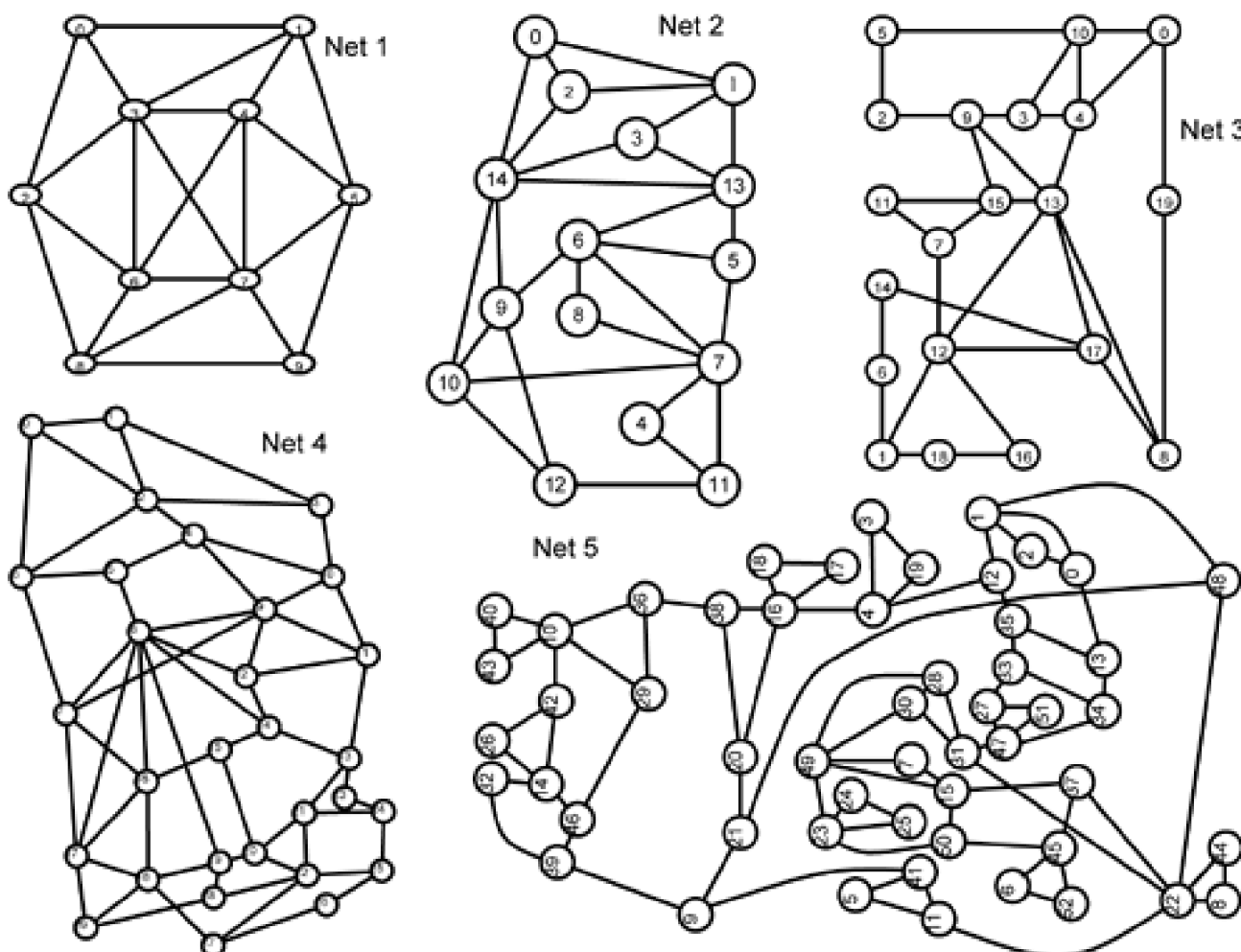


Table 6-1. Test Network Characteristics

Network	$ N $	$ S $	\bar{d}	$ D $	Total demand
1	10	22	4.40	45	90
2	15	28	3.73	67	824
3	20	31	3.10	153	2152
4	53	79	2.98	347	858
5	30	59	3.93	263	8312

The capacity placement for each network was optimized to restore all single-span failures. The first three cases optimize the placement of spare capacity only in span- and path-restorable networks. The working capacity design in cases 1, 2, and 3 split the routing of demand bundles between a node pair as evenly as possible over all the shortest hop-count paths between the nodes. The last three cases, 4, 5, and 6, repeat the first three design tests but with *jointly optimized* working capacity. Thus, in summary, the six capacity design cases are:

Non-joint designs

- Case 1: Optimize the placement of spare capacity only, in a span-restorable network.
- Case 2: Optimize spare capacity in a path-restorable network without stub release.
- Case 3: Optimize spare capacity in a path-restorable network with stub release.

Jointly-optimized designs

- Case 4: Optimize both working and spare capacity in a span-restorable network.
- Case 5: Optimize the placement of working and spare capacity in a path-restorable network without stub release.
- Case 6: Optimize the placement of working and spare capacity in a path-restorable network with stub release.

A joint path-restorable design obviously requires a large number of eligible working and restoration routes. For jointness, a set of eligible working routes needs to be specified for each O-D pair. From the fact that the restoration mechanism is path restoration, a set of eligible restoration routes is also needed for each O-D pair, under each failure scenario. In these results, rather than use a single explicit hop limit, eligible routes are limited to be no longer than the length of the respective shortest route between nodes plus a limited number of *additional* hops and a limited additional geographical distance above the shortest path. This route set is then supplemented with the set of all successively shortest span-disjoint routes between all pairs of nodes without distance or hop limits. This set of supplemental *k*-shortest routes is a relatively small addition but it ensures full representation of the network's topological connectivity between all nodes, which a single hop limit set at a feasible level set as $H = 6$ or 7 may fail to do, especially in a network such as network 5 [figure 6-10](#).

[Table 6-2](#) summarizes results for the 30 designs (6 design cases x 5 networks) in terms of total network capacities normalized in each case to the corresponding non-joint span-restorable SCA design result as a reference.

Table 6-2. Relative Total Capacity in Span and Path-Restorable Designs

Test Network	non-joint designs			jointly optimized designs		
	Case 1 (span)	Case 2 (path-w/o SR)	Case 3 (path with SR)	Case 4 (span)	Case 5 (path-w/o SR)	Case 6 (path with SR)
1	1.00	0.94	0.93	0.96	0.91	0.89
2	1.00	0.90	0.82	0.73	0.73	0.72
3	1.00	0.91	0.88	0.91	0.86	0.85
4	1.00	0.85	0.82	0.82	0.80	0.77
5	1.00	0.96	0.91	0.89	0.86	0.83
average reduction	0.0%	8.8%	12.8%	13.8%	16.8%	18.8%

Results show that without stub release, the reductions in spare capacity attributable to path-restorable design (in case 2) range from a minimum of about 6% in Network 1 to a maximum of 15% in Network 4. Note that the percentage reduction in spare capacity alone may be almost twice these numbers, but we have to make comparisons here on the basis of total capacity because in the joint designs working capacity is not constant across all alternatives. As expected, adding stub release (case 3) always helps, but in these tests it produced only 1% to 8% of additional spare capacity savings depending. If path restoration is used with stub release, but without joint optimization, the best improvement is about 18%, for both Networks 2 and 4.

For comparison of designs where working path routing and spare capacity assignment are *jointly* optimized we look at the last three columns of [Table 6-2](#). An interesting finding here is that joint optimization of the span-restorable designs has a fairly major effect on total capacity, up to 27% reduction here, and that in all the test networks the jointly optimized span-restorable designs were within a few percent of the best non-joint path restoration cases. In fact, networks 2, 4 and 5 require the same or less total capacity under joint span-restorable design than path-restorable designs with stub release. And four out of the five test networks are better under joint span restoration than non-joint path restoration without stub release (the latter case is of special interest because it closely approximates SBPP).

Considering case 6, we see that adding stub release to an already jointly optimized path-restorable design has little extra beneficial effect. A maximum of 5% further savings is observed. Finally it is worth noting that in case 6, we are looking at the most efficient designs possible: jointly optimized path restoration with stub release. These designs are on average 18.8% lower in capacity than (non-joint) span-restorable designs, with a maximum of 28% for network 2.

6.8.1 Demand Dispersion and Routing Effects

The jointly optimized designs were used to check on the demand dispersion principle that was hypothesized in the lower bounding exercise of [Section 6.5](#). When the ILP is allowed to jointly optimize the placement of working and spare capacity in a network, it will tend to choose working paths which are coordinated with the network restoration process. If the insights from the bounding exercise have validity we expect to see a leveling out of flow assignments over relation over nearly equal-length routes and possibly some routing via paths longer than the shortest path. [Table 6-3](#) shows the standard deviations (s.d.) of the w_j quantities in the different types of design, computed over all the spans of each network. In the non-joint designs, working path routing is identical in each design so cases 1, 2, 3 all have the same s.d.(w_j) values. Relative to the non-joint designs, [Table 6-3](#) indicates that with joint optimization the w_j quantities on each span do tend to be significantly more leveled out. This is consistent with the hypothesis of demand dispersive routing under path restoration.

Table 6-3. Measures of the Demand-Routing Dispersion Effect

Network	s.d.(w _i) cases 1, 2, 3	s.d.(w _i) case 4	s.d.(w _i) case 5	s.d.(w _i) case 6
1	1.59	0.891	0.940	0.656
2	68.2	34.6	33.4	34.6
3	115	97.5	94.5	106
4	27.3	22.2	22.3	25.7
5	617	536	590	593

[Table 6-4](#) also confirms that the average working path length in the combined capacity designs is longer than the average working path length in the spare capacity designs, but only slightly. This suggests that the effectiveness of path restoration can in part be attributed to the significant route dispersion of demands without dramatically increasing route lengths.

Table 6-4. Average Working Path Lengths under Joint Optimization

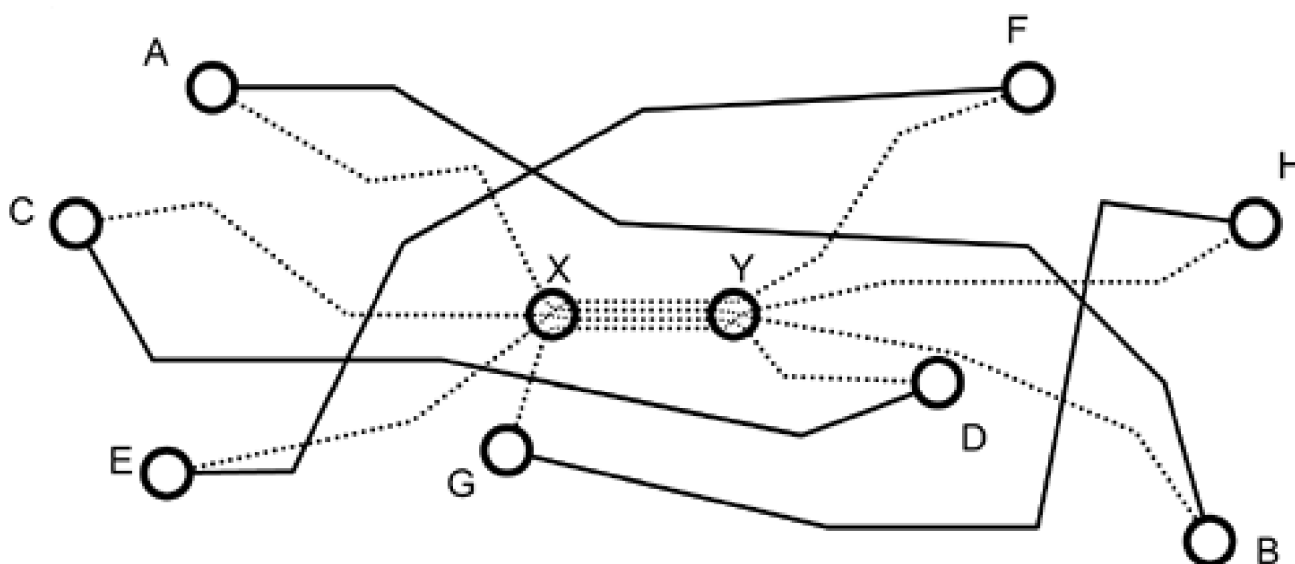
Network	Shortest path routing of demands (cases 1, 2, 3)	Jointly optimized designs		
		Span restoration design (case 4)	Path restoration design (case 5)	Path restoration /SR (case 6)
1	1.6	1.6	1.6	1.6
2	15.5	17.1	16.7	17.8
3	109.4	115.1	118.0	131.3
4	818.5	848.9	839.4	994.7
5	123.3	128.6	133.1	137.9

6.9 Shared BackupPath Protection (SBPP)

Shared backup path protection (SBPP) is a path-oriented protection scheme with a particular combination of operational simplicity, speed, and efficiency that makes it of special interest for IP-centric optical networking and MPLS layer contexts where OSPF-TE and CR-LDP type protocols enable service layer applications to set up their own SBPP path arrangements on demand. So far our discussion has been focused on dynamic *failure-specific* path restoration. SBPP is a *failure-independent* path-oriented scheme where the protection route is identified in advance, but spare capacity has to be cross-connected on the backup route in real-time. In other words, SBPP is an intermediate scheme in the sense used in [Section 3.5.1](#)—in between pure protection and pure restoration. It is perhaps best described as a *preplanned restoration* scheme. Any failure that affects the working path causes a switchover to the predefined backup route and crossconnection to form a backup path from shared spare capacity. One can equivalently think of SBPP as being like 1+1 APS DP but where the capacity used to form the protection path is shared over failure-disjoint working paths. In terms of the operational concept, SBPP is also logically identical to ATM Backup VP Protection [\[KaSa94\]](#) with an MPLS path or a lightpath replacing the VP. The understanding of capacity sharing rules associated with SBPP is improved, however, so that oversubscription effects are controlled in the MPLS context and are completely avoided in the lightpath context.

[Figure 6-11](#) illustrates the concept of spare capacity sharing on predefined backup routes. O-D pairs AB, CD, EF, and GH use primary paths that are mutually disjoint. They may therefore each employ a 1+1 APS DP arrangement that relies on sharing the same spare channels on span X-Y to form their backup paths. The key requirement is that no single failure will affect the mutually disjoint primaries, so their backup paths will not require the same spare channels at the same time. Each primary plus backup path pair, viewed in isolation, thus forms a 1+1 APS-like arrangement.

Figure 6-11. Example of four disjoint primary paths sharing the spare capacity on span (X-Y) in their backup paths.

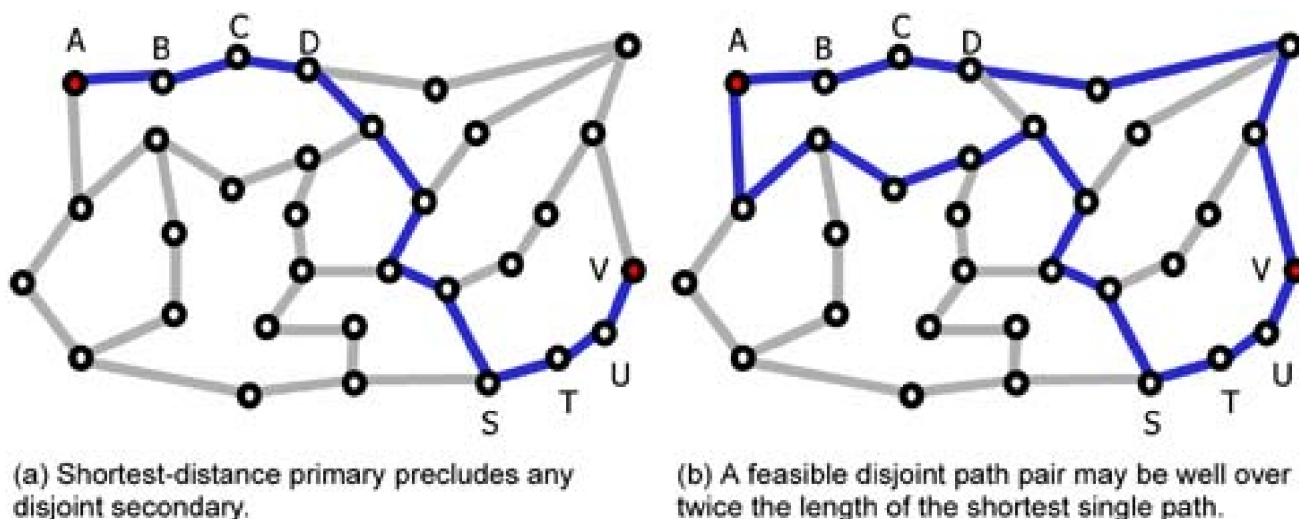


6.9.1 Infeasibility in Greedy Disjoint Path Pairs

A basic idea associated with SBPP is that service layer nodes, enabled with OSPF-TE and CR-LDP, will be able to set up their own primary and backup path arrangements dynamically. An interesting question is whether they can use a shortest path algorithm alone to

support the route selection for SBPP. In practice, SBPP primary plus backup path pairs can most often be found by taking the shortest route for the primary path and the next shortest route, excluding spans of the first, for the backup route. In effect this is ksp with $k=2$ assuming only one unit of available capacity on all spans (to find disjoint routes) and this would be easily supported in OSPF-capable nodes. This can fail, however, in circumstances where a second path is infeasible if the shortest route is taken for the first. The problem can be overcome by trial and error (different choice of primary routes) or with the min-cost disjoint path pair algorithm of [Section 4.7](#). Depending on the network such infeasibilities may, however, be rare enough that the phenomena is not recognized when it arises. (With the benefit of an overall view, the problem is easily seen, but when it is encountered from a more embedded standpoint it may cause some software process to halt or fail.) It is useful therefore to have an appreciation of when and why this happens. In effect this is the explanation of why we use Bhandari's or Suurballe's algorithms. [Figure 6-12\(a\)](#) shows a case of such an infeasibility.

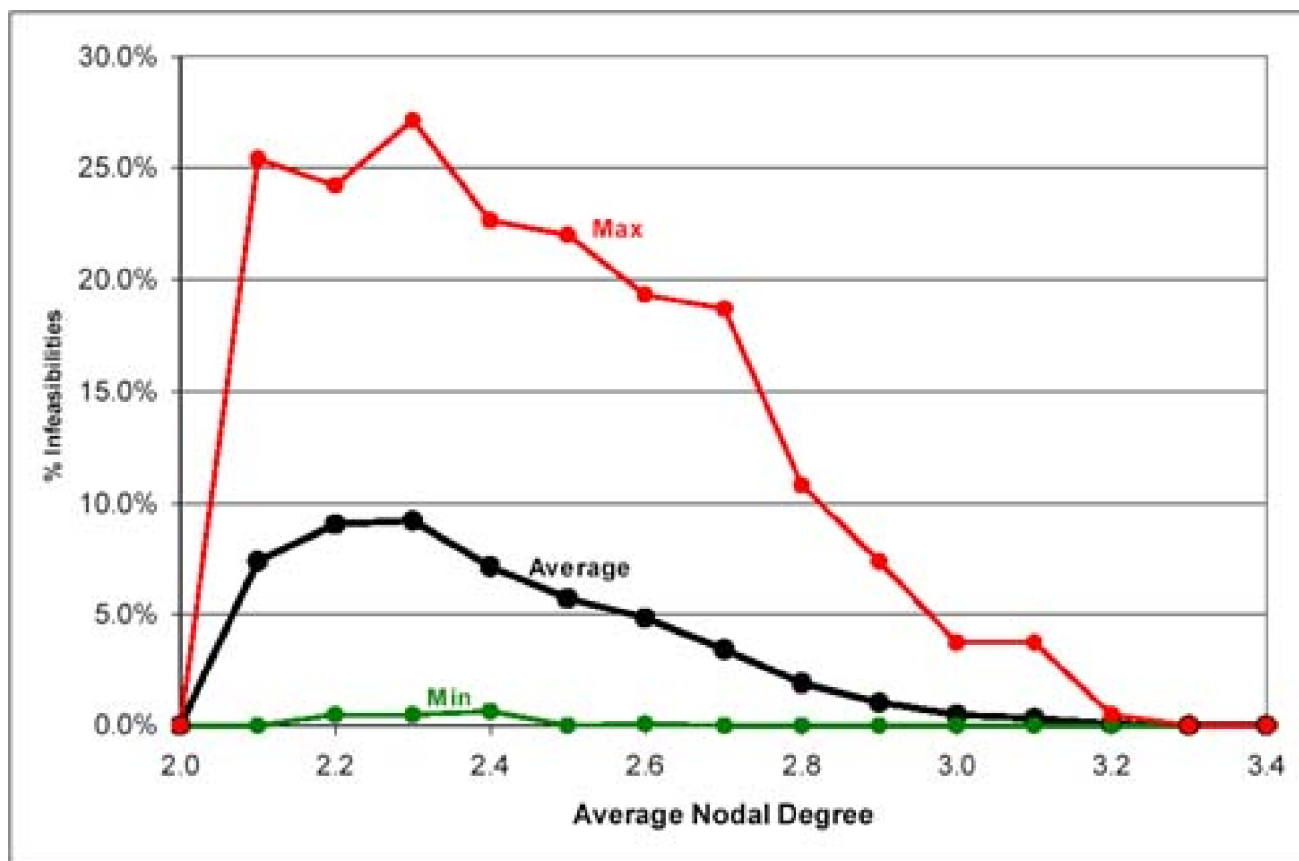
Figure 6-12. Example where the shortest primary path precludes a disjoint backup path.



The particular instance arises between nodes A and V because the shortest path bisects the graph and runs along its perimeter at both ends, leaving no disjoint route options. The same situation exists for all node pairings in $\{A,B,C,D\} \times \{S,T,U,V\}$ in this example. It is not difficult to overcome the problem, but the point is to be aware of the issue, especially in writing automated applications for SBPP provisioning: greedy will inevitably fail at some point possibly causing a software crash if the issue is not anticipated in the implementation. [Figure 6-12\(b\)](#) shows a feasible solution in which neither path is the shortest between nodes A and V. [Figure 6-12\(b\)](#) also illustrates some issues for SBPP-based network design. One is that the total length of a feasible path pair can well exceed twice the shortest path distance, especially in sparse graphs. This means that for capacity efficiency well-coordinated sharing of the spare capacity to form backup paths is going to be required. A second issue is that primary and backup paths each reach end-to-end across the network. This is a possible concern from an availability standpoint. Because the path pair is long and the backup is predetermined, any two failures where one hits anywhere on the primary and the other anywhere on the backup is an outage-causing state. In contrast, dual failures have to be much more localized (close to each other on the network) to cause outage under span restoration. More on availability and dual-failure considerations follow in [Chapter 8](#).

The issue of infeasibility under the greedy method for SBPP path determination is conceptually like the "trap topology" where ksp differs from max-flow for span restoration ([Section 4.6.4](#)). But the path-oriented manifestation of infeasibility can be much more frequent. The prior issue of whether a trap topology results in a ksp versus max-flow difference in span restoration also depends on the capacity distribution on the graph, whereas in the context of forming SBPP path pairs, the infeasibility depends solely on the topology. Some experimental data on the statistical frequency of this phenomena and how it varies with network connectivity is shown in [Figure 6-13](#). This data is from a set of trials conducted on all O-D pairs in a large number of random transport network graphs of varying average nodal degree. [Figure 6-13](#) shows the results of 672 individual test networks in terms of the worst-case, average and minimum percentage of O-D pairs exhibiting infeasibility at each nodal degree. It confirms that this is more of an issue in sparse networks. Above nodal degree of about 3.3 the infeasibility is hardly ever seen. However, the average frequency of infeasibilities reaches a peak of almost 10% of all O-D pairs for networks with degree in the 2.1 to 2.4 range. In the worst-case topologies in this degree range it is up to 25%. This is exactly where some North American inter-exchange carrier network graphs are found and it suggests that it is essential for them to use a true min-cost disjoint path pair algorithm, rather than the shortest successive path pair approach. Note that each infeasibility is also a marker of cases where an especially long total route length may be needed for primary plus secondary, aggravating both capacity requirements and availability concerns with SBPP on such sparse topologies.

Figure 6-13. Statistical frequency of the greedy path pair infeasibility versus network degree.



6.9.2 Discounting the Shared Backup Path: Asymmetric Path Pairs

A further consideration about finding min-cost path pairs for incremental provisioning of a new SBPP path is that, once other SBPP path arrangements are in place, the *min-cost* path pair for the next SBPP setup is not necessarily provided by either the greedy successive paths method or by a min-cost disjoint path pair algorithm in a context where other SBPP services are already established. The reason is that neither of these criteria for choosing a backup route directly take opportunities for the sharing of spare capacity into account. They give equal (or "symmetric") weight to primary and backup path costs. If there was no sharing of the spare capacity for backup paths (i.e., dedicated 1+1 APS) the minimum total capacity design would correspond to the sum of the individual minimum-cost path pair problems for each demand individually. The problem of finding a minimum cost pair of such disjoint symmetric-weight paths has been extensively studied by Bhandari [Bhan99] and is the motivation of the basic Suurballe algorithm [Surb74]. But in the SBPP context, the capacity on backup routes is *shared* over other failure-disjoint primaries, creating *asymmetric* weighting of cost. This leads to a surprisingly difficult problem for exact solution and is currently an open area of research. (See [LaTa01], [OkNo02] to tap into this line of work.)

To explain the problem, the minimum cost criterion for a symmetric disjoint path pair is

minimize

Equation 6.37

$$\left(\sum_{j \in \{pri\} \cup \{back\}} c_j \cdot L_j \right)$$

whereas with asymmetric path costs it becomes

minimize

Equation 6.38

$$\left(\sum_{j \in \{pri\}} c_j \cdot L_j + \sum_{m \in \{back\}} \left\{ \frac{c_m}{\sum_{k \in B} \delta_{mk} \cdot \left(1 - \bigcup_{j \in \{pri\}} \delta_{j, pri(k)} \right) + 1} \right\} \cdot L_m \right)$$

where the minimization is over the set of $\{pri\}$, $\{back\}$ route choices and

- $\{pri\}$ is the set of spans corresponding to the primary route decision. j indexes spans along this route. $c_j L_j$ is the cost of capacity used by the primary on span j .
- $\{back\}$ is the set of spans comprising the backup route. Its spans are indexed by m and contain no span also in $\{primary\}$.

A channel on span m on the backup route would cost $c_m L_m$ if no sharing of capacity was involved. As proposed, however, backup span cost is amortized over the total number other primaries having backups that share the same spare capacity. This is described by:

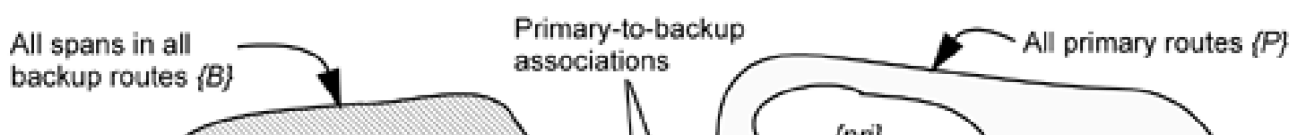
- B is the set of all backup paths in the network as a whole, indexed k , and
- $d_{mk} = 1$ indicates that a path k also crosses span m (and is zero otherwise).
- $d_{j, pri(k)} = 0$ indicates that the primary path associated with backup path k does not include span j (and is one otherwise).

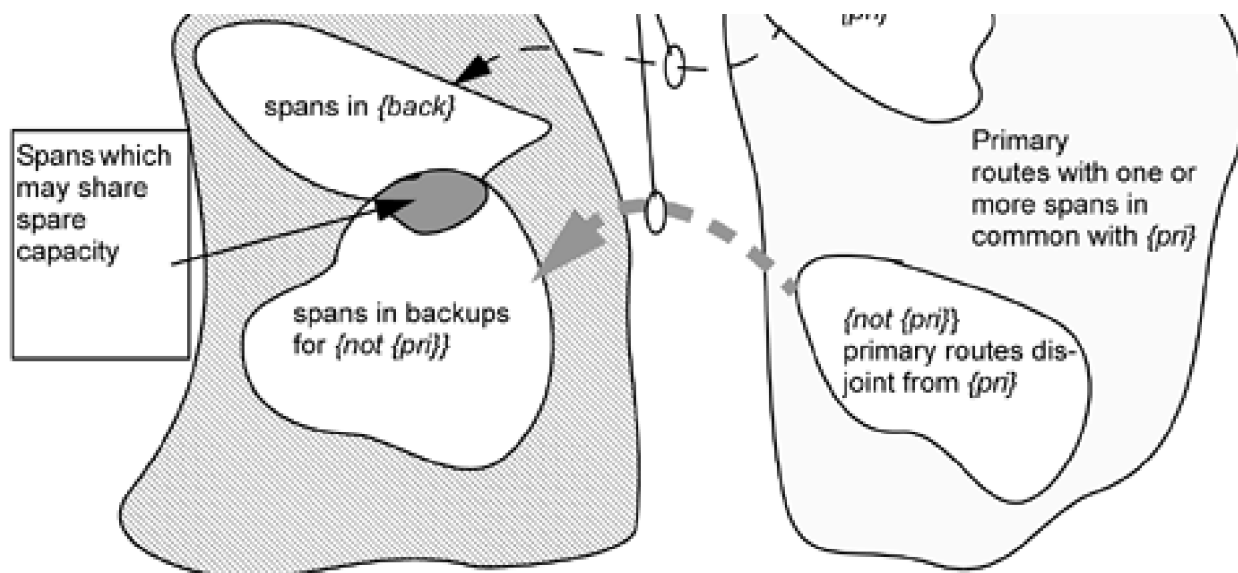
The idea is that if, say, four other backup paths already use span m in their routes, and all of the associated primaries are disjoint from the current primary, then span m can be considered to have 1/5th its normal capacity cost when being considered in the backup route for the current additional primary path. The denominator under c_m in the right hand term just expresses all the logical qualifiers to join the sharing relationship on span m . To share a spare channel with the current backup route on span m , another backup k must cross span m but be associated with a primary path that has no spans in common with the primary route $\{pri\}$ under consideration.

6.9.3 Disjoint Primary/Shared Backup Relationships: Venn Diagram

In [Figure 6-14](#) a Venn diagram approach illustrates the logical relationships between routes that are required for sharing of spare capacity on spans. The two universes; primary routes $\{P\}$ and backup routes $\{B\}$ are related by a one-to-one mapping between each primary and backup route. Between disjoint primary routes in $\{P\}$ there may be spans in common on routes that are not disjoint in $\{B\}$. Between routes in $\{P\}$ that are not disjoint there can be no spans in common on associated routes in $\{B\}$. (It is understood here that each route itself is just a specification of the spans it contains. Where routes intersect the result is a set of spans. The actual sharing occurs on spans which are common to one or more routes.)

Figure 6-14. Relationships between routes for spare capacity sharing on spans in SBPP.





It can be appreciated that a great deal of global information about network state is needed to formulate the incremental SBPP path planning problem (Equation 6.38) for independent on-line provisioning operations by each node. An alternative approach, and a topic worth studying in its own right, is the corresponding global capacity-sharing optimization model for SBPP. The global model will be able to collectively optimize the choice of primary and backup routes for best capacity efficiency, if all demands are forecast in advance. We will again face the dichotomy of having a global optimization that maximizes efficiency, but assumes full knowledge of the demand pattern, and an incremental path provisioning problem (above) which assumes nothing about the overall demands, but is not as efficient because it solves each dynamic demand problem in an isolated incremental (or "greedy") context. As with span restoration, however, knowledge of these two extremes enables overall guidance of the network planning process and provisioning policies and permits development of a number of "batch update" models which occupy an intermediate position between strictly global optimization and immediate single incremental path problems. The global design optimization model is also the one that is best suited to fundamental comparative studies between basic architectural alternatives.

6.9.4 Optimal Design of Shared-Backup Path Protected Networks

Accordingly, let us now look at two global capacity-design models for SBPP networks. One takes an arc-path approach and the other uses an arc-flow model. For the arc-path model we have the traditional concept of eligible routes, but a choice we have is whether to define eligible backup routes for already given primary routes, or, to define eligible path-pairs as whole decision entities. The first approach corresponds to a non-joint version of SBPP design and it admits the possibility of encountering the infeasibilities discussed above. The other approach is to define eligible primary and backup route pairs directly and make a single decision for each O-D pair to be served. This is the equivalent to the "joint" design problem for SBPP. Another design choice is whether to use a unique backup route for all demand flow for each O-D pair, or whether each backup will be defined on a unit-demand basis. For instance, in the first case if pair (O-D) is exchanging five lightpaths, then all five will follow the same primary route and be protected by spare capacity on the same backup route. In the latter case, each lightpath could be planned to have its own individual backup path and each could follow a different route.

Let us therefore introduce a basic model and discuss how it is adapted to each of these cases. In the initial model we will assume all demands between O-D pairs follow the single shortest route (or any single route provided as an input, in the event that it has to be a different route than the shortest, to avoid infeasibilities) and that all demand flow between a node pair stays as a bundle switched onto a set of backup paths on a common backup route. In this case:

- ζ_i^r is an indicator parameter for the primary routes. It is one if the primary route for relation r crosses span i , and zero otherwise.
- d^r is the total number of demand units on relation r .
- B^r is the set of eligible backup routes for relation r . All routes in this set are fully disjoint from the primary route, but simply distinct with respect to each other, index b .
- $\zeta_i^{r,b}$

ζ_j^r functions like ζ_j^r but encodes the eligible backup routes for each relation. It is one if the θ^{th} eligible backup route for relation r crosses span j , and zero otherwise.

- r_b, r are the backup route decision variables. Equal to one if the θ^{th} eligible backup route for relation r is chosen and zero otherwise.

SBPP-1

Minimize

Equation 6.39

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

1. One backup route per working demand (or demand bundle):

Equation 6.40

$$\sum_{b \in B^r} \rho_{b,r} = 1 \quad \forall r \in D$$

2. Spare capacity is sufficient to support simultaneously activated backup paths:

Equation 6.41

$$\sum_{r \in D} \sum_{b \in B^r} \zeta_i^r \cdot \rho_{b,r} \cdot \zeta_j^{r,b} \cdot d^r \leq s_j \quad \forall j \in S | j \neq i \quad \forall i \in S$$

3. The decision variables are binary $r_b, r \in \{0, 1\}$ and spare capacity s_j variables are non-negative integer.

The first constraint system is self-explanatory but the second is more involved. Each of the following considerations is built into the second constraint system ([Equation 6.41](#)):

1. The failure span i must hit the working path for relation r . This requires $\zeta_i^r = 1$.
2. Backup path b is the choice for protecting relation r , $\rho_{b,r} = 1$.
3. Backup path b crosses span j , (that is: $\zeta_j^{r,b} = 1$) in which case.
4. Demand flow d^r imposes itself on the spare capacity for span j .

The effect of all the constraints generated by [Equation 6.41](#) is to enumerate all non-simultaneous failure scenarios, triggering the

corresponding set of simultaneously activated backup flows for each scenario. Even though the backup path for each relation is not itself failure specific, the model still needs to consider all failure scenarios globally because each produces a different combination of simultaneous backup flows.

It may seem counterintuitive that while SBPP is based on a simpler and more constrained restoration mechanism than generalized path restoration, the SBPP-1 design model can be much more difficult to solve than *path_SCA*. The main reason is the number of binary decision variables and the fact that these are inter-linked in the constraints, (i.e., r_b, r appears in both). SBPP and *path_SCA* both have a set of $|S|$ integer spare capacity variables and they have identical meaning and roles in each model. It is the restoration flow variables in *path_SCA* that correspond to the backup decision variables in SBPP. In *path_SCA* one needs to solve for a larger number of the restoration flow variables than there are corresponding backup path choices in SBPP, but the former flow variables can be relaxed without effect on the spare capacities. But in SBPP there is no corresponding relaxation of the 1/0 decision variables associated with backup route choices. If a restoration flow quantity comes out as 7.6 units under *path_SCA*, we know how to "repair" the solution fairly easily and the objective function is not affected in any case. But we have no way to interpret or repair a relaxed solution if a backup path choice variable is fractional. There is no escaping that r_b, r must be held as a binary variable and this makes large SBPP problems generally harder to solve than a corresponding *path_SCA* problem.

As described so far, demands are handled in a bundled way on each working and backup route. In this context the set of relations is just the set of all O-D pairs. The simplest way to adapt the model so that individual demands may take individual backup routes is to expand the set of relations accordingly. That is to say, we let r index over each demand unit individually on all O-D pairs. In principle this can lead to greater capacity efficiency because protection decisions and spare capacity allocation are made at the unit-demand level, not the O-D pair bundle level. However, it also produces a corresponding expansion in the number of 1/0 variables. Somewhat fortunately for the capacity planner, however, having all demands on the same O-D pair follow the same primary and backup routes seems often to correspond to the operational preferences of network operators when considering SBPP.

Jointly Optimized SBPP

If one wants to provide eligible working and backup path pairs as intact units for choice in the design, the basic form of the model does not change, but the interpretation of the variables and the corresponding input parameters changes as follows:

- B^r is the set of distinct eligible primary-backup route pairs for relation r , index b . These sets are conveniently generated from an all-distinct cycles algorithm ([Section 4.10](#)) whose output is selected and sorted to represent the desired number of shortest cycles containing the O and D nodes of each relation r .
- $\Theta_i^{r,b}$ replaces ζ_j^r and is one if the *primary route part* of the b^{th} eligible path pair for relation r crosses span i , and zero otherwise. It can be assumed that the shortest path between the end nodes of relation r on the cycle is taken as the primary route.
- $\zeta_j^{r,b}$ is one if the *backup route part* of the b^{th} eligible path pair for relation r crosses span j , and zero otherwise.
- r_b, r equals one if the b^{th} eligible path pair is chosen to serve relation r and zero otherwise.

In addition, the direct choice of path pairs for the solution means that the working capacity on each span also becomes a variable (as opposed to an input above). The revised model then takes on the joint type of objective function and explicit w_j variables reappear:

joint_SBPP

Minimize

Equation 6.42

$$\sum_{j \in S} c_j \cdot (w_j + s_j)$$

subject to:

1. Working capacity is sufficient to support the routing of working paths:

Equation 6.43

$$w_j = \sum_{r \in D} \sum_{b \in B} \Theta_j^{r,b} \cdot d^r \quad \forall j \in S$$

plus:

2. [Equation 6.40](#), [Equation 6.41](#) and binary constraints on r, b , integer non-negativity constraints on w_j, s_j .

One way to address the computational difficulty of either of these SBPP design models is similar to the route enumeration strategies for prior SCA or JCA problems. That is, to set a minimum target number of eligible primary-backup pairs, typically 5 to 20, for each relation. One starts by running the all-distinct cycles algorithm on the graph, producing the complete file of all cycles (possibly only up to some hop limit). The master list is then filtered for each O-D pair to produce sub-files of cycles in which each O-D pair is contained. The sub-file for a given O-D pair can then be sorted by total length or cost of each cycle and the target number taken from the least cost end of the list. Note that this procedure will always include the min-cost disjoint path pair for each O-D pair as one of its options.

Adding modularity to the SBPP models is essentially the same as for span or path restoration. One changes the objective function to be of the form of [Equation 6.21](#) and modular capacity constraints of the form [Equation 6.28](#) are added.

6.9.5 Wavelength Assignment Under SBPP

SBPP is of particular interest for optical networks managed at the lightpath level. If these networks are based on "opaque" cross-connects that can change wavelengths as needed at each node, then the capacity-only type of design models above can be used directly, because there will be no wavelength blocking. By this we mean that while blocking due to sheer capacity limits remains, blocking will not occur while capacity is available due to any mismatch in wavelengths. This is otherwise known as the *virtual wavelength path* or VWP design context. At the other extreme is an assumption that no node will have any wavelength-conversion capabilities. This is the so-called "transparent" network in which wavelength assignments have to be explicitly coordinated so that clashes are avoided on the entire end-to-end path, known as the *wavelength path* (WP) context. Adding wavelength consistency constraints can make an already difficult SBPP problem considerably worse because the number of 1/0 decision variables is multiplied by W —the number of wavelengths assumed per fiber pair. In practice we also expect that the most realistic way to plan and operate an SBPP network is to effect a "capacity only" or VWP SBPP design and then solve a subsidiary problem for the number and placement of wavelength converters or o-e-o OXC nodes, so that wavelength blocking is effectively eliminated. This way the network obtains maximum operational flexibility from the given capacity investment, on-line RWA is greatly simplified, and the total capacity and number of distinct wavelength channels required on each fiber are all significantly reduced relative to the pure WP case.

Nonetheless it is worth seeing how the WP context can be added to the SBPP design models. The pure WP network design problem is also of continuing theoretical interest. The main considerations in the SBPP context are to assign the same wavelength to all links of any path (either a primary or backup) and to avoid wavelength assignment conflicts on each span. It is usually assumed that the primary and backup paths of the same path pair may have different wavelength assignments because fault detection and payload switchover is done in the electronic domain at path-end nodes. The backup path launch wavelength can thus be different than its corresponding primary. To reflect the WP context we need additional input parameters and it is also more natural to consider a modular viewpoint in which each module of capacity corresponds to addition of a fiber pair with W wavelength channels on a span. For compactness we will work with the WP counterpart to SBPP-1 (i.e., the non-joint problem where only spare capacity costs are minimized). For the non-joint problem we can also assume that the routing and wavelength assignment for working demands is summarized for us as input data in terms of the number

of wavelengths already used by working paths on each wavelength on each span. This leads to a model which is non-joint in the sense that we do not assign working routes and wavelengths when optimizing the spare capacity, but there is working-spare coordination in that the model chooses backup routes and wavelengths in a way that reflects the working wavelength use of the working paths to minimize the total fiber counts needed after backup routes and wavelength assignments are added. Thus, we will have:

- K = set of bidirectional wavelength channels available on each fiber pair, index k . $|K| = W$ is the number of wavelengths provided per fiber pair.
- c_j^f = cost of a fiber pair on span j . Supports up to W channels but does not include cost to equip all W channels individually.
- c_j^λ = cost per incremental wavelength channel used ("turned up") on a fiber, whether for working or spare.
- w_j^k = number of instances of wavelength k already used on span j for working paths.
- $\zeta_j^{r,b}$ = one if the b^{th} eligible backup route for relation r crosses span j , and zero otherwise.
- ζ_j^r = one if the i^{th} failure scenario causes failure of the working path on relation r . (Obtained as an input from the routing of working paths prior to this problem).
- $r_{b,r}$ equals one if the b^{th} eligible backup route is chosen to serve relation r and zero otherwise. (binary route decision variable).
- $\lambda_{b,r}^k$ = one if wavelength k is assigned to the backup route chosen for relation r , zero otherwise. (binary wavelength assignment decision variable).
- n_j^f = number of fiber pairs to be installed on span j (integer variable).
- s_j^k = number of spare channels on span j in wavelength k (integer variable).

WP-SBPP-1

Minimize

Equation 6.44

$$\sum_{j \in S} \left\{ c_j^f \cdot n_j^f + \sum_{k \in K} c_j^\lambda \cdot s_j^k \right\}$$

subject to:

1. One backup route assignment per working demand:

Equation 6.45

$$\sum_b r_{b,r} = 1 \quad \forall r \in D$$

$$\sum_{b \in B^r} \lambda_{b,r}^k = 1 \quad \forall r \in D$$

2. One wavelength assignment per backup route:

Equation 6.46

$$\sum_{k \in K} \lambda_{b,r}^k = 1 \quad \forall r \in D$$

3. Spare capacity is logically sufficient to support all simultaneous restoration flows on each span and in the required wavelength on each span: [\[2\]](#)

[2] While this model serves to specify the WP-SBPP-1 problem, it is not strictly linear as stated—constraint 3 contains the product of two 1/0 variables. Ways to handle this are discussed later in [Section 10.4.3](#).

Equation 6.47

$$\sum_{r \in D} \sum_{b \in B^r} \zeta_i^r \cdot \rho_{b,r} \cdot \zeta_j^{r,b} \cdot \lambda_{b,r}^k \cdot d^r \leq s_j^k \quad \forall j \in S | j \neq i \quad \forall i \in S \quad \forall k \in K$$

4. The number of fibers (each providing one set of all wavelengths) on each span is sufficient to support the wavelength assignments made for each combination of simultaneously activated WP backup paths:

Equation 6.48

$$n_j^f \cdot W \geq \sum_{k \in K} (w_j^k + s_j^k) \quad \forall j \in S$$

5. The $\lambda_{b,r}^k, \rho_{b,r} \in \{0, 1\}$ decision variables are binary and the spare capacity and number of fibers are non-negative integer.

6.9.6 SBPP Design with Limits on the Sharing of Spare Capacity

Returning to the VWP or "capacity-only" context, note that the SBPP models so far given do not have any control over the number of primaries that may share the same spare capacity on their backup routes. One concern about SBPP is that its service availability is affected not only by any combination that hits both arms of its 1+1 structure but also through unavailability of the backup capacity. In latter case a service path affected by a fiber cut cannot be restored because part of the spare capacity along its backup path has already been used for failure protection of another apparently unrelated service path.

More follows on the aspects of dual-failures and related availability considerations in [Chapter 8](#), but for now these considerations suffice to suggest that we may want a way to design SBPP networks so as to control the maximum extent of sharing on spare channels. Allowing fewer service paths to share the same spare capacity reduces the probability of a situation where two otherwise unrelated span failures, with overlapping repair intervals, may lead to outage not because they hit both primary and backup of any one service, but because the two primary paths are in a sharing on their backup paths. The question of interest is the extent to which sharing can be limited without

unacceptable increases in total capacity requirements. To control the maximum sharing, we can add the following constraint to any of the SBPP models above:

Equation 6.49

$$\sum_{r \in D} \sum_{b \in B^r} \rho_{b,r} \cdot \zeta_j^{r,b} \cdot d \leq F \cdot s_j \quad \forall j \in S$$

where F is the maximum allowed number of shared dependencies on spare capacity to form backup paths.

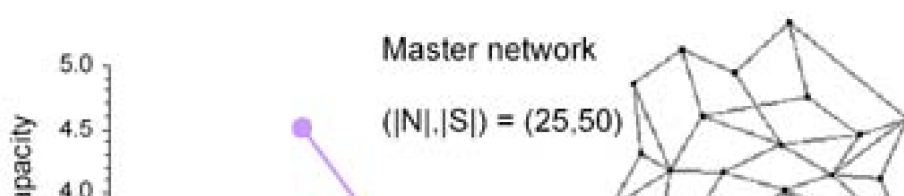
This constraint looks similar to [Equation 6.41](#) which generates the spare capacity requirements for each span based on all non-simultaneous failure scenarios. But the sharing limits constraint does not consider failures as being non-simultaneous. Instead it measures the total flow that would be asserted on each span if all backup paths that cross it were suddenly activated (i.e., simultaneously). In effect this measures the total "bookings" made upon the spare capacity on span j . Say in total there are 10 spare wavelengths on span j . If there are no more than 20 other primary paths which for one failure or another would use backup capacity on that span, then we can say that no unit of spare capacity on span j is shared over more than two primaries.

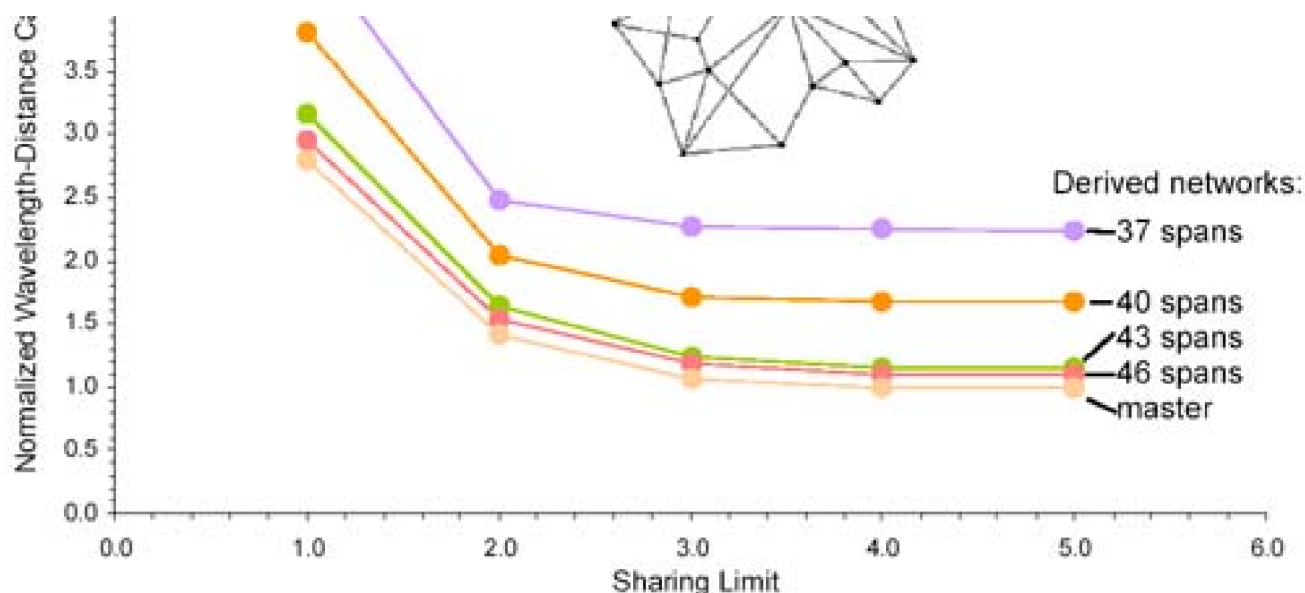
6.9.7 Capacity Effects of Sharing Limits in SBPP

There are two expected effects from adding [Equation 6.49](#) to the SBPP formulation. We expect that dual span failures will have a reduced impact on service availability but that capacity requirements will increase as a consequence. In [\[DoCl03\]](#) these effects were studied in detail on three families of networks consisting of a master network and a series of progressively sparser networks derived from the master, as in the method employed in [\[DoGr01\]](#). We will leave the availability-related aspects for discussion in [Chapter 8](#). Here let us consider how capacity requirement varies with the SBPP sharing limit.

[Figure 6-15](#) is a selected result showing how restricting the maximum sharing increases the capacity requirements. In [Figure 6-15](#), each curve shows capacity costs versus sharing limits for the specified network derived from the 25-node 50-span master network shown. A specific network from one of these families is denoted by the number of spans it contains. Costs are calculated as the sum of the wavelength-distance product required on each span, including both primary and backup capacity. Over the three families of test networks studied in [\[DoCl03\]](#), the cost of designing the networks with no sharing at all (i.e., sharing limit of 1, representing 1+1 APS) was on average 85% greater than the design cost if we allowed two O-D pairs to share the same backup capacity (sharing limit of 2). Going from a sharing limit of 2 to 3 reduced the design cost an additional 23% on average. However, allowing more than three O-D pairs to share the same backup capacity (a sharing limit of 4 or above) provides only a small further improvement (5% on average). A practical guideline from this is that there seems to be no significant capacity-related motivation to allow sharing factors higher than 4 or 5. Conversely, it suggests that a network audit could be conducted in real SBPP-based networks to highlight any span on which spare capacity is found to have more than about a five-fold booking for its use over disjoint primary failures. Changes to backup path assignments to cap this at about four would yield network availability improvements and based on the capacity findings should usually be feasible through a route change only without an added capacity requirement. Interestingly, in independent work and on different test networks Ramamurthy et al [\[RaLa03\]](#) also found that minimum-capacity requirements could be approximated with a cap on sharing at a factor of five or six depending on the network (a sharing cap of 10 was sometimes required to strictly reach the minimum capacity). They also noted (in presentation) that if a sharing cap is not enforced the resulting designs could contain individual spare channels with up to nearly 20 sharing relationships.

Figure 6-15. Effect of sharing limits in SBPP design on total capacity requirements for networks of varying connectivity (from [\[DoCl03\]](#)).





6.9.8 Arc-Flow Models for SBPP

A general advantage of arc-flow models (see [Section 4.11](#)) is that they require no preliminary processing to generate eligible route sets as in an arc-path model. On the other hand, arc-flow models do not inherently allow control over the restoration route lengths or other properties. In addition, actual path constructions have to be inferred from the pure flow variables of an arc-flow model. Generally, however, there are advantages to having both arc-flow and arc-path models to use in different contexts. Especially because arc-path SBPP involves so many 1/0 variables it may at times be easier to solve a corresponding arc-flow model. 1/0 variables are not avoided in the arc-flow model but they become span-wise flow decision variables under the special transshipment-like network problem structure. In contrast the arc-path models are more purely combinatorial 1/0 decision problems about selection of entire routes for backup paths, without any "network structure" in the sense of an arc-flow model for which commercial ILP solvers have special solution tactics.

In this section we therefore consider an arc-flow model for the basic SBPP problem where working demands are already routed on a single path per relation. The aim is to minimize the total spare capacity that provides for one shared backup route for each working route. As generally above, i denotes failure scenarios, j denotes spans in a non-failed context, D_j is the set of relations or O-D pairs affected by failure i , d^r is the amount of demand on relation r (and may be 1 if each pair of individual lightpath end-nodes is defined as a relation). In addition:

- ξ_i^r is one if the primary route for relation r crosses span i , and zero otherwise (an input).
- $O(r)$ = the origin node of demand pair r .
- $D(r)$ = the destination node of demand pair r .
- x_{ij}^r is a backup flow-structuring variable. It is one if the decision is to route backup flow for relation r over span j when span i fails. It is through assertion of a 1/0 nature on these structuring variables that we keep the arc-flow solution constrained to use a single backup route. Normally an arc-flow model would otherwise permit a general spread of flow.

In addition, because this is an arc-flow model, we need to index over nodes and to identify spans incident on a specific node. Therefore let N be the set of nodes, indexed by n , and let the notation $j@n$ define the set of spans incident on node n in the graph.

Arc_flow_SBPP

Minimize

Equation 6.50

$$\sum_{j \in S}$$

subject to:

1. Unit logical flow to structure backup routes; source, sink, and transshipment for simultaneously affected relations on each failure scenario:

Equation 6.51

$$\sum_{j \in S} x'_{i,j} = \begin{cases} 1, & \text{if } n = O(r) \\ -1, & \text{if } n = D(r) \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in S \quad \forall n \in N \quad \forall r \in D_i$$

2. Mutual exclusion between backup routes and their corresponding primaries:

Equation 6.52

$$x'_{i,j} \leq 1 - \zeta'_i \cdot \zeta'_j \quad \forall r \in D_i \quad \forall j \in S$$

3. Spare capacity sufficient to support application of the protection flows to backup routes:

Equation 6.53

$$\sum_{r \in D_i} x'_{i,j} \cdot d^r \leq s_j \quad \forall j \in S | j \neq i \quad \forall i \in S$$

4. Exclusion of the failure spans for each failure scenario:

Equation 6.54

$$x'_{i,j} = 0 \quad \forall j \in S \quad i = j$$

5. Flow structure variables $x'_{i,j} \leq 1$ and spare capacity variables are non-negative integers.

The "transportation-like" expression of source, sink, and transshipment of flows in [Equation 6.51](#) does not directly involve the demand magnitudes to be restored (as might normally be expected) because that would permit bundle-splitting solutions. Instead by asserting

integrality on $x'_{i,j}$ and then expressing a unit flow requirement in [Equation 6.51](#), the solution is forced to identify only a single logical route for the backup flow. In a sense this defines the skeletal structure of the backup routes. The flesh is put on the bones by [Equation 6.53](#) where the logical flow structure is weighted by the demands in generating the spare capacity requirements.

Note that because ζ_i^r is an input parameter describing the primary routes, Equation 6.52 generates a mixture of precomputable equality assertions and bounds on the $x_{i,j}^r$ variables: if $\zeta_i^r = 1$ and $\zeta_j^r = 1$, it means that when span i fails, span j must be precluded from use in restoration because (although not failed) it is another span on the primary of relation r and therefore $x_{i,j}^r = 0$. For all other spans under that failure scenario we assert only that $x_{i,j}^r \leq 1$.

The arc-flow model for SBPP can be extended to include joint selection of both primary and backup routes by defining two "types" of flow-structuring variables for each relation (one for working flow, the other for backup flow) and expressing the corresponding transshipment relationships to define the routes and mutual exclusion between the flow types on all spans.

In this case we replace parameters ζ_i^r which describe the primary routes with a second family of flow-structuring variables $x_j^{A,r}$. Any "A" variable pertains to working flow on the given relation. We correspondingly replace $x_{i,j}^r$ with $x_{i,j}^{B,r}$ backup flow-structuring variables for each relation. Now the model becomes:

Joint_arc_flow_SBPP

Minimize

Equation 6.55

$$\sum_{j \in S} c_j \cdot (s_j + w_j)$$

subject to:

1. Source, sink, and transshipment on structure variables to determine primary routes:

Equation 6.56

$$\sum_{j @ n} x_j^{A,r} = \begin{cases} 1, & \text{if } n = O(r) \\ -1, & \text{if } n = D(r) \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N \quad \forall r \in D_i$$

2. Working capacity to support primary route choices:

Equation 6.57

$$\sum_{r \in D_i} x_j^{A,r} \cdot d^r \leq w_j \quad \forall j \in S$$

3. Source, sink, and transshipment for backup routes under each failure scenario:

Equation 6.58

$$\sum_{j \in S} x_{i,j}^{B,r} = \begin{cases} 1, & \text{if } n = O(r) \\ -1, & \text{if } n = D(r) \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in S \quad \forall n \in N \quad \forall r \in D_i$$

4. Mutual exclusion between backup routes and corresponding primaries:

Equation 6.59

$$x_j^{A,r} + x_{i,j}^{B,r} \leq 1 \quad \forall r \in D_i \quad \forall j \in S \quad \forall i \in S | i \neq j$$

5. Spare capacity sufficient to support application of the protection flows to backup routes:

Equation 6.60

$$\sum_{r \in D_i} x_{i,j}^{B,r} \cdot d^r \leq s_j \quad \forall j \in S \quad j \neq i \quad \forall i \in S$$

6. Exclusion of the failure spans from use for protection in each failure scenario:

Equation 6.61

$$x_{i,j}^{B,r} = 0 \quad \forall j \in S | i = j$$

7. Flow structure variables $x_{i,j}^{A,r}, x_{i,j}^{B,r} \leq 1$ and spare capacity are non-negative integers.

6.10 Lagrangean Relaxation for Path-Oriented Capacity Design Problems

In practice we find that JCA problems for span-restorable design are fairly easily solved at high quality with route budgeting approaches at up to ~60 nodes, and span SCA problems are still manageable at 100 nodes. In contrast, it is often much more difficult to solve path-restorable or SBPP problems at even smaller sizes. Subsequent sections of the chapter are therefore devoted to approaches to obtain approximate high quality solutions for the whole class of path-restoration (or path-protection) problems. The first of these is an application of Lagrangean relaxation which was introduced in [Section 4.17](#). The key idea in Lagrangean relaxation (LR) is to take one or more of the constraint systems of the initial model and replace them with a corresponding penalty term in the objective function. The penalty is usually some weighting of the extent to which the corresponding constraints are not satisfied by the solution vector. Moreover, it is known through the theory of duality that a separate maximization problem can produce weighting coefficients for which the LR solution corresponds to the optimal solution of the unrelaxed problem. We will now develop this topic for application to the path-oriented problems of this chapter. Our basic approach follows Wynants' treatment of the LR method applied to path-restorable design problems [[Wyna01](#)].

In each of the path-oriented models we can find a "complicating" constraint which couples the restoration flow patterns (or backup route choices) for each particular failure scenario under a set of spare capacity variables that must serve for all scenarios. Aside from additional details such as modularity or jointness, the core structure of a restorable capacity problem is always a two-part constraint system expressing: (i) the restorability requirement and, (ii) the capacity requirement to support restorability. To illustrate, let us restate the basic model for non-joint path restoration:

path-SCA

Minimize

Equation 6.62

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

Equation 6.63

$$\zeta_i^r \cdot d = \sum_{p \in P_i^r} f_i^{r,p} \quad \forall r \in D \quad \forall i \in S$$

Equation 6.64

$$\sum_{r \in D} \sum_{i \in S} \delta_{i,j}^{r,p} \cdot f_i^{r,p} \leq s_j \quad \forall j \in S \setminus \{i\} \quad \forall i \in S$$

$$r \in P_i$$

[Equation 6.63](#) says that if the failure span i is on the working route for demand pair (i.e., $\zeta_i^r = 1$) then the total of restoration flow assignments to eligible routes for restoration of relation r upon failure of span i must match the demand on relation r , d^r . This is the generic "restorability" constraint system. The second constraint is the coupling or complicating one which creates interactions between the decisions about flow assignments for all failure scenarios under the spare capacity decision variables. Assuming a full mesh of logical demands, there will be $(|N|(|N| - 1))/2 \cdot |S| \cdot |P_i^r|$ restoration flow variables in $(|N|(|N| - 1))/2 \cdot |S|$ constraints. In addition there will be $|S|$ spare capacity variables in $(|S|(|S| - 1))/2$ constraints where $|N|$ is the number of nodes and $|P_i^r|$ is the average number of eligible restoration routes per relation on each failure scenario.

The corresponding constraint systems for SBPP are:

subject to:

Equation 6.65

$$\sum_{p \in P^r} x^{r,p} = 1 \quad \forall r \in D$$

Equation 6.66

$$\sum_{r \in D} \sum_{p \in P^r} \delta_j^{r,p} \cdot \zeta_i^r \cdot x^{r,p} \cdot d^r \leq s_j \quad \forall j \in S \setminus j \neq i \quad \forall i \in S$$

where variables are defined in the previous section. The appearance of the $x^{r,p}$ variables in both [Equation 6.65](#) and [Equation 6.66](#) is the same type of "complicating" linkage between constraints as mentioned for the $f_i^{r,p}$ variables which appear in both [Equation 6.63](#) and [Equation 6.64](#) for path restoration. The SBPP model has $(|N|(|N| - 1))/2 \cdot |P^r|$ 1/0 backup route choice variables in $(|N|(|N| - 1))/2$ constraints and the same dimensions for spare capacity variables as above.

Although there are more flow variables in the path restoration model, it is usual to relax these as mentioned since it is known that if the demands and capacities are integral a flow repair procedure can always construct a corresponding integral flow solution. SBPP has fewer corresponding variables but they are pure 1/0 decision variables which cannot be relaxed. Thus, the failure specific path restoration model is often easier to solve in practice than SBPP. In both cases, however, the idea in applying LR will be to replace the complicating constraint system with corresponding penalty terms in the objective function. The two LR models become:

LR-path-SCA(m)

min

Equation 6.67

$$\sum_{j \in S} c_j \cdot s_j + \sum_{j \in S} \sum_{i \in S|j \neq i} \mu_{i,j} \cdot \left(\sum_{r \in D} \sum_{p \in P'_i} \delta_{i,j}^{r,p} \cdot f_i^{r,p} - s_j \right)$$

subject to:

Equation 6.68

$$\zeta_i^r \cdot d^r = \sum_{p \in P'_i} f_i^{r,p} \quad \forall r \in D \quad \forall i \in S$$

LR-SBPP(m)

min

Equation 6.69

$$\min \sum_{j \in S} c_j \cdot s_j + \sum_{j \in S} \sum_{i \in S|j \neq i} \mu_{i,j} \cdot \left(\sum_{r \in D} \sum_{p \in P'_i} \delta_j^{r,p} \cdot \zeta_i^r \cdot x_i^{r,p} \cdot d^r - s_j \right)$$

subject to:

Equation 6.70

$$\sum_{p \in P'_i} x_i^{r,p} = 1 \quad \forall r \in D.$$

In each case the new term in the objective function "dualizes" the second constraint system. The problem-specific interpretation of the new terms is that they form a weighted total measure of the extent to which simultaneous restoration flows on each span j exceed the spare capacity of span j for each span failure scenario i . The difference by which flows exceed capacity is a measure of the infeasibility of the current s_j values under the dualized constraints. These infeasibility measures are weighted by elements of a non-negative vector of multipliers μ_{ij} . Through duality theory it is known that for any arbitrary vector of μ_{ij} values, the solution to the LR(m) problem is a lower bound on the optimal solution of the corresponding original problem. Moreover, the particular vector μ_{ij}^* that maximizes the LR(m) problem also produces the optimal solution to the original problem.

How does this work in our case? In the original problem we know that, although there is an inequality relating imposed flows to capacity for each failure scenario, there may generally be only one binding instance of this type of inequality, which sets the spare capacity on the span j . (The corresponding failure scenarios are said to be the forcers of the spare capacity on the corresponding spans where inequality is binding in the solution.) In effect, if we knew ahead of time which failure scenario was going to force the capacity for each span, we could have written a single *equality* constraint for each such binding inequality, and dropped all the nonbinding constraints. In a similar way, in our context of restorable mesh networks, each m_{ij} multiplier represents the importance of failure i in forcing the spare capacity needed on span j . If we knew a constraint was not binding in the solution, we could give the corresponding $m_{ij}=0$, since any remaining difference in its condition is not relevant to the solution. On the other hand if a constraint is binding we want $m_{ij} > 0$ so that any remaining difference from equality will be forced to zero under the minimization operator. Restated for a single span, if we consider only the spare capacity ultimately needed on span j due to all failure scenarios, then we will find that at optimality in [Equation 6.67](#) (or [6.69](#)) $m_{xj} > 0$ for x that force span j and for every non-forcer $d_{i,j}(j \neq x) = 0$. When the m_{ij} corresponding to all of the nonbinding constraints is zero, the objective function then gets no "false credit" for instances where the required flow is less than the capacity (contributing a negative value to the objective function). And at the same time, for the binding constraint(s) the sum of flows is exactly equal to the spare capacity, so the penalty term is again zero. Through these considerations we can see why:

- a. Solution of the problem $\min LR(m)$ where m is given as an input and optimization is with respect to the usual problem variables corresponds in general to an infeasible lower bound on the unrelaxed original problem, and,
- b. Solution of the problem $\max LR(m)$ with respect to m_{ij} as the decision variables produces the m^* at which $\min LR(m^*)$ is optimal for the original problem.

So how does this simplify overall solution of path-SCP or SBPP in practice? At first it looks like we have just traded the original minimization problem on flows and capacities to a maximization problem involving the same flows, capacities and now m_{ij} values as well. The key to the benefit of this approach lies in two further realizations:

- a. The $\min LR(m)$ problem is much easier to solve (for a given m) than the unrelaxed problem. The LR problem separates into two independent and easily solved subproblems, one for the flows, one for the capacities.
- b. A simple "sub-gradient" type of optimization can be used to find m^* through iteration of the relatively fast-to-solve $\min LR(m)$ problem.

In other words, our strategy to solve the original problem, denoted P , will be to solve instead:

Equation 6.71

$$\max_{\mu} \{ \min_{f,s} LR(\mu) \} .$$

The rationale is that $LR(m)$ can be much more quickly solved than P and we can use an adaptive gradient type algorithm to iterate on the elements of m to maximize the lower bound on P that $LR(m)$ gives us. When the lower bound is maximized, capacities just reach feasibility for the largest flows they support and this corresponds to the optimal feasible solution to P . The next sections will look at solving the LR-relaxed version of the original problem for a given m vector, and then how to find the m^* at which the latter is also optimal.

6.10.1 Solution Method for the LR Problem for a Given m Vector

As we have seen, Lagrangean relaxation replaces a constraint system with a corresponding penalty term and Lagrangean multiplier in the objective function. When constraints that are relaxed in this way are "less than or equal to" inequalities, and $m \neq m^*$, the objective value of $LR(m)$ will correspond to an infeasible lower bound on P . In these cases the objective value underestimates the best feasible solution because, under an arbitrary set of m_{ij} , the penalty function contributes negative values for difference terms that correspond to the nonbinding inequality constraints. Consequently it follows that there is a corresponding maximization problem that can be defined on the multipliers m_{ij} themselves. The dualized objective function $LR(m)$ will be maximized with respect to m when multipliers are such that they just extinguish any negative contributions from nonbinding constraints and have non-zero values for binding constraints which, by

definition, reach equality and therefore also contribute zero penalty to $LR(m)$ at optimality in m . In what follows, the basic procedure is to iterate between the problem of minimizing $LR(m)$ with respect to the primal variables (routes and flows etc.) and maximizing $LR(m)$ with respect to the dual variables m_{ij} . The max problem on m_{ij} and the min problem on flows and routes converge at optimality of the $LR(m)$ problem which also represents an optimal solution to the original problem P .

Wynants [Wyna01] points out that the dualized problem for non-simultaneous multi-commodity solutions for capacity and routing separate into two easily solved subproblems. To see this, reconsider Equation 6.67. By grouping together variables that share the same summation ranges, we can rewrite the $LR(m)$ objective function as:

min

Equation 6.72

$$\sum_{j \in S} \left(c_j - \sum_{i \in S | i \neq j} \mu_{i,j} \right) \cdot s_j + \sum_{j \in S} \sum_{i \in S | i \neq j} \mu_{i,j} \cdot \sum_{r \in D} \sum_{p \in P'_i} \delta_{i,j}^{r,p} \cdot f_i^{r,p},$$

which continues to be accompanied by constraint Equation 6.68 and non-negativity on flows and capacities. Inspection shows, however, that Equation 6.72 is the sum of two functions on separate variables, i.e., spare capacities and restoration flows. (When solving *min* $LR(m)$, all m_{ij} are constants.) Moreover the constraints in Equation 6.68 in the $LR(m)$ problem pertain only to the flow variables. Consequently there are no trade-offs involved between the two terms of Equation 6.72 in seeking a total minimum, so the $LR(m)$ problem can be minimized by breaking it down into two separate simple minimization problems; one of capacities and one on flows.

The problem on capacities (arising from the first term in Equation 6.72) becomes:

LR1(m)

min

Equation 6.73

$$\sum_{j \in S} \left(c_j - \sum_{i \in \{S-j\}} \mu_{i,j} \right) \cdot s_j$$

subject to:

Equation 6.74

$$s_j \geq 0 \text{ integer}$$

Remembering that all m_{ij} (and c_j) are constants at this stage this minimization problem is solved by setting:

Equation 6.75

$$s_j = \begin{cases} 0 & \text{if } \left(c_j > \sum_{i \in \{S-j\}} \mu_{i,j} \right) \\ \infty & \text{if } \left(c_j < \sum_{i \in \{S-j\}} \mu_{i,j} \right) \\ \text{free} & \text{if } \left(c_j = \sum_{i \in \{S-j\}} \mu_{i,j} \right) \end{cases}$$

It is important in the face of seeing $s_j = \infty$ etc., to remember that when we solve this problem with m not equal to m^* we are only producing a suboptimal feasible solution as a step in the iteration between $\min LR(m)$ (with respect to capacities and flows) followed by $\max LR(m)$ (with respect to m). The notation *free* in [Equation 6.75](#) means only that because of equality, the term contributes zero to the $LR1(m)$ objective function and so the variable can take on any value. As m evolves toward m^* the condition of equality in [Equation 6.75](#) arises, leaving each s_j "free" to be determined by considerations outside of [Equation 6.75](#) itself. The other criteria will be the sums of restoration flows from $LR(m)$. In effect, it is only when

Equation 6.76

$$c_j = \sum_{i \in \{S-j\}} \mu_{i,j}$$

that m has reached m^* and only the binding constraints of [Equation 6.64](#) in the original problem P then have non-zero m_{ij} in the Lagrangean objective function of [Equation 6.67](#). In fact we will shortly see that except for its role in iteration of $\max_m \{ \min_{f,s} LR(m) \}$ to converge m to m^* we do not really rely on subproblem $LR1(m)$ for a final interpretation of the spare capacities.

The second term of the regrouped LR objective function gives us the following subproblem:

LR2(m)

\min

Equation 6.77

$$\sum_{j \in S} \sum_{i \in \{S-j\}} \mu_{i,j} \cdot \sum_{r \in D} \sum_{p \in P'_i} \delta_{i,j}^{r,p} \cdot f_i^{r,p}$$

subject to:

Equation 6.78

$$\zeta_i^r \cdot d^r = \sum_{p \in P_i^r} f_i^{r,p} \quad \forall r \in D \quad \forall i \in S$$

Equation 6.79

$$f_i^{r,p} \geq 0 \text{ integer}$$

which is essentially a sum of individual minimum cost flow problems without interdependencies or capacity limits on edges. For each demand that crosses a failure span i ($\zeta_i^r = 1$) we get a total finite flow requirement to be routed at minimum cost over a graph whose cost coefficients are the m_{ij} values. The $\delta_{i,j}^{r,p}$ parameters identify the eligible routes for restoration of demand r when span i fails. These were originally needed to detect superimposed flows on each span under cases where the total restoration flow had to be split to respect finite spare capacities on spans. Here $\delta_{i,j}^{r,p}$ effectively just pick out the spans that comprise each eligible restoration path. Thus the outer pair of summations in [Equation 6.77](#) enumerate all failure span and other span combinations, selecting a specific m_{ij} coefficient to cost-weight the spans of path p for restoration of demand pair r under failure scenario i . The minimization of $LR2(m)$ thus requires making the flow assignment that minimizes cost for each demand for each failure scenario independently within the m_{ij} "cost" environment. Since there are no edge capacity constraints this amounts to solving a set of independent shortest path problems, choosing the least cost single route for each flow requirement. The actual solution information is in the assignment of restoration flow values $f_i^{r,p}$. For each set of $f_i^{r,p}$, s_j values can be generated directly by trapping the largest total of simultaneous restoration flows on each span over all failure scenarios, that is:

Equation 6.80

$$s_j = \max_i \sum_{r \in D} \sum_{p \in P_i^r} \delta_{i,j}^{r,p} \cdot f_i^{r,p} \quad \forall j \in S.$$

6.10.2 Iterative Maximization of the LR Problem on m

We can now look at an iterative procedure for finding m^* that maximizes $LR(m)$ when capacities and flows (such as given by an intermediate solution of $\min LR(m)$ above). The conditions at m^* will be such that all of the original mutual capacity constraints are satisfied and the Lagrangean penalty term in the objective function vanishes leaving the highest lower bound possible for the $LR(m)$ problem. The basic procedure will be to assume an initial m vector, solve $LR1(m)$ and $LR2(m)$ problems for capacities and flows, respectively, check for conditions of m^* as a stopping point, and update m based on the local gradient of error between flows and capacities on each span for each failure scenario.

The first consideration is to choose an initial m vector. This can be an arbitrary $m_{ij} > 0$ assignment, but inspection of the objective function for $LR(m)$ suggests a more targeted initial condition. If we consider m in the vicinity of m^* then we can make certain observations about how

m_{ij} must evolve to drive $LR(m)$ toward a maximum with respect to m . First, for any combination of failure scenario i and surviving span j where restoration flow equals capacity, the Lagrangean term goes to zero regardless of its multiplier. This corresponds to a binding constraint in the original problem that has reached feasibility. For failure scenarios and span combinations where flows are less than capacity, m_{ij} must be approaching zero as otherwise the solution is still in the lower bound region, underestimating the minimum feasible solution cost due to the negative weight contributed to the objective by the corresponding nonbinding inequality constraints. Finally, in any cases where flows exceed capacity, m_{ij} must be positive to either drive the flow down or the capacity up in $LR(m)$. This is conceptually the main way that feasibility is enforced on the original solution, through the penalty term on the objective of $LR(m)$. In this last important case we can see that there is a trade-off between terms of the $LR(m)$ objective. Adding capacity will drive toward feasibility, reducing the penalty term, but costing c_j per unit of capacity in the first term. Thus for the drive to be toward feasibility (by adding capacity in the remaining infeasible constraints) we must have:

Equation 6.81

$$c_j < \sum_{i \in S|j \neq i} \mu_{i,j}$$

and near the optimum we would expect near equality in [Equation 6.81](#) as m evolves to a final state that extinguishes the last remaining infeasibilities in binding constraints and zeros out the nonbinding constraint effects. This suggests that a m vector that solves the following

$$\sum_{i \in S|j \neq i} \mu_{i,j} = c_j \quad \forall i \in S$$

system of equations is of specific interest as a starting point

The iterative phase then proceeds as follows:

1. Initialize: $n=1$, $m^1 \geq 0$, $best_dual := -\infty$, $l=2$ is a step-size scalar, UB is an upper bound on the original problem. Also initialize a *stop_limit*.
2. Solve $\min LR(m^n)$ using $LR1(m^n)$ for the s_j capacities, $LR2(m^n)$ for the flows $f_i^{r,p}$. In this step m^n is a set of constants in the problem.
3. Where the solution to $\min LR(m^n)$ leaves an s_j value "free", in the sense of [Equation 6.75](#), assign the capacity to be the largest simultaneous restoration flow over the span over all non-simultaneous failure scenarios, i.e.,

Equation 6.82

$$s_j \leftarrow \max_i \left\{ \sum_{r \in D} \sum_{p \in P_r} \delta_{i,j}^{r,p} \cdot f_i^{r,p} \right\}$$

4. If the restoration capacity constraints of the original problem (here, [Equation 6.64](#)) are all satisfied and:

Equation 6.83

$$\sum_{i \in S} \sum_{j \in S|j \neq i} \mu_{i,j} \cdot \left(\sum_{r \in D} \sum_{p \in P_r} \delta_j^{r,p} \cdot \zeta_i^r \cdot d^r - s_j \right) = 0 \quad \forall (i,j)$$

then stop. The s_j capacities and restoration flows $f_i^{r,p}$ solve the original problem. Otherwise go on to step 5.

5. $LR(m^n)$ is the sum of the objective values from $LR1(m^n)$ and $LR2(m^n)$. If $LR(m^n)$ is higher than $best_dual$, $best_dual := LR(m^n)$. If $n > stop_limit$ without improving $LR(m^n)$, stop.
6. If iteration is to continue, calculate the "subgradient" vector which records the mismatch in flows to capacity for each span under each failure scenario. In this stage the flows and capacities are constants and the attempt is to close toward $max LR(m)$ with the maximization being with respect to m as the variables. The subgradient vector is defined as:

Equation 6.84

$$e_{ij}^n = \sum_{j \in S | j \neq i} \left(\sum_{r \in D} \sum_{p \in P_r} \delta_{i,j}^{r,p} \cdot f_i^{r,p} - s_j \right)$$

7. Update m :

Equation 6.85

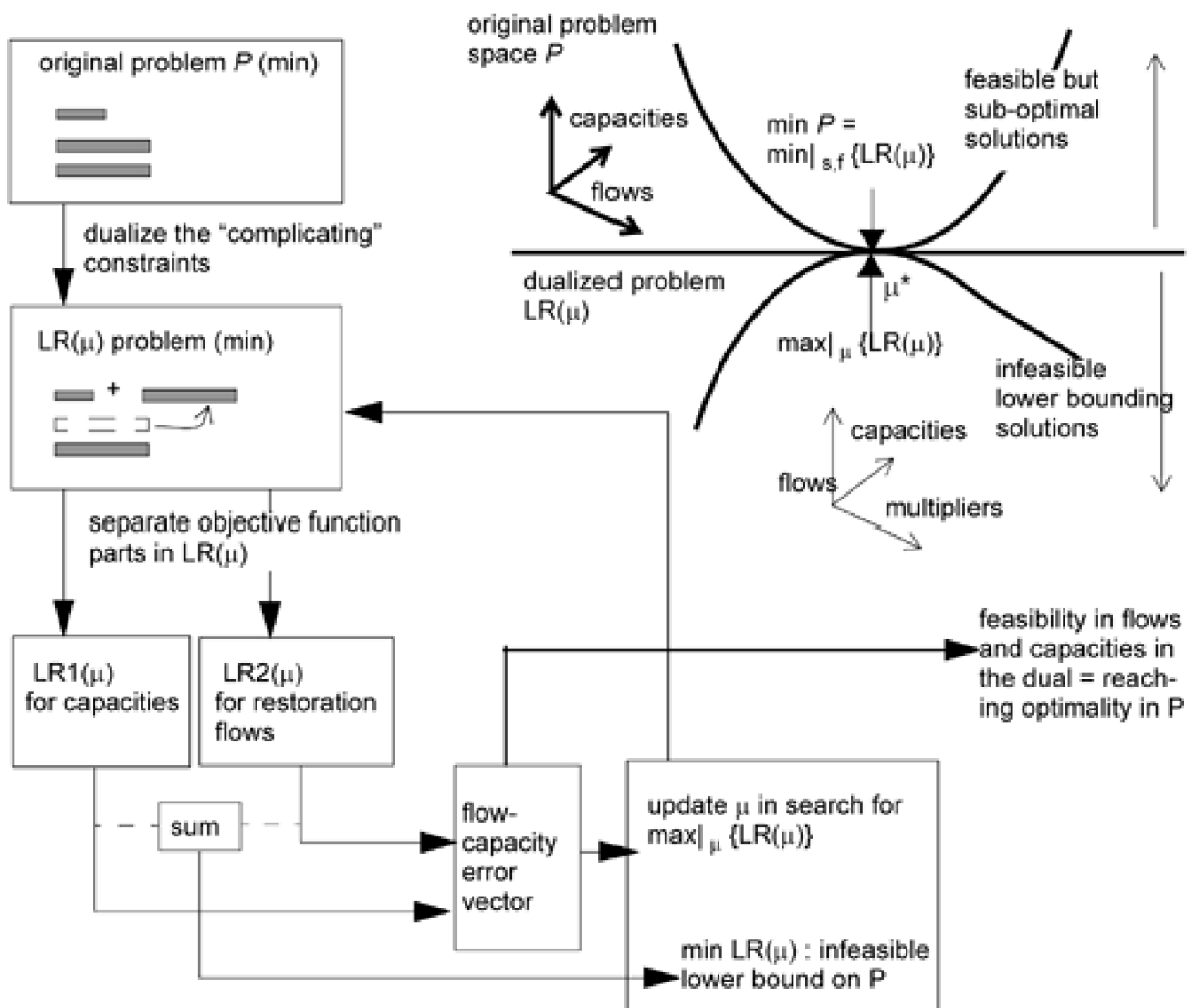
$$\mu_{ij}^{n+1} = \max \left\{ \mu_{ij}^n + e_{ij}^n \cdot \lambda \cdot \frac{UB - LR(\mu^n)}{\|e^n\|^2}, 0 \right\} \quad \forall ij$$

8. $n := n+1$, (Optionally: update step size here), Go to step 2.

In step 3, the assignment of spare capacity to the largest non-simultaneous restoration flow crossing the span gives a practical linkage between flows and capacities in $LR1(m)$ and $LR2(m)$ to speed up convergence. The subgradient vector computed at step 6 can be thought of as a multi-dimensional error vector. Where flows exceed capacities, the error is positive and drives the corresponding m_{ij} values up, penalizing the Lagrangean objective function, driving the $min LR(m)$ solution down toward optimality by eliminating excess feasibility in the dualized constraints. Where capacities exceed flows, the error vector values are negative and the update process drives the corresponding m_{ij} values toward their lower limit of zero. This contributes to maximization of the dual by eliminating negative weight from non-binding dualized constraints and driving the $min LR(m)$ lower bounding solution up toward feasibility in the binding dualized constraints. Step 7, which updates m is a standard way of effecting a multi-dimensional gradient search, suitable here because the $LR(m)$ function is concave and continuous (m_{ij} are all real). At each iteration a step is taken to increase each m_{ij} in the direction of, and proportional to, the individual error value (positive if flows exceed capacities, negative if opposite, no change for equality). The step size constant λ may be scaled down every few iterations by an adaptation strategy that considers the absolute magnitude of the most recent error vector. If it is lowering, the steps proceed more cautiously with a smaller λ . λ can also be used in a strategy to break out of possible oscillation around a convergence point. If no improvement is seen within q steps (q being considerably less than the overall stop limit), then λ is halved.

[Figure 6-16](#) provides a conceptual summary of the LR procedure and the mathematical ideas it is based upon. In dualizing the selected constraint system the dimensionality of the problem space is actually increased. In effect two higher, but continuous, dimensions are added to the complete problem space, for the multipliers on each span under each failure scenario. Within this higher dimensional space, however, the problem decomposes into two simpler subproblems for any given set of multipliers. Finally duality theory tells us that maximization of the dualized problem with respect to the multipliers, which can be done by the continuous gradient search, produces a solution that corresponds to the minimum cost feasible solution in the original problem space. Note that although we used the path restoration as a vehicle to work through the overall procedure above, the steps for application to the solution of SBPP problems are completely parallel. The computational advantage on the SBPP case may be considerably greater because the original SBPP problem involves $(|N|(|N|-1))/2 \cdot |P^f|$ 1/0 decision variables in inter-linked constraint systems. The corresponding LR model breaks this into simplified $LR1$ and $LR2$ problems effectively replacing a difficult search on a 1/0 combinatorial decision space with a continuous gradient type search in Lagrangean multiplier-space.

Figure 6-16. Conceptual overview of the Lagrangean Relaxation approach to solving path restoration or SBPP capacity design problems.



6.11 Heuristics for Path-Restorable Network Design

In this section we consider heuristic methods for approximate solution of path restoration or SBPP capacity and routing design problems. Our aim in this last part of the chapter is to provide an overview and extension of the main ideas and tactics that have been applied to these problems in various forms in the following prior works:

- Xiong and Mason "MCR Algorithm" [[XiMa99](#)]
- Kawamura "Backup VP Path Protection" [[KaSa94](#)]
- Wynants "Greedy heuristic" [[Wyna01](#)](p. 108)
- Pióro "Simulated Allocation" [[PiGa97](#)]
- "Max-latching" [[GrRa97](#)]
- "Design Tightening" [[GrBi91](#)]
- Lee "Modular aggregating routing" [[LeGr99](#)]
- Morley "Demand packing" [[Mor01](#)]
- Successive survivable routing [[Liu01](#)], [[LiTi01](#)], [[LiTi01b](#)]

Similar basic concepts recur, but in different detailed contexts, in each of these works. Rather than summarize each paper individually, our aim has been to distill the central ideas and principles from all these works and bring them to bear in a single overall heuristic strategy, within which we can use different techniques on the subproblems that are involved. We summarize the key high-level ideas from all the above works as follows:

1. We should route any new restoration flow requirements so as to first take advantage of any previously decided capacity placements for other non-simultaneous failures.
2. We need to minimize the forcing of additional capacity to support restoration flows above the capacity already placed. For example, a small addition to one span might link together other existing capacities for use in restoration paths. This (as well as the first statement) falls under the broad principle of "forcer minimization."
3. When generating capacity requirements by successive routing, we need to explore the order-dependence of handling failure scenarios and routing for affected demands to find solutions with a lower capacity cost. This recognizes that almost any heuristic that "constructs" capacity based on routing has an order dependency when such capacity is shared over different failure rerouting scenarios.
4. Following any initial feasible design, random search methods can seek further improvements by releasing, reassigning or rearranging certain flows. This is the broad principle of "design tightening."
5. Where capacity is modular, we should bias routing processes to avoid module exhaustion and, periodically during design development, we should seek opportunities to "pack" flows from any small or lightly loaded modules into the residual capacities of other large and better-loaded modules. This is the broad principle of "economy of scale" or modular aggregation.

We now outline a generic heuristic framework within which remaining sections provide detailed options for each stage, being guided by and addressing the five main principles above. The overall approach is to first develop a feasible suboptimal solution produced by construction through a sequence of minimum incremental-cost flow addition problems for each failure scenario. The basic process of producing a feasible solution by construction we call Phase 1. A variety of processes for seeking improvements on the initial feasible design then aim at reducing capacity while retaining feasibility, we call this Phase 2. Phase 2 is fundamentally optional in that it seeks only to improve the quality of the feasible solution from Phase 1. At the highest of overview levels the basic heuristic framework is:

Phase 1: Design Construction

- Select a failure scenario, i .
- Select a demand pair, r , affected by failure i .
- Find a restoration routing solution for the total demand affected, d^r , such that the cost of capacity *added* at this stage is minimal.
- Spare capacity already placed for a prior (non-simultaneous) failure scenario may be first reused before adding new capacity, but capacity added for other demand pairs affected in the *same* failure scenario cannot be reused.
- Repeat until all demand pairs affected under each failure scenario have been considered.

Alternate Phase 1

An alternate approach to Phase 1 is also introduced in the following sections. It constructs a feasible solution somewhat more directly, using an ILP or LP *subproblem* for the multi-commodity routing in each failure scenario, but leaves the multi-scenario optimization with sequence dependency. The basic procedure is:

- Select a failure scenario, i .
- Solve a minimum incremental cost multi-commodity flow problem for all demand pair requirements affected by failure scenario i . Any already placed capacity can be reused.
- Repeat until all failure scenarios are considered.

Phase 2: Design Tightening

- Seek improvements to the design. An improvement is defined as any change in the assignment of restoration flows or backup paths (and more generally any changes to the working routes as well) that retains the target level of restorability but permits removal of capacity. The entire Phase 2 is strictly optional depending on the required solution quality and computing time available.

The following sections discuss each of these phases and the options for their implementation in more depth.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.12 Phase 1 Heuristics—Design Construction

Let us first consider the steps involved in the basic Phase 1 Construction heuristic above. In some cases there may be a certain minimum capacity already present on spans, but the more general case is that there will be no existing spare capacity when the first demand pair of the first failure scenario is considered for restoration routing. In this case the minimum cost solution is to follow the shortest path over the graph (with exclusion of spans in the failure scenario) and generate d units of spare capacity on each span of that route. If initial capacity exists, the starting iteration simply takes this capacity into account as it would below in one of its own subsequent iterations.

6.12.1 Modeling Existing Capacity

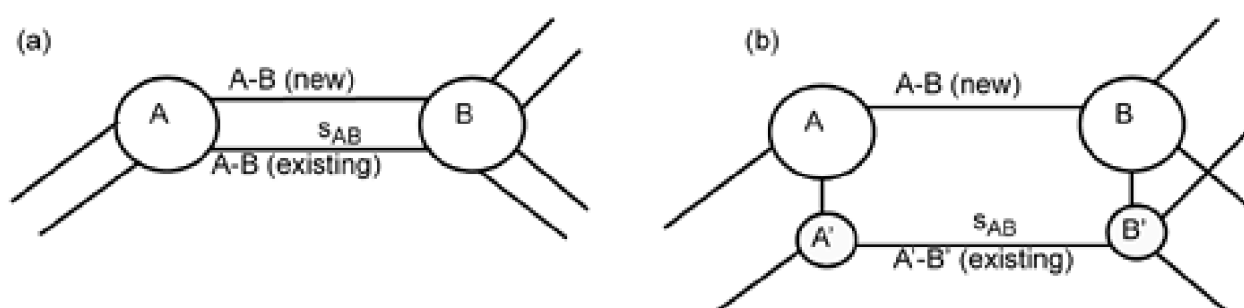
All subsequent routing and capacity augmenting subproblems will be influenced by existing capacity, i.e., capacity that we have already committed to the design to accommodate the routing of flows for previously visited failure scenarios. Existing capacity complicates the process of finding a route of least incremental cost with either a shortest path algorithm or a minimum cost flow model because the edge costs effectively become nonlinear in capacity. By this we mean that if an edge j has existing capacity s_j and new additions of capacity to that edge cost c_j^+ per unit, then the incremental cost model to support a flow f over j is:

Equation 6.86

$$c_j(f) = \begin{cases} 0, & \text{if } f \leq s_j \\ (f - s_j) \cdot c_j^+, & \text{if } f > s_j \end{cases}$$

Since the idea of reusing spare capacity placed for one failure scenario to wholly or partly address other restoration flow needs in other scenarios is so central to capacity efficiency, we need to represent this effectively nonlinear cost of routing at each iteration. One way to represent this effect of existing capacity for use with a shortest path algorithm (or for solution with LP / IP models), is to represent *two* edges between every pair of adjacent nodes in the graph. One edge will provide cost-free but finite capacity. The other has no limit on capacity but use is charged at the incremental cost rate. [Figure 6-17\(a\)](#) illustrates. In this environment an ordinary min-cost flow model or successive shortest path routing for each demand unit can be easily employed and will find the incremental routing pattern which minimizes added cost by reusing existing span capacity where possible.

Figure 6-17. Representing existing capacity in minimum incremental cost routing problems.



Cost	Capacity	
$c_{A-B(\text{existing})} = 0$	$f_{A-B(\text{existing})}$	$\leq s_{AB}$
$c_{A-B(\text{new})} = c^+_{A-B}$	$f_{A-B(\text{new})}$	$\leq \infty$

Cost	Capacity	
$c_{A'-B'(\text{existing})} = 0$	$f_{A'-B'(\text{existing})}$	$\leq s_{AB}$
$c_{A-B(\text{new})} = c^+_{A-B}$	$f_{A-B(\text{new})}$	$\leq \infty$
$c_{A-A'} = 0$	$f_{A-A'}$	$\leq \infty$
$c_{B-B'} = 0$	$f_{B-B'}$	$\leq \infty$

A practical problem may sometimes be that representation of distinct parallel spans with the same end nodes is not supported in all design files or software packages. If so one can create a pseudo-node that is connected at zero cost (and unlimited capacity) to each actual network node. The pseudo-node terminates the otherwise parallel edges used to represent existing capacity. This is illustrated in [Figure 6-17\(b\)](#) accompanied with the corresponding cost coefficients that would apply and capacity bounds that would be asserted for their representation.

6.12.2 Minimum Incremental Cost Routing

Now that we can represent a mixed environment of zero-cost but finite existing capacities and unlimited potential for added capacity at finite augmentation costs, the rest of the basic Phase 1 algorithm can be described. Let us assume that the type of dual-edge model in [Figure 6-17\(a\)](#) is used. Failure scenarios will continue to be indexed by i and spans in general by j . Existing capacity on spans will be

denoted by $s_j \in S^+$ and incremental capacity additions by $s_j^+ \in S$ where S^+ is the set of artificial spans added to represent existing capacities and S is the ordinary set of spans that can sustain any capacity augmentation at incremental cost. Index j indexes the union set of all logical spans. For each O-D pair r under the current failure scenario i we then solve a *minimum incremental cost network flow* (MIC_NF) problem, which can be expressed in the following transportation-like formulation:

MIC_NF(i,r)

Minimize

Equation 6.87

$$\sum_{j \in S} c_j^+ \cdot s_j^+$$

subject to:

1. Single-commodity source, sink, and transshipment:

Equation 6.88

$$\sum_{j@n} f_j^r = \begin{cases} d^r, & \text{if } n = O \\ -d^r, & \text{if } n = D \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N$$

2. New capacity added after using existing capacity on spans:

Equation 6.89

$$s_j^+ = f_j^r - s_j \quad \forall j \in S$$

3. Zero flow on failed span (in general all spans of the failure scenario):

Equation 6.90

$$f_i^r = 0$$

Equation 6.91

$$f_j^r \geq 0, s_j^+ \geq 0, \text{ integer}$$

In MIC_NF, n denotes a node and N is the set of nodes. $j@n$ means all spans j incident on node n . f_j^r is the amount of flow assigned to span j . Span i is the failure scenario and relation r is the specific demand pair affected by this scenario for which we are currently providing a restoration path. Note that [Equation 6.89](#) cannot go negative because of the separate bound on s_j^+ in [Equation 6.91](#). Thus, the capacity added to each span at this stage is:

Equation 6.92

$$s_j^+(r) = \max(f_j^r - s_j, 0) \quad \forall j \in S.$$

This capacity is *not*, however, immediately available for reuse in restoration of the next demand pair *under the same failure scenario* because these are simultaneous requirements. At the same time as a result of running MIC_NF(i, r) the existing capacity *used up* at this stage is:

Equation 6.93

$$\min(s_j, f_j^r) \quad \forall j \in S$$

and this is capacity that is no longer available for the routing demands of another node pair r under the *same* failure scenario, i . Therefore, as we solve the succession of MIC_NF(i,r) problems for a given i we use up existing spare capacity which must be reflected immediately at the next iteration on r for a given i , but we also place new spare capacity which *is not* available for reuse until the next complete failure scenario is considered. Thus, the overall heuristic procedure using MIC_NF(i,r) to build up a feasible spare capacity solution is as follows:

path_rest_greedy_1

Enter with an initial set of s_j (which may be zero).

Initialize $s_{totj}=s_j$ on every span j ;

For every failure scenario i ;

$s_j=s_{totj}$ for every span; (all accumulated spare is available
at the start for the new scenario)

For every demand pair r affected in scenario i ;

Solve MIC_NF(i,r) with existing spare s_j ;

$\min(s_j, f_j^r) \quad \forall j \in S$

Update existing spare $s_j:=s_j-$

Update total capacity present $s_{totj}:=s_{totj}+s_j^+ \quad \forall j \in S$

end for;(all affected demands in current scenario)

end for (all failure scenarios).

It is important to note as mentioned that s_{totj} is being updated in each inner iteration (for individual demand pairs) within the current failure scenario but that this capacity will only be available for *reuse* on the *next* failure scenario. Within the current scenario the restoration routing for individual demand pairs sees the total spare capacity built up by previous failure scenario iterations but thereafter only ever reduces remaining spare capacity as more restoration flow is assigned for each affected demand pair. Under different terminologies and detailed contexts Wynants [Wyna01], Grover [GrRa97], Xiong and Mason [KiMa99], Liu [LiTi01] and Kawamura [KaSa94] have all experimented with this form of this basic constructive heuristic. By its constructive nature, it always results in a feasible solution for path restoration but the total spare capacity is in excess of optimal and depends mainly on the order in which failure scenarios are considered, and secondarily on the order of processing the affected demands under each scenario. Results within 15% of optimal are, however, fairly common if:

1. Failure scenarios are handled in *increasing* order of the total amount of affected demand in each scenario (i.e., fewest number of failed paths first).
2. Restoration flow requirements are handled in *decreasing* order of the demand volume within each scenario (i.e., the relation with the most failed paths is handled first).

The reason for taking the "lightest damage" scenarios first is that if we consider starting with no existing spare capacity, restoration flows will tend to take the single shortest-path routes with all restoration flow piled on the same route. There is nothing to *explicitly* assert the splitting and spreading of the total restoration flow requirements for each O-D pair, which we know is key to spare capacity sharing. What will, however, *implicitly* result in bundle splitting and flow spreading is the accumulation of existing spare capacity as more failure scenarios are handled. For this reason if we first handle the lighter failure scenarios first, we lay down some spare capacity that attracts the later and larger scenarios to spread out their restoration flow more widely over the network. If the converse was done, more localized accumulations ("ridges") of spare capacity would be established early to serve the large simultaneous flows. Such localized accumulations are then less efficiently reused under the later scenarios to be considered.

However, *within* each failure scenario, the opposite orientation is found to work best; that is to say that we provide restoration rerouting for the largest individual demand quantities first. When existing capacities are already available from previously handled failure scenarios, this tends to avoid inefficient mutual capacity conflicts among the simultaneous flows within the current failure scenario. The point is that if a small demand is satisfied first, it may have several routing options, but it unknowingly takes a route that uses up key capacity on one or more spans that forces a significantly longer route with more new capacity establishment for some larger flow. These are the issues about multi-commodity simultaneous flow problems explained previously in [Figure 6-4](#). The idea, therefore, is to first establish relatively efficient restoration paths for the largest demand bundles. By "largest" we also mean in the sense of the (demand x distance) product as the

ranking criteria for each affected bundle of demand flow. Thus right at the start of a new failure scenario iteration we let the most distant and heavily affected node pair "see" the initial environment of existing spare capacity for its reuse first. Once the largest and longest flows are efficiently restored, closer range and smaller flows can be accommodated around them, rather than the opposite. This is especially important in the presence of modular capacity additions.

As so far presented, we have specified and discussed use of the MIC_NF(i,r) subproblem as a simple min-cost network flow problem suitable for solution as an LP. In many cases it may be more convenient to have MIC_NF(i,r) implemented by ordinary shortest path algorithms instead and this is what we consider next.

6.12.3 Iterated-Dijkstra to Solve MIC_NF

Where the restoration flow requirements and capacities are integer the single-commodity minimum incremental cost flow problem above can be solved by LP as a network flow problem, or by iterative application of a shortest path algorithm where the appropriate edge weights are updated after each iteration. In the following, a basic shortest path (SP) algorithm finds a least cost unit-capacity path on each iteration within the graph handed to it by the edge-cost updating routine. Thus, where previously in *path_rest_greedy_1* we had:

Solve MIC_NF(i,r) with existing spare s_j ;

$$\text{Update existing spare } s_j := s_j - \min(s_j, f_j^r) \quad \forall j \in S$$

$$\text{Update total capacity present } s_{totj} := s_{totj} + s_j^+ \quad \forall j \in S$$

we expand this to an implementation as follows:

```
For every unit demand on relation  $r$ :
  For every span (not  $i$ ): (set up the edge costs for the next
    path to be least-cost routed)
    If  $s_j > 0$  then  $edge\_cost(j) := 0$ 
    otherwise  $edge\_cost(j) := c_j^+$ ;
    Call SP algorithm (edge_costs);
    For every edge on route found: (update spans)
      If  $edge\_cost(j) < 0$  then  $s_{totj} := s_{totj} + 1$ ; (added cap.)
      If  $edge\_cost(j) = 0$  then  $s_j := s_j - 1$  (used exist cap.)
    end for;
  end for;
end for.
```

Note that this works to exploit existing capacity while also adding new capacity as needed, without requiring representation of dual parallel edges to effect the nonlinear capacity versus cost model. Because we iterate the SP algorithm once for each individual demand to be routed, we just decrement s_j on each span where existing spare capacity is used by the currently routed path, and increment s_{totj} on other spans. It is then simple to see when existing capacity on a span is fully used and to switch the edge cost associated with that span to indicate on the next iteration that there is no more existing spare to be used on a span. For the same reason we can simply write $s_j := s_j - 1$ instead of needing $\max(s_j - 1, 0)$ because when handling each path individually (not the entire flow for an affected demand pair at once), we do not run the risk of s_j going negative if the total restoration flow assigned to span j exceeds the s_j .

6.12.4 Alternate Phase 1 Algorithm

It is fitting that the considerations of "mutual capacity" returned to our attention in discussing the order of handling restoration flow assignment for affected demand pairs within each scenario. At the heart of the alternate Phase 1 algorithm is the use of an optimally solved subproblem for the routing of flows that occur simultaneously within one failure scenario, so as to eliminate at least one form of order dependency from the overall procedure. Instead of solving greedily (and hence in an order-dependent way) for both the failure scenarios *and* demand requirements under each scenario, we will solve a subproblem called minimum *incremental cost multi-commodity network flow* (MIC_MCNF) for each failure scenario as a whole. The advantages in terms of solution quality are that we eliminate one of the levels order dependence and harness an optimal process for addressing mutual-capacity effects and for reusing capacity from one scenario to the other. With this approach the search space for Phase 2 approaches that seek design improvements is also reduced to searching in the $|S|$ -dimensional scenario space, which is much smaller than the $|S|$ by $O(N^2)$ dimensional space of all scenarios and routing for all node pairs. On the other hand, solving an MCNF type problem for each failure scenario may take more time than greedily rerouting every affected demand with single-commodity MIC_NF. In this particular context there are, however, several things we can do to manage the MIC_MCNF run times because we can still have a good heuristic without requiring optimal termination to each failure-scenario subproblem. We will discuss these options further after considering the incremental cost MCNF subproblem and the revised Phase 1 heuristic procedure that uses it. At some problem size, however, direct optimal solution of the per-scenario MIC_MCNF subproblem would start to take longer than order-dependent iteration above, so this alternative approach just provides an additional option to consider depending on problem size.

The subproblem to be solved for each failure scenario is that of simultaneously rerouting demand for all affected node pairs with a minimum of additional cost of capacity, given a set of existing capacity quantities to start with which may be reused. The same model for existing and new capacity and costs applies as for MIC_NF above. In addition we will let:

- D_j = the set of O-D pairs requiring restoration in failure scenario i , index r .
- $O(r)$ = the origin node of demand pair r .
- $D(r)$ = the destination node of demand pair r .

MIC_MCNF(i)

Minimize

Equation 6.94

$$\sum_{j \in S} c_j^+ \cdot s_j^+$$

subject to:

1. *Multi-commodity* source, sink, and transshipment for simultaneously affected O-D pairs:

Equation 6.95

$$\sum_{j @ n} f_j^r = \begin{cases} d^r, & \text{if } n = O(r) \\ -d^r, & \text{if } n = D(r) \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N \quad \forall r \in D_i$$

2. Incremental capacity generated on spans:

Equation 6.96

$$s_j^+ = \sum_{r \in D_i} f_j^r - s_j \quad \forall j \in S$$

3. Zero flow on failed span (or all spans in failure scenario):

Equation 6.97

$$f_i^r = 0 \quad \forall r \in D_i$$

Equation 6.98

$$f_j^r \geq 0, s_j^+ \geq 0 \quad \text{integer (or relaxed)}$$

Using MIC_MCNF, the overall Phase 1 heuristic becomes:

path_rest_greedy_2

Enter with an initial set of s_j ;

For every failure scenario i ;

 Solve MIC_MCNF(i) with existing spare s_j ;

 (Optionally): repair non-integral capacity from relaxations

 Update total capacity $s_j := s_j + s_j^+$ for all ($j \neq i$)

end for.

This generates a feasible capacity design from a single round of MIC_MCNF(i) problems; one for each failure scenario. Order dependence remains, but it is now only a dependence on the order of handling whole failure scenarios. The strategy to reuse spare capacity already decided upon for previous failure scenarios, and the minimum cost capacity augmentation to serve each scenario is solved with an optimal model for the intra-scenario multi-commodity routing subproblem. Depending on run-time considerations each MIC_MCNF subproblem may also be time-limited, stopped shortly after feasibility is reached, or run with a larger MIP gap and/or run with relaxed flow variables. As mentioned in [Section 4.12.1](#) with ILP the solution quality shortly after reaching the first feasible solution is often very good or even optimal, but then may run a long time to reach a full termination which in effect only proves the optimality. So here, as part of an overall Phase 1 heuristic, there is no real need to run the MIC_MCNF subproblems to full termination. In addition there are highly evolved solution strategies for solving problems with transshipment type network structure, including multi-commodity problems, so there are thus many avenues available to trade run-time for solution quality with path_rest_greedy_2.

As long as feasibility is reached in each subproblem before stopping, the result will at least be a feasible solution. If flow and capacity relaxations are used within MIC_MCNF, however, it would be better to "repair" any non-integral capacity results following each iteration, rather than attempt a single repair phase at the end. By repairing inside the loop we know that even if we simply round up, any excess capacity that this might strictly create for the current scenario will still be optimally exploited in the subsequent scenarios. In relaxing flows (but not capacities) in MIC_MCMF, however, we also know that not many non-integral flows will arise because the demand requirements are integral as well as the existing and added capacities.

6.13 Putting Modularity Considerations in the Iterative Heuristic

From one standpoint where heuristics are concerned, modularity is sometimes a welcome reality because suboptimality in the exact integer capacity requirements is then less important. Small errors in integer capacity assignments tend to be washed out if the final result is simply mapped into modular totals. On the other hand, in a modular capacity environment it can be important to realize economy of scale opportunities by aggregating routings, and to eliminate lightly loaded capacity modules entirely from the design. In [Section 5.6](#) and [Section 6.6](#) we saw how to incorporate modularity, with economy of scale effects, into optimal solution models for both span and path restoration, respectively. In the results on span restoration we also saw that strong modularity effects are not adequately captured by simply post-modularizing an integer design. Significantly more cost effective overall designs result if the economy of scale effects associated with modularity are brought right *into* the design problem, such as with MJCA or path-MJCA. What is not clear cut, however, is how to bring such modularity effects into account when we are using algorithmic incremental routing procedures to place capacity, such as in `path_rest_greedy_1`. One area where modularity has been studied in incremental capacity design heuristics is ring-based network design where the concept of *modular aggregating prerouting* (MAPR) has been developed [\[Lee99\]\[LeGr99\]](#). Let us now see how that approach can be integrated into the least incremental cost type of shortest-path procedure.

6.13.1 Concepts of Aggregating Prerouting and Pseudo-Cost Functions

The central issue is that a succession of individual shortest-path routing processes, or even MIC_NF routing, will generally not result in a distribution of the capacity requirements that are good fits to the available modular transmission capacities that can actually be provisioned. It is even less likely to result in a composite routing pattern that is well matched to a minimum cost selection of modules with economy of scale benefits taken into account. To exploit economy of scale benefits, routings generally must coordinate themselves so as to collectively warrant purchase of the largest but proportionately cheapest module sizes. But there is nothing to effect such coordination in routing each demand individually. In addition there is nothing to discourage the basic routing criterion from placing a single unit flow on a span, requiring a whole module to be provisioned in the solution. A person inspecting such a design would easily see how that whole module could be eliminated by rerouting the one requirement. But the question is how to get a mechanized unit-capacity routing program to build up capacity iteratively in such a way as to avoid this effect, or more generally, to avoid blindly crossing modular capacity boundaries, triggering additional lightly loaded modules.

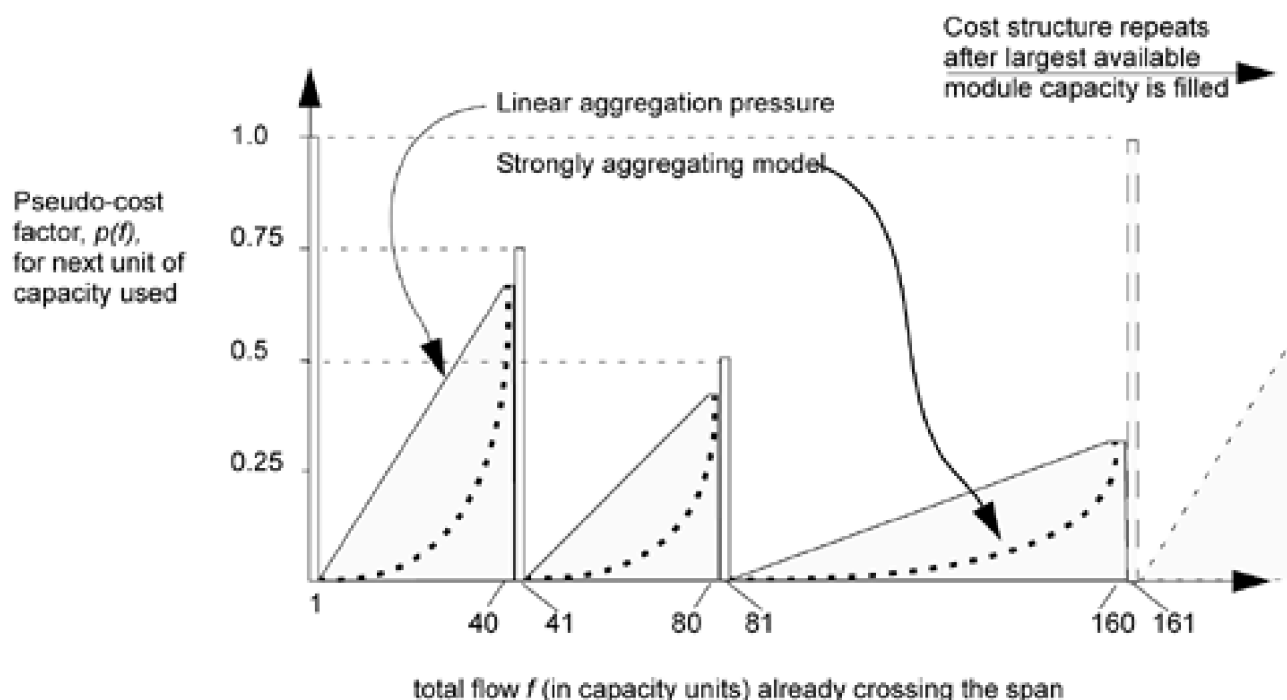
Although developed for design of networks employing multiple highly modular rings, MAPR itself is a general way to route demands in a manner that tends to aggregate flows on spans into total accumulations that are well-suited to a family of available system modularities. The basic idea is that once a span has some demand on it, it is important to make it look more attractive to subsequent least-cost routing determinations, but to cancel that effect as the capacity accumulation approaches what would be a modular capacity boundary. MAPR does this through the use of capacity-dependent pseudo-cost functions.

The pseudo-cost assigned to each span is determined by its length and its current capacity requirement based on demands so far routed over the span. Many pseudo-cost functions can be considered. In the work of Lee [\[Lee99\]](#), the *rise* pseudo-cost function (described below) was found to be the most effective general approach to effect the idea of biasing the routing to try to create good fill-levels in the anticipated modular sizes but to defer triggering whole new module requirements. An important point is that economy of scale only produces a *realized* benefit if the load is actually present to fill the larger modules, so the pseudo-cost approach wants to aggregate flows so they are good fits to the corresponding module sizes. The basic *rise* pseudo-cost function assigns a maximum routing cost to a span when it has zero capacity accumulation or when it reaches a capacity accumulation that is a multiple of the modularity constant M or any specific modularity threshold if the modular capacities do not all derive from a common factor. Specifically, if the modularity family is not regular, the cost is highest whenever the first unit of capacity of a new module would be implied. At the other extreme, as soon as the span has one more demand unit crossing it than one of the module capacities, the cost is set at its lowest value. The logic is that in this state an extra module is implied anyway, so it makes sense to make this span more attractive to subsequent routing decisions. In effect the tour bus has been chartered but there's only one passenger on it so far. Since the commitment to pay for it has already been made, it might as well be loaded up. Between the extremes on either side of just nearing and just passing a module boundary, the pseudo-cost rises linearly, steadily reducing the attractiveness or aggregation pressure on this span as it rises toward the next full module total.

One can fairly easily work out detailed pseudo-cost functions that reflect a considerable intuition about routing to exploit modular

capacities and economy of scale, including the fact that several module sizes may be available. As a general example [Figure 6-18](#) shows a pseudo-cost function for a planning environment where, for example, DWDM systems of 40, 80 or 160 wavelengths are available at relative module costs of 1, 1.5 and 2 respectively (implying per-channel fully-allocated relative costs of 1.0, 0.75 and 0.5).

Figure 6-18. Example of a multi-modular pseudo-cost function with economy of scale.



In the example a deliberately high impulse-like barrier is provided to routing decisions that would trigger establishment of the next module requirement. But once that threshold is passed, the apparent routing cost drops to its lowest. To reflect economy of scale in larger module sizes the basic barrier plus rise cost behavior is repeated but scaled in absolute value according to the economy of scale. The linear pseudo-cost function discourages further use of the span in direct proportion to its current use. Experience in [\[Lee99\]](#) shows this tends to be a strongly flow-leveling and dispersive routing model which leads to designs that embody relatively more module placements and more unused module capacity. This is in comparison to the more "strongly aggregating" pseudo-cost curves illustrated as in [Figure 6-18](#) that tends to result in fewer, but higher-fill, modules in the completed designs. Families of exponential functions can be worked out to implement variable aggregation pressure [\[Lee99\]](#).

Another important consideration in addressing the design problem with modularity is that ideally the working flows, not just restoration flows, should be subject to the same variable cost routing picture because the module decisions we will ultimately provide the capacity to support both working *and* spare capacity requirements. In other words, the best choice of modular capacity placement for a span with a requirement of, say, 25 *spare* capacity units may be considerably different if it also has 100 units of *working* flow crossing the span. To revisit the first heuristic with modularity in mind we have, therefore, to first revisit the single-commodity minimum incremental-cost routing model and see how it changes with modularity.

6.13.2 Modular Minimum Incremental Cost Network Flow (mod_MIC_NF)

Under modular capacity provisioning there are arguably two different aims that one could define for the minimum incremental cost network flow problem. In a standalone context of a single incremental requirement to be routed one would state the aim as "route the additional flow with the least cost of modular additions." In this view all existing unused modular capacity would be without cost and the objective function would try to avoid triggering additional whole modules to bear the incremental flow. There are, however, several problems with this in the context of repeated iteration to build up the spare capacity needed by each failure scenario in a Phase 1 heuristic. First is that the model will be too easily able to resist module additions, by implementing tortuously indirect routings to exploit existing capacity. Each use of a unit capacity in an already placed module needs therefore to reflect some cost for its use because in the larger picture, judicious use of already placed modular capacity avoids greater overall accumulation of modules in the longer run. In other words, we are saying that the

goal of "adding the least new modular capacity" at each routing iteration is too greedy. Rather, we need to incorporate pseudo-cost ideas for the use of existing modular capacities as well as cost for triggering explicit module additions. A modular single-commodity routing model that corresponds to MIC_NF(i,r) is therefore proposed as follows.

First, to represent modularity, let us reuse the following from [Chapter 5](#):

- $M=A$ set of modular capacity sizes, index m .
- Z^m =Number of capacity units in the m^{th} module size.
- C_j^m =Cost of a module of the m^{th} capacity size installed on span j . Economy of scale is introduced through these coefficients.

In addition, we need to add the following new inputs to the problem:

- w_j = working capacity on span j arising from routing of working demands.
- f_{ij}^{accum} = restoration flow on span j accumulated under prior routing iterations for the *current* failure scenario i .
- $P_j^M(f)$ =the pseudo-cost function for span j as a function of existing flow on the span, for the modularity environment M .
- n_j^m = Number of modules of the m^{th} size already decided to exist on span j from the *prior* failure scenarios considered already.

And let us also define a minimum-cost modular capacity allocation function:

$$\langle m^*(f) \rangle \equiv \operatorname{argmin} \left\{ \left(\sum_{m \in M} C^m \cdot n^m \right) \mid \sum_{m \in M} n_j^m \cdot Z^m \geq f \right\} \text{ is a function that returns}$$

the set of module choices that provides a total capacity of at least f at minimum cost.

To clarify regarding $\langle m^*(f) \rangle$, the point is that under a general cost regime for different modular capacities, the smallest module that exceeds the capacity requirement is not always the least cost way to serve the requirement. In general, for a requirement of, say, 50 capacity units in an environment of $M = \{20, 40, 80, 160\}$, the least cost modular provisioning solution could be any of (40 + 20), 80, or 160, simply depending on the relative module costs. The function $\langle m^*(f) \rangle$ will therefore generate least-cost module placement decisions following each failure-scenario iteration below. Also, in the context of the same set of modularities M just mentioned, an $\langle m^*(f) \rangle$ vector = <0, 2, 0, 1>, for example, would specify a capacity composition of 2 modules of size 40 plus one of size 160.

Modular minimum cost incremental network flow (mod_MIC_NF(i,r)) is then defined as follows. The model is again an arc-flow type of

model and the solution variables, as in MIC_NF(i,r) and we are solving primarily for f_j^r —the amounts of flow assigned to span j for relation r . The intent is to use mod_MIC_NF(i,r) on each iteration for a new demand bundle within each failure scenario to find a routing (for one relation at a time) that minimizes the *incremental pseudo-cost* incurred for new capacity arising from the routing chosen, recognizing that certain capacity modules already exist from prior failure scenario iterations and that flows may have already accumulated for other relations under the same failure scenario (the f_{ij}^{accum} values) before entering mod_MIC_NF(i,r) for the current relation r . Parts of these modules will be essential to support working flows but in the context where this problem is solved, there is no cost for using any unused modular capacity for restoration under the current failure scenario.

mod_MIC_NF(i,r)

Minimize

Equation 6.99

$$\sum_{j \in S} c_j^+ \cdot f_j^+ \cdot p_j^M [f_j^r + w_j + f_{ij}^{accum}]$$

subject to:

1. Single commodity source, sink, and transshipment: i.e., [Equation 6.88](#).
2. Incremental flow subject to pseudo-cost after reuse of already placed modular capacity from prior failure scenarios:

Equation 6.100

$$f_j^+ = f_j^r - \left(\sum_{m \in M} n_j^m \cdot Z^m - (w_j + f_{ij}^{accum}) \right)$$

3. Zero flow on failed span(s), non-negativity and integrality (as in [Equation 6.91](#)).

To understand the objective function note that $p_j^M(f)$ is the modular pseudo-cost function and that $f_j^r + w_j + f_{ij}^{accum}$ is just the *total* flow on span j , used as the argument to determine the pseudo-cost "bias point" against which actual cost is accrued only by the amount of the *incremental* new flow required to be supported by new capacity on span j , i.e., f_j^+ —which is multiplied by $p_j^M(f)$. Note also that for ILP/LP type solvers the pseudo-cost function $p_j^M(f)$ would have to be implemented with the piece-wise linear approximation technique, which can increase run times. Also, to accurately apply pseudo-cost and avoid crossing module boundaries a bundle of flow, d should not really be routed all at once over a single route. Rather each unit of flow in d is preferably routed by an individual instance of the modular MIC_NF model. An alternative is to route the bundle as a whole assigning the pseudo-costs of $d/2$ (i.e., the "median" demand of the bundle) to each span en route. However, for moderate to large d relative to the module boundaries this would allow new-module triggering effects without reflecting the true pseudo-cost of doing so. For these reasons, the approach of using an algorithmic shortest path routine may in practice be preferable to solving modular MIC_NF with ILP methods. With these modularity considerations, the first heuristic procedure above ([Section 6.12.2](#)) is changed to:

mod_path_rest_greedy_1

Enter with an initial set of w_j and modules n_j^m (the latter may be null).
 For every failure scenario i ;

Initialize $f_{ij}^{accum} = 0$ for every span j ;
 For every demand pair r affected in scenario i ;
 solve mod_MIC_NF(i,r) with existing modules;
 $f_{ij}^{accum} = f_{ij}^{accum} + f_j^+$ on every span (not i);

end for
 Update modular capacity allocations on spans:

$$\langle n_j^m \rangle = \langle m^* (w_j + f_{ij}^{accum}) \rangle \quad \forall j \in S$$

end for.

To solve mod_MIC_NF with iterative application of an SP algorithm (instead of using an LP solver) the procedure of [Section 6.12.3](#) would correspondingly change to:

Define: $\text{mod_cap}_j = \sum_{m \in M} n_j^m \cdot Z^m$; $\text{used_cap}_j = w_j + f_{ij}^{accum}$ for every span j

For every *unit* of demand on relation r .
 For every span j (excluding failure span i):
 If $\text{used_cap}_j < \text{mod_cap}_j$ then $\text{edge_cost}(j) := 0$
 Otherwise $\text{edge_cost}(j) := c_j^+(j, \text{used_cap}_j)$;
 Call SP algorithm (edge_costs);
 For every edge on route found: $\text{used_cap}_j = \text{used_cap}_j + 1$;
 end for;
 end for.

6.13.3 Modular Minimal Incremental Cost Multi-Commodity Network Flow

Recall that the difference between greedy heuristics 1 and 2 is that the latter solves directly with an optimal model for the multi-commodity flow pattern and related minimum incremental capacity addition for each restoration scenario. In the non-modular case this was based on replacing MIC_NF in path_greedy_1 with MIC_MCNF in path_greedy_2 and above, we looked at a modular capacity variant for the first. Let us now, therefore, consider the corresponding modular version of path_greedy_2.

The key change will be to solve for the minimum *modular* incremental cost multi-commodity flow solutions for each new failure scenario. We can designate this mod_MIC_MCNF(i). Here, the aim will be to accept any modular capacity placement decisions made by prior iterations for other failure scenarios and to address the added capacity needs for restorability of the current scenario in a way that minimizes total modular cost of *the complete design* up to and including the current failure scenario. Thus, there is a new aspect in the problem in that each iteration will be able to *revise* the accumulated capacity design as an ever more global view builds up. It will never be able to take away spare capacity channel totals needed for prior scenarios, but it will be able to continually revise the decisions about what modular stacking to use to meet all prior and current incremental requirements at minimum modular cost. In addition, we will have no need of pseudo-cost techniques here because we will be solving directly for the complete routing and minimum capacity solution using absolute total costs for whatever modules are employed and our decision variables are directly in terms of module decisions. (Pseudo-cost approaches are only needed when demands are routed individually in a successive greedy way within an environment that is later modularized on the resulting totals.) This model will be capable of global optimization of that aspect within each iteration. To take advantage of the context that Phase 1 iterative construction offers to continually revise and improve the span-wise modularity decisions we need to add to our vocabulary:

- $n_j^{0,m}$ = Number of modules of the m^{th} size already decided to exist on span j from the *prior* failure scenarios considered already (an input).

n_j^m = Revised number of modules of the m^{th} size on span j (a decision variable).

The model is:

Mod_MIC_MCNF(i)

Minimize

Equation 6.102

$$\sum_{m \in M} \sum_{j \in S} c_j^m \cdot n_j^m$$

subject to:

1. Multi-commodity source, sink, and transshipment:

Equation 6.103

$$\sum_{j @ n} f_j^r = \begin{cases} d^r, & \text{if } n = O(r) \\ -d^r, & \text{if } n = D(r) \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N \quad \forall r \in D_i$$

2. Sufficiency of (revised) modular capacity for *current* failure requirements:

Equation 6.104

$$\sum_{j \in D_i} f_j^r + w_j \leq \sum_{m \in M} n_j^m \cdot Z^m \quad \forall i \in S$$

3. Sufficiency of (revised) modular capacity for *prior* failure scenarios:

Equation 6.105

$$\sum_{m \in M} n_j^{0,m} \cdot Z^m \leq \sum_{m \in M} n_j^m \cdot Z^m \quad \forall i \in S$$

4. Zero flow on failed span (or all spans of failure scenario):

Equation 6.106

$$f_i^r = 0 \quad \forall r \in D_i$$

Equation 6.107

$$f_j^r \geq 0, n_j^m \geq 0 \text{ integer}$$

Using Mod_MIC_MCNF, the overall Phase 1 heuristic becomes:

mod_path_rest_greedy_2

Enter with an initial set of w_j and modules n_j^{om} (may be null).
For every failure scenario i ;

Solve mod_MIC_MCNF(i) with existing modules n_j^{om} ;

Update modular capacity $n_j^{om} := n_j^m$ for all (j not= i)
end for.

This model is the "least greedy" of the Phase 1 heuristics presented, from the following standpoints: First, it solves optimally for the multi-commodity routing and capacity increment within each failure scenario, reusing any already-placed modular capacity. In this regard by itself it can be thought of as a modular counterpart to path_rest_greedy_2. But the extra feature of the model is that, under [Equation 6.105](#), each iteration is also enabled to see some of the short-sightedness of prior iterations in their module decisions and can revise the modular construction of capacity on spans to reduce the objective function under economy of scale effects. What this can do is have an effect such as superseding an accumulation of two 40-channel modules placed early in the sequence of failure scenarios, with a single 160 channel module later near completion of the design when a third 40-channel module might otherwise have been added. Near the end of the iteration on failure scenarios an essentially global revision of final modular capacity costs occurs, but this is not the same as simply post-modularizing the solution to minimum cost modular stackings because at each prior iteration, the restoration routings that were decided upon were already taking into account, the prior modular capacity placements.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.14 Phase 2 Forcer-Oriented Design Improvement Heuristic

The concept of forcers and forcer analysis was initially developed in [Chapter 5](#) in the context of span-restorable networks. The basic idea can be applied to path-restorable networks (or SBPP networks) as well. This section introduces the basic forcer-related idea for design refinement in a path-restorable network context.

We have already seen in various design models that the spare capacity is always generated by a series of inequality constraints which relate non-simultaneous restoration flows on each span to the spare capacity of the span. In the context of path restoration these take the general form as follows when all the failures scenarios are listed in detail for one span x in terms of how they impose restoration flows on that span:

failure scenario a --> affected demands r1, r2, r3:

flow for r1 + flow for r2 + flow for r3 \leq spare on
on span x on span x on span x span x
.... (statements for other spans)....

failure scenario b --> affected demands r4, r2, r6, r7:

flow for r4 + flow for r2 + \leq spare on
on span x on span x span x

.... (statements for other spans)....

and so on.

The general point is that over all the failure scenarios (a,b,c..etc) (which are non-simultaneous), it may be that only one of the sums of simultaneous restoration flows imposed on span x reaches equality in the solution. In other words, there may be only one failure scenario, say scenario k , that is a binding constraint with respect to spare capacity on span x in the solution. This is the one scenario under which the spare capacity on span x is fully used. All other failure scenarios are restored with a lower requirement for spare capacity on span x than the one particular scenario requires. We would thus say that failure scenario k is the *forcer* of the spare capacity on span x in the path-restorable context, just as we would for span-restorable forcer analysis. Having said this, we must recognize in general that two or more scenarios could be equally binding on span x , but for simplicity in what follows we will assume a unique forcing failure scenario for each span's spare capacity. (The generalization to multiple co-forcers then follows easily.) Finally, although we are making these points with reference to the structure of an ILP model for path restoration, the same general structure of forcing failure scenarios will underlie any design generated by a Phase 1 heuristic.

The link to a strategy of design improvement is that forcer relationships highlight the key failure scenarios. Especially in cases where the order dependency has resulted in an inefficient design, the forcing effect of one scenario on a span may be far above the next highest requirement on the same span. The strategy is then to identify the strong forcers and somehow reduce the forcing pressure of scenario k on span x , especially where the spare capacity on span x due to scenario k is seen to be far in excess of that needed by any other scenarios. There are two general ways in which this may be done:

1. Seeking changes to restoration routes for scenario k so as to use less spare capacity on span x (but not using more elsewhere than is saved on span x).
2. Changing working routes to alter the set of demands affected in scenario k and hence alter the forcing magnitude of scenario k on span x .

In any case where the spare capacity has been solved optimally, net improvements through the first avenue cannot be found, by definition. In cases where the design has been constructed by a heuristic however, it is quite likely to find opportunities of the first type. Improvements of the second type will also be available if the design is optimal but non-joint or if it is a heuristic of any type because the initial routing of working demands was done independently without any consideration of effects of restoration capacity.

A first step to improving the design efficiency by forcer-related strategies is to conduct a forcer analysis of the initial design. In a path restoration context the basic information obtained in a forcer analysis is the answer to the question: "For each span, how much of its spare capacity is used under each failure scenario?" If the design is developed with the iterative Phase 1 heuristics above, it may be tempting to think of recording the total restoration flow across each span for each failure scenario as the design is developed, but this would not be the desired analysis of the completed design's forcer structure. A separate round of all-failure scenario simulation is needed on the final iteratively constructed design from Phase 1 to see the global opportunities for capacity reduction in the final design. Given any starting design, the forcer analysis procedure is:

Forcer_analysis

For every failure scenario i ;
 Identify all failed paths for each O-D pair affected;
 Solve PRR-1(arc-path) or PRR-1a (arc-flow);
 (Alternately: simulate the actual restoration process)
 Record:
 Spare capacity used on each span j (not $j = i$).
 Restorability level of the failure scenario.
end for.

This amounts to running a trial of the corresponding restoration mechanism and observing the results. PRR-1 would be the appropriate model for the restoration process if centralized on-line computation or preplans were used with specifically defined "eligible restoration routes" that reflect certain hop limits and/or optical path considerations such as amplifier noise and regeneration requirements. PRR-1a is a pure transportation like flow solution without explicit control of the implied restoration paths, but which may solve faster and be easier to set up for solution. Alternately, the forcer analysis stage can also be a convenient time to check for compatibility of the capacity design with any actual distributed restoration or preplanning protocol to be employed in the real network. The output of the forcer analysis is the data for a matrix of spare capacity usage by each failure scenario on each span. It is also convenient while executing the forcer analysis loop, to validate the restorability level of each failure scenario, thus flagging anywhere that the design intent is perhaps not met by the specific restoration mechanism within the given capacity distribution.

Denoting $s(i,j)$ as the amount of spare capacity used (i.e., the total restoration flow) on span j under failure scenario i from the forcer analysis, we want to extract the following further information to seek design improvements:

- $s_j^1 = \max_i(s(i,j))$ is the maximum amount of spare capacity used on span j by any failure scenario.
- $A_j^1 = i | (s(i,j))$ is maximal over all i . This means that it is failure scenario i that forces the most simultaneous restoration flow over span j . If more than one scenario forces exactly the same maximal flow over span j we temporarily ignore any such spans other than the first-found instance.
- $s_j^2 = \max_i(s(i,j))$ excluding $i = A_j^1$. This is the *second largest* amount of restoration flow on span j over all failure scenarios. It may be that $s_j^1 - s_j^2$ in which we have at least two equal forcers of span j .
- $A_j^2 = i | (s(i,j))$ is maximal over all i excluding $i = A_j^1$. This is the failure scenario i that is the *second-largest* forcer of span j .

To summarize, for each span j we wind up finding A_j^1 , the failure scenario that uses the most spare capacity on span j , and the amount of such capacity, and we also find the "second most forcing" scenario for each span A_j^2 and its amount of spare capacity usage. The

$s_j^1 - s_j^2$ difference is referred to as the *forcing strength* of failure scenario A_j^1 on span j . With these values extracted from the forcer analysis procedure, we first check for any trivial opportunities for spare capacity removal:

Simple_excess_removal

For every span j :

If $s_j^1 < s_j$ then: $s_j = s_j^1$

end for.

In the non-modular designs any span where $s_j^1 < s_j$ supports a direct removal of capacity units from the design. This is equivalent to the prior procedure of "add0_sub1" design tightening for span restoration [GrBi91]. In a modular design the discovery of such simple excess spare channel counts does not directly suggest a whole module removal. Rather it may suggest a reduction in the number of individual logical spare capacity channels that need to be equipped within the module, and it may also trigger a module resizing. That is, if s_j drops, then m^* ($w_j + s_j$) may possibly also be revised downwards.

Following any such simple removals, the forcer data is inspected in a more targeted way to seek reassignments of restoration flow that may be feasible in the initial capacity design. The basic idea is to find the most dominant forcer relationships, and try a new solution for the routing of restoration flows so that the forcing magnitude of this failure on span j is relieved. The greatest amount of restoration flow on span j will be relieved by seeking reroutings under the current failure scenario, but the net removal of spare capacity on span j cannot reduce it below that required by the second largest forcing scenario:

Adjust_rest_paths

Rank all spans j in decreasing order of $(s_j^1 - s_j^2)$ (and $>$ a threshold).

Repeat

For each j and corresponding forcing failure $i = A_j^1$:

1. List all restoration paths used for scenario i that cross span j ; i.e.

$$P(i, j) \equiv \left(\bigcup_{r \in D_r} p \in P_i^r \mid \delta_{i,j}^{r,p} = 1 \cap f_i^{i,p} > 0 \right)$$

2. Temporarily reduce the spare capacity on span j to s_j^2 .

3. Solve PRR-2a for all demands in $P(i, j)$.

4. Temporarily change the restoration flow assignments for O-D pairs whose rerouting was feasible under PRR-2a.

5. For all restoration requirements in $P(i, j)$ not satisfied solve MIC_MCNF(i , unsatisfied demands)

6. If spare capacity added at step.6 exceeds $s_j^1 - s_j^2$ abandon

all changes. Otherwise make all changes permanent including spare capacity added at step 5. Until all spans with forced magnitudes above threshold are done.

Recall that PRR-2 and PRR-2a (Section 6.2) are formulations for path restoration rerouting that seeks to maximize the total restoration flow under general conditions where 100% restorability of all affected flows may *not* be possible. (PRR-2a is the version that gives us diagnostic information about which specific relations are not restorable.) The above procedure thus leaves in place all restoration flows for scenario i that do not cross span j . For those that do, however, it drops the spare capacity on span j to the next lowest known

requirement of other scenarios (i.e. s_j^2), and then tests to see how much of the flow crossing span j could be supported by other routes through the existing spare capacity. All restoration requirements that previously relied on span j but were feasible under PRR-2a without span j are reassigned to those new restoration routes permanently for scenario i . Those that were not are then allowed to rely on span j again, as much as any other span, under a final MIC_MCMF treatment for any demands that cannot be rerouted with span j at its

reduced level of s_j^2 . If the final MIC_MCMF solution adds more spare capacity than has so far been saved on span j ($s_j^1 - s_j^2$) then the forcer reduction attempt is abandoned. In general this will produce some intra scenario reduction of spare capacity requirements on span j , but we do not allow or attempt a reduction to go below the requirements of the second-largest forcing failure scenario on span

s_j^2 otherwise we begin to interfere with design considerations *ir other* scenarios and would have to revisit all other scenarios. The reason for the ranking by largest $(s_j^1 - s_j^2)$ difference is now apparent: we want to focus on those forcing relationships that could most be reduced by intra-scenario considerations before cross-scenario considerations would come into play to limit the possible

improvement. The threshold on $(s_j^1 - s_j^2)$ difference is just to avoid considering cases where the potential benefit is too small to be worth including in the analysis time.

The above procedure can be repeated with an updated forcer analysis after each round of restoration path adjustments, until no further benefit is found. At that stage the initial heuristic design has been evolved to use less spare capacity by arranging for restoration flow assignments to routes for each failure scenario that more nearly use the same amount of spare on spans across multiple scenarios. The resulting design can be said to have been "forcer-leveled."

The remaining domain of search for improvements is to make changes in the working path routes in a way that reduces the stronger forcing effects. This is a fairly accessible technique in the case of span restoration; a strongly forcing w_i quantity can be reduced by rerouting some of the working flow crossing span i over other non forcing spans. For path restoration, however, it is not clear how a suitably simple direct adjustment heuristic could work (i.e., simple enough to hold an advantage over directly approaching the joint design problem). The central problem is that whereas *adjust_rest_paths* can operate within the conditions of one failure scenario at a time, any change to working path flows affects at least two failure scenarios. And moves of entire working paths to nearly equal length alternatives can affect many scenarios at once. Also, if we aim to change the working flow assignments to reduce the forcing strength of a certain scenario on a certain span, we need subsidiary heuristic principles to tell us which individual working flows in the scenarios should be reassigned without aggravating other forcer effects. It seems that improvements of this type may therefore be best sought through some type of random search. We thus defer further discussion of working path forcer-related improvement searches until the later discussion of a "simulated allocation" strategy for working routes.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.15 A Tabu Search Heuristic for Design Tightening

The basic structure and concepts of tabu search were introduced in [Section 4.18.3](#). The central idea is of conducting a local steepest descent or ascent type of search, but to use memory of the search history to avoid revisiting the same local minima states. In devising a tabu search procedure a key decision is the operation or process that defines a "move." The set of all design states reached by applying a single move to the current state is called the neighborhood.

If our aim is to minimize the spare capacity only (i.e., without changing the working routes), the current network state can be specified by the spare capacities \vec{s} . The most obvious move in this context (as *irdesign tightening* for span restoration [GrBi91](#)) is to apply a reduction of spare capacity to one span, test for restoration feasibility, and repair the design if needed. Thus each state \vec{s} has $|S|$ neighboring states which are reached by applying a predefined policy of capacity removal. The removal could consist of:

- A single spare channel.
- A fixed percentage, say *min* (1 unit or 5%) of the spare capacity of the span.
- Removal of a module of capacity (which could affect working capacity and needs separate considerations below).
- Conversion of a module to the next-lower capacity or next-lower cost configuration.

Of course, applying the removal of capacity may cause a state in which not all failure scenarios are fully restorable. Thus, the cost of each neighbor state has to be generated by repairing the restorability of the design as needed on each move (and if the move is ultimately taken, the new state vector \vec{s} is that of the repaired network following the step). To force search diversification in the usual tabu search way, each span that has been changed can be kept on a tabu list for a prescribed number of further iterations. When progress slows, large scale search diversification (or "restarts") can be based on taking a new initial capacity design solution. The tabu search heuristic can also make convenient use of basic subproblems and tactics already discussed above.

Path_rest_tabu_search

1. Obtain an initial feasible but sub-optimal solution \vec{s}_1 from *path_rest_greedy_1* or *path_rest_greedy_2*.

2. Initialize $n:=0$, *tabu_list*:= null

Repeat

$n:= n + 1$ (basic iteration)

3. For every span j :

4. $s(j):= \text{Remove_policy}(s(j), \text{modularity, etc});$

(other spare capacities are unchanged from current \vec{s}_1)

5. For every failure scenario i :

6. Find the set of restoration paths that cross span j .
(e.g. $P(i,j)$ from above)

7. Solve PRR-2 for all demand requirements of O-D pairs associated with $P(i,j)$ in the modified spare capacity

8. Case of (I from PRR-2a):

$l = 1$, no design repair is needed:

\rightarrow
cost(n,j):= cost(\overline{s}_1) less removal on j

$l < 1$, not fully restorable:

options:

Iterate MIC_NF (or mod_MIC_NF) to add
minimum incremental cost that restores
feasibility for each O-D pair with $l_r < 1$,

or

Run single instance of MIC_MCF
(or mod_MIC_MCF) for min cost addition
that returns all O-D pairs with $l_r < 1$ to
full restorability.

\rightarrow
cost(n,j):= cost(\overline{s}_1) less removal on j plus
cost of MIC_xx repair procedure.

end for (on failure scenario testing / repair)

end for (on basic capacity move on all spans)

\rightarrow
9. Change \overline{s}_1 to adopt the most cost-improving span removal move, i.e., min cost(n,j) unless j

is in current tabu_list.

10. (Aspiration criterion:) If the design cost with move(j) is the best ever seen, the

take move(j) even if it is currently tabu.

11. Update tabu_list.

Until (rate of improvement below threshold) then:

12. If total_CPU_time still acceptable, restart (at 1) with a different initial feasible

design and change sequence of failure scenarios in initial *path_rest_greedy_design*

generation.

The *remove_policy()* implements whichever of the possible capacity reduction schemes is intended on the named span. The removal procedure will include consideration that working capacity is not subject to removals and spare capacity cannot become negative. As described, the effect of this removal on restorability is first detected by an instance of PRR-2 (or 2a). The idea is that if PRR-2 produces $l = 1$ then restoration under the given failure scenario remains feasible even with the removal. If $l < 1$ in PRR-2 it means that not all affected O-D pairs can be restored in the presence of the removal, indicating the need for a minimum-cost incremental capacity addition to repair the design for those demand pairs r having $l_r < 1$. Undoing the removal would repair the design but the intent here is to see if augmentation(s) elsewhere in general could restore feasibility at less cost than the removal from span j represents. For clarity of the logic in *path_rest_tabu_search* as stated above, we have presented an explicit PRR-2 step followed by an appropriate MIC_xx step if needed. (By MIC_xx we just mean to recognize that either MIC-NF or MIC_MCNF could be used as outlined above). It should be noted, however, that the MIC_xx problem instance could also be used directly in place of both steps 7 and 8 because if no repair is needed, any of the MIC_xx formulations will simply return an incremental cost result of zero. Thus in practice the choice between explicit PRR-2 tests and selective MIC_xx repair problems depends on how often the test-case removal of spare capacity is without effect on restoration feasibility and the relative solution times of PRR-2 and MIC_xx problems.

Step 11 is where any of the more generalized techniques and strategies of tabu search "recency" and "frequency" memories may be implemented. Recency in this context would mean that each span that received a removal would be tabu after that for a certain time. Frequency notions could prohibit a certain span from being revisited too many times in the elapsed period even when not tabu. Many problem specific tabu strategies can also be applied. For instance a span may possibly be made tabu for increasing lengths of time if it is frequently found that when it is tested for a removal the repair procedure costs more than the removal saves. There may be many spans in this category and more so as the design improves its overall efficiency. Thus a policy of long tabu after three tests which all increased cost, and perhaps permanent tabu status after another such outcome, could accelerate the overall progress by narrowing the search down to spans where removals are more promising. The reason that *all* spans are considered at step 3 (not just those that are not in the current tabu list), is that this is required to support implementation of the *aspiration criterion* at step 10, so as not to exclude tabu list members from evaluation. Only by evaluating all moves, even if tabu, can we detect if the move would produce the best design yet seen in any iteration, then it can be taken regardless of tabu status.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

6.16 Simulated Allocation Type of Algorithm for Design Tightening

Finally, we will look at a heuristic technique that can explore for improvements in the overall design of a path-restorable network through changes in its working capacity routing. This is based on a simulated annealing-like approach inspired by the work of Piore [Piore97] on *Simulated Allocation* and by Tinkler's approach of *Randomized Route Addition and Deletion* [CiHe00]. To this basic style of searching the routing dimension of the solution space, we add forcer-based considerations to prioritize and structure the search for improvements in working path routing and hence overall capacity reductions. The basic procedure is to generate and test different allocations of working flow to routes between each O-D pair and recompute the spare capacity (selectively, if possible, for efficiency). Improvements will always be kept but in addition there will be a time-decreasing probability that a change that increases cost will also be kept (to escape local minima).

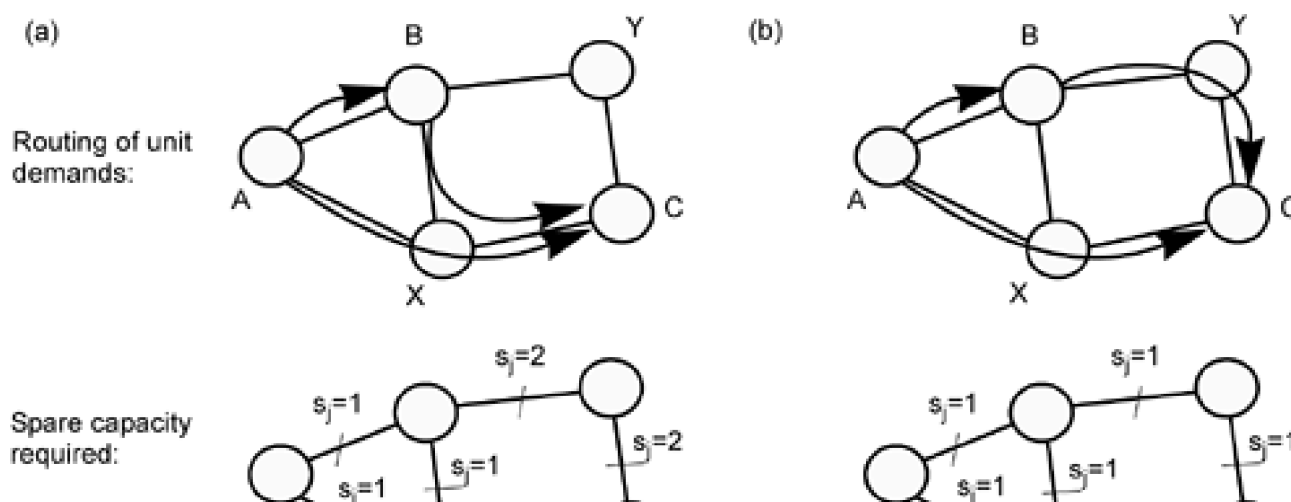
As we will use it in the path restoration problem, the process can be thought of as experimenting with moving certain restoration flow requirements for O-D pairs from one failure scenario to another. The possible interchanges between failure scenarios are highly structured by the network topology so the sets of failed demand pairs grouped under each failure scenario cannot be arbitrarily posed. They can only be explored by testing changes in the routing of working demands over the specific topology. But as we know from forcer analysis, not all failure scenarios are equally demanding in terms of the spare capacity they require in the design. The ranking of failure scenarios by their forcing strength on individual spans, and the network as a whole, can therefore help us prioritize the search effort. The greatest potential gain is expected by seeking changes in working route assignments that effectively reduce the most strongly forcing failure scenarios first.

A simple example of how certain reallocations of working routes can lead to a reduction in spare capacity is given in Figure 6-19. In Figure 6-19(a) demand units for three O-D pairs each follow their shortest path and the arrangement of spare capacity for their survivability is shown below. The result is that failure scenario (X-C) requires two units of spare capacity on spans (B-Y) and (Y-C). No other single-failure

scenario in the example requires more than one unit of spare capacity on these spans. Thus the forcing strength $(s_j^1 - s_j^2)$ of failure scenario (X-C) on spans (B-Y) and (Y-C) is one. If, on the other hand, the demand for O-D pair (B-C) were routed as in Figure 6-19(b), the change in working capacity may be insignificant (or nil) but we can see that a unit capacity savings arises in the spare capacity required. This occurs because the change in routing moves restoration of the (B-C) demand into a different set of failure scenarios, with the result of forcer-leveling across the set of all failure scenarios involved. The revised forcing strength of failure scenario (X-C) on spans (B-Y) and

(Y-C) is still one (because $s_j^2 = 0$ now). But the reduction in spare capacity occurs because the rearrangement from (a) to (b) has targeted the strongest forcing failure scenario in (a) with the result that scenarios (B-X) and (C-X) become equal co-forcers of spans (B-Y) and (Y-C) in the revised design. In other words, the forcer structure has been leveled by the rearrangement in working path assignments. The more general manifestation of these effects also involves varying total demand bundles on each O-D pair and splitting of the bundles over several working routes to discover forcer-leveling rearrangements.

Figure 6-19. How working route changes can reduce spare capacity cost for path restoration.





total capacity ($w_i + s_i$) = 12



total capacity ($w_i + s_i$) = 11

Two further considerations before summarizing an overall heuristic are:

- From studies of jointly optimized designs, we know that even in optimal joint designs the routes taken by working paths still deviate relatively little from shortest paths.^[3] This will allow us to predefine a limited set of routing changes that it makes sense to consider and exclude a vastly larger number of longer routes for working path rearrangements.

^[3] One exception to this is under strong modular-capacity economy of scale effects, which we are not considering in the above discussion.

- Each change to a working route will affect at least two failure scenarios: at least one other span (but possibly several) will lose the corresponding path from its failure scenario and at least one other failure scenario will now include the path. Computationally it would be desirable not to have to resolve the entire spare capacity design problem to evaluate the effect of each working route change. A classification of spans based on forcer relationships can guide the decisions about when and which failure scenarios need to have their spare capacity design revisited.

We can now state the overall heuristic, followed by more detailed discussion of key steps.

Forcer_leveling_simulated_allocation

[\[View full width\]](#)

1. The problem is entered with an initial restorable network design that may be either
 - ➡ from a greedy algorithm or a non-joint optimal design solution.
2. Preliminary: Find all distinct routes between each O-D pair with lengths that are
 - ➡ within (say) 10% more than the corresponding shortest path distance, or, if distance is
 - ➡ based on hop counts, all distinct routes with no more that (say) two hops more than the
 - ➡ shortest path.
3. Perform a forcer analysis and classify all spans as non-forcers, unique forcers, or
 - ➡ co-forcers, with respect to each other span.
4. Rank all failure scenarios by decreasing maximum forcer strengths.
5. For each failure scenario i (by decreasing forcing strength):
 6. Set $t = 0$, temperature $T(0) := T_{max}$, and cooling schedule $T(t)$;
 7. While $T > T_{min}$ do
 8. For $n = 1$ to N_C do
 9. Randomly choose an O-D pair affected under scenario i ;
 10. Assign the working O-D flow to a different route;
 11. Update the working capacity cost;
 12. Update spare capacity design (if needed-see below);
 13. $new_cost - old_cost = DC$;
 14. $P_{accept}(t, \Delta C) = e^{-\Delta C / kT(t)}$
 15. If $(DC < 0)$ or $(rand() < P_{accept}(T(t), DC))$ then:
 - keep the routing change and revised capacity;
 - (optionally: update forcer classifications);
 - otherwise, reject the change;

```

end for;
16.  $t := t + 1$  (or  $T := a T$ )
end while.

```

The above gives a basic framework for the simulated annealing-like process. For each failure scenario it provides N_C random routing changes to be tested at each temperature step. The probability of random acceptance of changes that increase cost goes down both with the time/ temperature schedule on the inner loop and also with the magnitude of the cost increase.

6.16.1 Efficiently Updating the Spare Capacity Design

At step 12 above, a base-line option is to completely repeat the spare capacity design for each change in a working path route. A more efficient approach to updating the spare capacity design is, however, to revisit the spare capacity design selectively with respect to only those failure scenarios that are altered by the move of a single working path. In this regard we note that a shift in a working path can affect failure scenarios in three ways:

1. **Spans from which the path was removed:** The failure scenarios associated with these spans can only cause changes in the direction of requiring *less* spare capacity elsewhere in the network. In particular, if a forcer span is relieved, we can directly remove a unit of spare capacity from each span that is *uniquely* forced by that span without reevaluating the restorability. In no case, however, there is no need to reinspect the restorability of the spans that had a *working* path removal.
2. **Spans to which the new paths were moved:** The failure scenarios associated with these spans can only cause changes in the direction of requiring *more* (or the same) spare capacity elsewhere in the network. We know that a spare capacity increase will only be needed, however, if we add working capacity to an existing forcer span. Our move generating process can try to avoid increasing any forcers, or simply make path moves at random. In either case, where the path shift adds a unit working capacity to a non-forcer span, there is no need to reevaluate restorability for those spans or add any spare capacity elsewhere to the design. Where forcers are incremented, we must reevaluate the spare capacity design to support their restorability, however.
3. **Spans not involved in the original or new routes:** The restorability of the failure scenarios involving spans that were not on the prior or new working path can only be affected by reductions in spare capacity on other spans for which they are a forcer. Thus, there is no cause to revisit any failure scenario that is a global non-forcer. In addition, where reductions in spare capacity are made under 1 above, on uniquely-forced spans, there is also no impact on other spans. If the shift in path is away from a span that is a co-forcer, however, the spare capacity design for all co-forcer failure scenarios must be reevaluated to determine if a net spare capacity reduction is possible.

These general considerations guide the determination of which failure scenarios to revisit to reassess restorability following a working path shift. We will see that they also can directly suggest certain rerouting improvements. Based on these considerations, we want to consider how to classify failure scenarios by virtue of their forcer status.

6.16.2 Classifying Spans by Forcer Status

If we consider the set of failure scenarios corresponding to each individual span failure, then each span can be given a certain classification that is relevant to determining when the spare capacity design has to be reassessed to evaluate the benefit of a given routing change. Recall the basic information produced by forcer analysis: $s(i,j)$ is the amount of spare capacity used on span j under failure scenario i . $s(i,j)$ can be thought of as a table where each row lists the spare capacity usage of each failure scenario on a particular span and each column lists the spare capacity usage of each failure scenario on all spans under that scenario. Thus, a span i is a:

1. **Non-forcer** if $s(i,j) < s_j \quad \forall j \in S(j \neq i)$. In other words, when failure i occurs, it is restored without fully using all of the spare capacity on any other span. The smallest difference between actual needs and the spare capacity present on each span, i.e. $\min_j (s_j - s(i,j)) \quad \forall j \in S(j \neq i)$, is called the *non-forcer depth* of the failure. The significance is that if a span is a non-forcer of depth u , we can route up to u more units of working capacity over that span *without having to reconsider the spare capacity design*.

2. **Unique forcer** of any other spans j for which $[s(i, j) = s_j] \cap [s(x, j) < s_j \quad \forall x \in S | j \neq i]$. In other words, span i is a unique forcer of j if it is a forcer of span j and there are no co-forcers of j , i.e., no other failure span fully uses the spare capacity on span j . The *unique forcing strength* (of span i on span j) is defined relative to the next largest use of spare on span j

by other failure scenario, i.e., $s_j^1 - s_j^2$. The significance is that if a span i is a unique forcer (of strength u) of span j and we remove working paths that cross span j , then, for each removal on span i up to the unique forcing strength we can remove a matching unit of spare capacity on span j *without having to reconsider the spare capacity design*.

3. **Co-forcer** of any spans j for which $[s(i, j) = s_j] \cap [\exists x | s(x, j) = s_j \quad x \neq i]$. In other words, where the failure of span i fully uses the spare capacity on span j , but so do other failure scenarios x , then i and x are *co-forcers* of j . Note that a span i can be a unique forcer of one span k , while being a co-forcer on another span j . The practical relevance of co-forcer status is that if we change the number of working paths crossing any co-forcer, we *do* need to recompute the spare capacity requirements over all failure scenarios involved.

6.16.3 Path-Shift Strategies for Direct Forcer-Based Improvements

In the basic Simulated Allocation framework above, one allows that path shifts may be generated randomly over the set of eligible working paths for each relation. Following each proposed path shift, the forcer-based considerations above can then indicate which spare capacity can be removed, and which failure scenarios have to be revisited, possibly reinstating all or more of the removed spare capacity at other spans. In this way the net benefit of a random move is assessed. But forcer considerations also directly suggest certain path move opportunities that we should always check for first as these are changes in working route assignments that would be guaranteed to yield a net reduction in spare capacity, and they can be quickly checked to see if such a reduction exceeds the possible increase in working path length associated with the working route change. Specifically:

- **Criterion 1:** If any of the eligible working routes for an O-D pair is comprised entirely of non-forcer spans (and the original route is not):
 - a. Working paths should be reassigned to that route until the non-forcer depth of at least one span on the route reaches zero. The failure scenario is then updated as being a co-forcer of the span(s) that reached zero non-forcer depth.
 - b. For each unit-capacity path moved off of the initial route, spare capacity on other spans can be directly relieved as follows:
 - for each *non-forcer*: no reductions
 - for each *unique forcer* i on the original path: one unit spare reduction from each other span j that is uniquely forced by i . This may be repeated for additional unit path removals until the unique forcing strength of span i on each other span is depleted. At that point the status of span i relative to span j is updated to co-forcer.
 - for each *co-forcer*: no spare capacity reduction can be directly obtained.

If the total reductions in spare capacity exceed any increase in working capacity due to the path shift itself, then the routing change can be adopted. If a route comprised exclusively of non-forcers does not exist a somewhat more general but improvement criterion is:

- **Criterion 2:** If any of the eligible working routes for an O-D pair has fewer unique forciers plus co-forcers en route than the original route has unique forciers. In this case, reassign working paths to the route until the criterion no longer holds as non-forcer spans are built up to reach co-forcer status on the new route. The forcer structure is then updated indicating co-forcer status of the span(s) that reached zero non-forcer depth and the spare capacity removals associated with reducing the unique forciers on the original path are claimed.

Again, the net benefit is weighed against possible increases in the working path due to the proposed shift in routing. Note that a truth condition in Criterion 2 can arise for an O-D pair following a number of initial path moves on the same O-D pair under Criterion 1. Thus we have an initial improvement heuristic that can be applied explicitly before entering the random simulated allocation search phase for further

improvements. A procedure for search for such direct improvement opportunities is summarized as follows:

Direct_forcer_improvements

1. Conduct initial forcer analysis and forcer classifications
 2. For every demand pair r :
 - Repeat (for a unit path at a time):
 - Test for Criterion 1 among eligible paths;
 - Exploit any improvements, update spare capacity, routes, and forcer table;
 - Until Criterion 1 not true;
 - Repeat (for a unit path at a time):
 - Test for Criterion 2 among eligible paths;
 - Exploit and update;
 - Until Criterion 2 not true;
- end for.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

Chapter 7. Oversubscription-Based Design of Shared Backup Path Protection for MPLS or ATM

Both ATM and MPLS are instances of a combined circuit and packet-switching strategy. Almost as in co-evolution by different organisms toward the same solution in nature, the ATM and MPLS communities have independently found that a circuit-switching paradigm is superior for structuring and management of transport functions, including protection, while random asynchronous cell or packet flows are more flexible at the application level and allow the bandwidth efficiency of statistical multiplexing within the circuit-like transport connections. It is even more interesting that the ATM community began from a circuit-switching background and MPLS arose from the packet-switching community. From opposite starting points they both reached essentially the same logical combination of statistical multiplexing of payloads over circuit-like paths for transport. The central idea in this chapter is that there is yet another desirable dimension to this combination. That is the opportunity to design for *controlled oversubscription* of capacity upon restoration. This is a concept and opportunity that applies only when stochastic flows are routed over circuit-switched logical pipes.

One of the risks of performing restoration at the MPLS or ATM VP layer is that the sudden rerouting of stochastic flows may cause congestion on other surviving links. In SONET OC-n or DWDM lightpath-level restoration there is no corresponding concern about congestion, only about whether enough discrete replacement paths can be formed to support the desired restoration level. One way to avoid congestion in MPLS or ATM VP layer restoration is to provide additional bandwidth allocations of at least 100% on links that will sustain the restoration flow. By 100% here we mean that when a link fails, the network planning has provided bandwidth allocations on other links that are at least equal to the bandwidth allocation of the failed working paths. In this way, cell or packet-level performance in any restored state should be no worse than that of the working paths before failure. In practice, an often reported reality of IP/MPLS-based networks is that they are simply kept in such an overprovisioned state that average utilizations of link capacity hardly exceed 15 or 20% in any case. Experience shows that this is empirically the simplest, albeit brute force, way to stabilize IP networks so that they "stay out of trouble" even when routing of flows are altered for restoration or congestion reasons. In the longer run, however, this is not the intended or desired way to run an IP network. It has so far been simply an expedient way of operating such networks until methods and understanding improve to enable their operation at much higher efficiency levels.

Our view in this chapter that one of the keys to this higher efficiency can be capacity planning for restoration that involves deliberate, but controlled, strategies of *oversubscription*. Under oversubscription-based capacity planning the stochastic nature of flows provides both an extra challenge and an opportunity. The extra challenge or risk is that, because flows are stochastic, we can in principle cause massive congestion events that are not at issue in the lower layers. However, the opportunity is that the stochastic nature of traffic in these layers can be deliberately exploited to reduce total cost in a way that is not available at the lower layers. Our aim is to propose the basic concept of oversubscription-based survivability design, provide some basic design methods, and give a quantitative indication of the potential advantage of methods for deliberate design for controlled oversubscription of capacity upon restoration.

7.1 Concept of Oversubscription

Oversubscription is notionally the extent to which the ratio of available capacity to nominal capacity allocations for stochastic flows may be stressed during a restored network state. It is related to, but not a synonym for "overload" in terms of packet level congestion. An oversubscribed state may or may not lead to overload—it depends on the utilization levels of the allocated capacities at the time of the oversubscription. Additionally overload is a traffic condition, while oversubscription is a comparative measure of actual capacity relative to pre-planned capacity allocations. Oversubscription-based capacity planning leads to assurances on the worst-case possible overloads, but it is a simpler and more practical planning framework than trying to plan capacity for protection while dealing directly with statistical traffic descriptions.

Consider a stochastic data flow for which a 1 GbE link, for example, is the nominal bandwidth allocation. The link may be said to be overloaded (based on predefined measures of delay and packet loss, and assumed traffic statistics) if its utilization reaches, say, 60%. Normally, however, the flow in the example would operate below congestion levels, say at 20% average utilization. Thus, packet-level considerations (of possibly great detail) would have already decided that for the nominal flow of 200 Mb/s, the "right-sized" capacity allocation is 1 Gb/s. Now if, during restoration, two of the same flows had to make due with sharing the single 1 Gb/s link, we would say that the capacity allocations are *oversubscribed* two-to-one. But if both flows were at nominal utilization or below, the combined utilization in the oversubscribed state is still only 40%, so there is no *overload* per se. Thus, oversubscription-based planning is about considering the how pre-failure bandwidth *allocations* relate to post-failure bandwidth *availability* in a restored state. By limiting the maximum oversubscription factor in capacity planning, we can strictly control the worst-case overload effects and we get the simplifying advantage of working entirely with capacities and capacity allocations, rather than a huge set of stochastic traffic descriptors and all possible mixes of assumed traffic. The latter problem—of deciding on the proper bandwidth requirements of various traffic types and mixes—still needs to be solved, but there is a large body of existing theory (or simulation methods) for doing so. The main advantage of the oversubscription-based planning approach is therefore that it lets the problem of bandwidth allocation to stochastic flows be decoupled from the problem of network-level planning of capacity to efficiently protect these nominal bandwidth allocations.

More specifically, oversubscription under a restored or rerouted state in general is the ratio of the total of the pre-failure bandwidth allocations made to flows that now cross the link, relative to the actual (or available) capacity of the link. The technical feasibility of providing restorability with an oversubscription factor larger than 1.0 is a unique property of a connection-oriented statistically multiplexed environment such as ATM or MPLS. Oversubscription of capacity (in the same sense) is clearly not an option in the DWDM or SONET layers because these layers require exact matching of discrete working signals with corresponding signal paths for restoration. Oversubscription in the WDM layer would in effect correspond to simply not having enough protection lightpaths to cover a set of failed working lightpaths. This would be a failure to provide the required level of restorability and all services on one or more affected lightpaths would experience abrupt and total outage. The only options are perfect protection for some or hard outage for others. There is no sense in which a general performance degradation due to a failure can be spread out—a bit of the pain being shared by all the users of the network. Even the multi-QoP strategies of [Section 5.9](#) do not escape this. Any one path—no matter what its service class—is still ultimately restored entirely, or not at all, under a given failure.

In an ATM or MPLS context there are ways to "share the pain" more generally over the network as a whole. It is technically meaningful to consider that the stochastic flows from more than one LSP (or VPs) may merge under protection rerouting to produce a combined load on a given link that somewhat exceeds the *nominal* utilization of that link—but still produces acceptable delay and cell loss probabilities in a restored-network state. In other words, we may plan to allow certain amounts of oversubscription of planned capacity during a failure. This offers a way to engineer for a softer degradation in performance overall, so that no path is faced with a complete outage, or, where we can use the "softness" of degradation to consider yet further schemes of multi-QoP service offerings. The challenge, however, is to manage the worst-case extent of the traffic-level degradation in performance and to exploit this to offer a continuous trade-off, under the network operators control, between capacity investment and the worst case performance impact of failure. The same approach will also require less total protection capacity than a corresponding SONET or WDM-layer protection scheme for 100% restorability.

In the MPLS layer we approach protection by *rerouting payloads between the containers*, whereas the lower layers *protects the containers* themselves. Both basic strategies offer different advantages and assurances to the network operator and their customers. The ongoing advantage of optical layer survivability is less complex capacity design and the relative simplicity, speed and certainty of the rerouting mechanisms in that layer. The optical layer always sees fewer objects to manipulate and they are generally all of the same discrete dimensions, making the optical layer a fundamentally less complex and less data dependent domain in which to provide survivability. But protecting the containers is less efficient than protecting the payload flows directly, especially at times when utilization is

low. The two types of protection or restoration are not incompatible together, however. It is reasonable to expect that both layers will host their respective protection mechanisms in future networks. One set of lightpaths can be marked for optical layer protection. A different set of lightpaths may be marked "do not protect" in the same optical layer. These lightpaths bear the logical links of an ATM or MPLS layer which embodies its own protection strategy. It is in the latter network layer that the opportunity for oversubscription-based restoration planning is an attractive option. Therefore, the following design approach is not positioned as a competitor against STM or optical layer survivability schemes. Rather it adds to the repertoire of options and techniques available for many different network design contexts and priorities.

Of course, ultimately one wants to control QoS, not simply the oversubscription ratio. In the approach here, the linkage to cell-level performance guarantees is through a separate set of considerations to determine the maximum tolerable oversubscription ratio, denoted X_{tol} , with which to undertake the restoration capacity optimization. The choice of the limiting value for the oversubscription design strategy could vary considerably with operator assumptions and policies, such as whether to assume worst-case or average-case LSP / VP utilization and traffic statistic for the prospective failure event times. A price-competitive provider might take the view that all VPs may suffer slightly during a restoration event, accepting a high X_{tol} . A more conservative operator may choose $X_{tol} = 1$ which is the STM-equivalent case of restoration bandwidth equivalent to failed VP bandwidth, with no oversubscription. The assessment of tolerable QoS risk may also consider that the actual QoS impact incurred depends on the time of the failure relative to the peak periods, and to the equipment-provisioning interval. It would be pessimistic to presume that all failures occur right in the busy hour for the services affected, and at the growth build-out of the equipment. These kind of considerations (supplemented with results from quantitative studies, such as follow) would go into determining some reasonable $X_{tol} > 1$ for use in the spare capacity design. At this stage we do not presume a value of X_{tol} , but explore the savings possible in network design as X_{tol} increases.

7.1.1 Historical Background and Some Misconceptions

The basic idea for *controlled* oversubscription of capacity allocation emerged in an ATM context following the proposal of ATM layer backup path protection. Kawamura, Sato, and Tokizawa ("KST") [\[KaSa94\]](#) were early proponents of ATM restoration using so-called "zero-bandwidth backup VPs." For every working VP, a disjoint backup VP route is defined. The VP identifier and routing table entries for the backup VP are created, logically establishing each backup VP as if it was an ordinary working VP. (The ATM Backup VP scheme is logically identical to MPLS-layer SBPP [\[KiKo01\]](#).) The term 'zero bandwidth' was not actually meant to imply that no additional (spare) capacity allocations would be needed to support restoration. It meant only that until a failure arises, the end-nodes of the backup VPs apply no traffic and so the total bandwidth on the links is available for working flow performance enhancement. The "zero bandwidth backup VP" technique was widely presented at numerous forums, however, and the precise meaning (above) was not immediately appreciated by all—some network operators euphorically reiterated that "the ATM layer is a panacea for restoration—you do protection in the ATM layer and it doesn't require any spare capacity!"^[1] Of course, as stated, this is impossible. From first principles (and regardless of technology—ATM, MPLS, GMPLS, and so on) added or planned-in redundancy is fundamentally required to have any failure recovery. On the other hand, one must volunteer a certain practical truth to the statement in that when built capacities tend to be relatively overprovisioned, the added redundancy to support restoration would often be found when needed in practice. The trouble is that nothing can ever be assured in terms of restoration performance unless redundant capacity is provided in some *planned* way. Thus, as with many technical misunderstandings, there is some basis of truth—that there *are* efficiencies to be exploited in doing restoration in the ATM layer—but things were taken too far—the belief that no redundancy would be needed, and no planning either.

^[1] Note the similarity of the ATM-era misunderstanding about "zero bandwidth VP protection" to the notion of GMPLS auto-reprovisioning as a new solution for restoration problem, but again without consideration of the capacity planning needed if many paths are to be restored simultaneously.

In fact the intent in [\[KaSa94\]](#) was actually only to provide capacity for 100% bandwidth replacement upon *single VP failures*. This is a spare capacity planning problem for ATM networks that is not that much different from a path-restorable STM network if the intent is to exactly replace each failed working VP with a backup VP of equal bandwidth, and consider only single path failures, not mass failures arising from cable-cuts. A capacity allocation algorithm with this intent, which we call the "KST algorithm," was accordingly given in [\[KaSa94\]](#). The KST algorithm plays an important role in this chapter.

The context for the KST algorithm was to provide shared spare capacity allocations that are sufficient for 100% replacement of single VP failures in an ATM layer born over SONET. If the underlying physical layer is already ring or mesh protected against *span* cuts, then independent VP failures would be all that ATM layer would have to recover against. For these cases KST provides just enough additional capacity. The added spare capacity for such single VP failure protection could be very low indeed, as little as ~35% redundancy. However, many in the industry misunderstood the Backup VP protection scheme as a proposal to also recover directly from span failures in the physical layer. The mechanism itself is capable of a response in such circumstances, but the KST capacity planning

algorithm was intended for the single-VP failures alone. To protect against multi-VP span failure scenarios, such low redundancy levels would result in high and unpredictable oversubscription levels. Thus, the question arose: could the essentially uncontrolled oversubscription effects of KST under span failures be somehow limited in their worst-case, while still obtaining KST-like spare capacity requirements?

Interestingly, a great deal of work was done on ATM-based protection switching in general in the 1990s. See for instance [Chapter 6](#) in [\[WuNo97\]](#) and the accompanying list of references. But, as far as we can tell, all of this work focused on the signaling mechanisms, the real-time aspects of the switchover and the capture of backup path capacity, and capacity considerations oriented around the protection of independent single VP (or predefined VP group) failures. But none of these works seem to have directly considered the capacity implications of contention arising from correlated simultaneous failure of multiple VPs arising from a common-cause physical layer failure. A few papers such as [\[LeSo98\]](#) recognized that under physical layer failures, these schemes would often not produce 100% restorability, but emphasized VP/VC splitting or other routing enhancements to try to increase the recovery level, rather than directly planning the capacity as required for 100% restorability under (what we now call) worst case oversubscription limits.

Today the SBPP scheme for lightpath provisioning with shared backup capacity is a close logical cousin of the ATM backup VP scheme, but the issue of common-cause failures is explicitly recognized in SBPP through the SRLG concept: no two primary paths that are members of the same SRLG can share a backup link (because they can fail together). But lightpaths are discrete circuit-like entities; any two either can or cannot share backup capacity on other links. This leaves the capacity planning of ATM or MPLS networks for assured worst case bandwidth allocation effects against physical layer failures as an almost unaddressed combination which defines the main contribution of this chapter. The key logical difference relative to SBPP is that ATM or MPLS paths can (technically) share backup capacity that is needed simultaneously. Our aim is to plan in a way that exploits, but strictly controls, this ability.

The rest of the chapter has two main parts. The first develops a capacity planning framework that permits controlled oversubscription of capacity by design. The idea is to set a limit on the worst-case tolerable oversubscription of capacity allocations, which is simpler and far more practical than dealing with stochastic traffic models directly in the design optimization. The degree of exploitation of the statistical nature of traffic in the capacity planning for restoration is reflected in how this limit is set. Results show that significant capacity savings can be obtained, relative to the corresponding DWDM-like case even with a modest restoration-induced oversubscription of bandwidth. The second part of the work addresses the determination of an acceptable maximum oversubscription level for use in the above design framework. This stage is where statistical details and worst-case congestion implications are considered.

Some note before proceeding: First, a brief background on further aspects of ATM and MPLS technology and concepts is provided in [Chapter 1](#). The design of shared-backup protected lightpath networks *without* oversubscription is covered in [Chapter 6](#) and is assumed background here. Finally, it is important at places in this chapter to remember that the failures we consider arise in the physical layer (of *spans*), but all of the capacity allocations and recovery actions pertain to the logical layer *of links* between ATM or MPLS nodes. A span is the physical aggregation of operating capacity between adjacent nodes. A link is the logical "pipe" between routers or MPLS switches, which may be realized by routing over one or spans of the physical layer.

Each IP/MPLS or ATM link may have many LSPs or VPs routed over it. We will, however, consider networks where ATM or MPLS links are built only over nodes directly connected by a physical span. The convenience of this is that each span failure produces a single link-failure scenario. The methods developed apply, however, for an arbitrary case of multiple link failures arising from a single span failure. Because all restoration is path-oriented in this chapter, the details of logical link routing over physical spans leads only to different sets of end-node path failure pairs for restoration under each single span failure scenario. These can be obtained in advance from more complex MPLS or ATM link topologies by failing each span and including all path-end pairs affected by all link failures arising from each single span failure. From that point the methods of this chapter are applicable in the same way that we use them for the single-span
→ single link failure scenarios.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

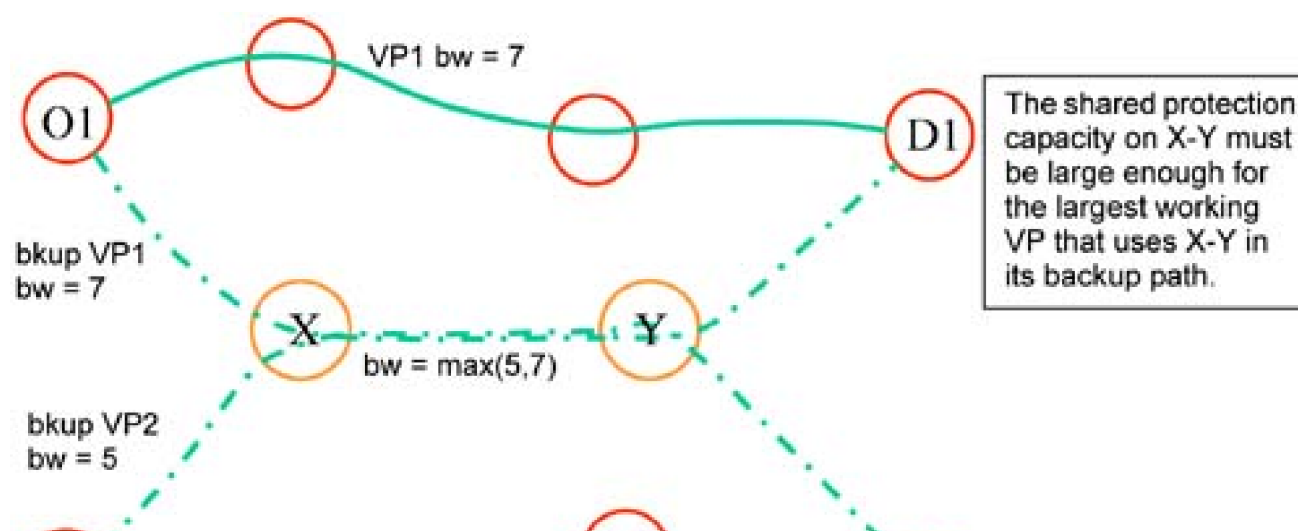
7.2 Overview of MPLS Shared Backup Path Protection and ATM Backup VP Concepts

The Backup VP Protection scheme for ATM proposed pre-failure establishment of a set of backup VPs that would be fully disjoint from each working VP and unused until needed in a failure. The pre-failure setup phase would write the label-swapping entries at the nodes en route so the path is logically established. Upon failure high priority OA&M signaling cells would activate the backup path on detection of the working VP failure at the path terminating nodes and these end nodes would switch label entries to redirect VP flow into the backup path. The nodes terminating the working VP are also the nodes terminating the backup VP. When a failure occurs, the downstream end node of the failed VP detects the failure and sends a restoration message along the backup route and switches the failed VP to the backup VP. Each node that receives the restoration message captures the appropriate bandwidth on the links, and retransmits the message to the next node on the backup route. When the upstream end-node of the failed VP receives the restoration message, it switches traffic from the failed VP to the backup VP. This completes, at least functionally, the restoration process for the failed VP. It is in effect an ATM layer 1:1 APS protocol.

When not in use, the backup VPs consume no bandwidth. But such bandwidth is always there as part of the logical pipe thus always benefiting average cell-level queuing performance. This is a major advantage over the SONET or WDM layers where (aside from "extra traffic on protection" services) spare capacity is strictly separate from working capacity and in no way benefits service performance in normal times. The original proposal for the backup VP planning scheme reported spare capacity levels as low as 35% for full "single failure" restorability. Such low spare capacity was, however, attributable in part to the fact that capacity allocations would have been sufficient to restore any single VP failure, but would have unbounded oversubscription effects under complete span failures. Understanding this serves well to introduce the idea of design for *controlled* oversubscription.

To do so, [Figure 7-1](#) shows the basic principle for shared backup VP protection under the "KST" proposal. The basic orientation is to assure adequate bandwidth for the largest backup VP that crosses a link. The figure gives an example of two backup VPs that are established for restoration using backup paths that both cross link X-Y. The bandwidth allocation of the working VPs is 5 and 7 units respectively. Assuming statistically independent single failures of VP1 and VP2, we allocate the largest bandwidth of the two working VPs that have backup VP assignments crossing the link X-Y. Note that here, and in what follows, traffic on a working VP is restored over only one backup VP. There are quite a few technical reasons (beyond the present scope but easily seen with some reflection) why splitting VPs into constituent VCs for restoration, (or any corresponding splitting of MPLS paths into component subflows for rerouting) is not considered. This is not to say that more than one backup option could not be logically established in advance, with a choice made in real-time, but that only one backup path can be chosen to receive all the working path restoration flow.

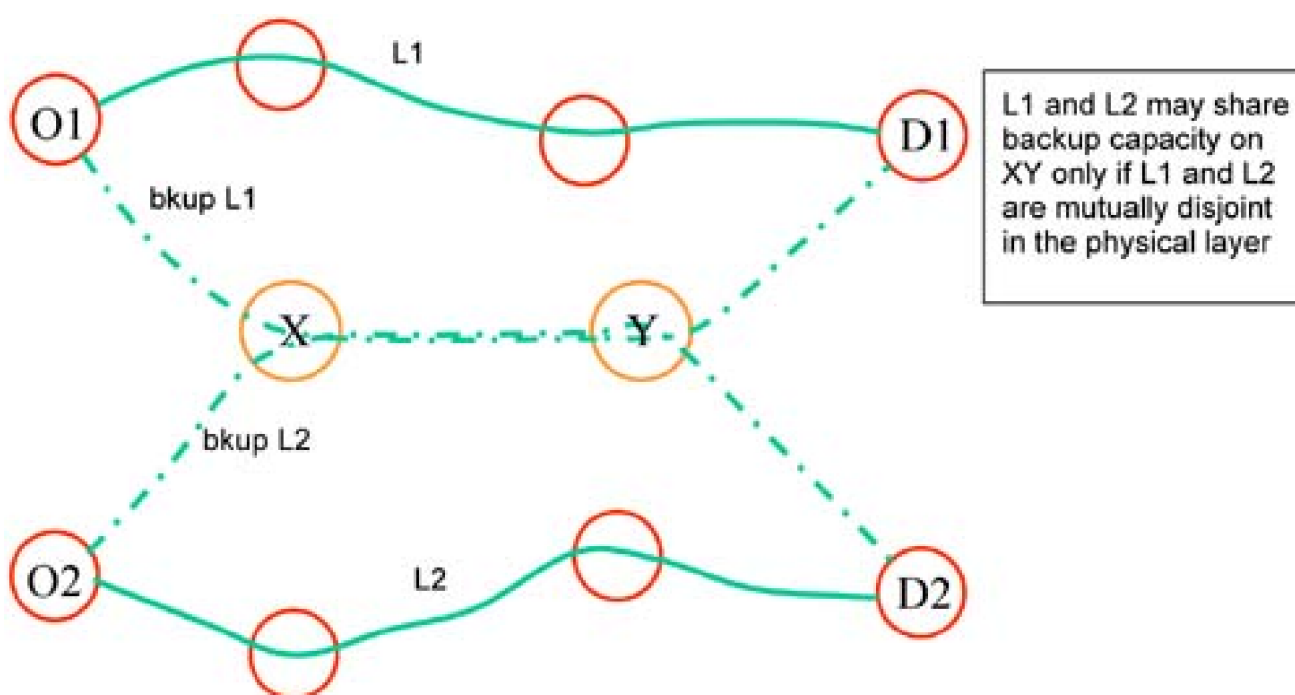
Figure 7-1. KST viewpoint: If VP1 and VP2 fail independently, then link (X-Y) needs only the maximum bandwidth allocation of the two.





Later the MPLS working group of the Internet Engineering Task Force (IETF) was looking at a number of MPLS-based recovery techniques and came up with the shared backup path protection (SBPP) scheme for IP-centric control of either lightpath or LSP restoration. When contemplating lightpath restoration, it is implicit that one is in the domain of physical spans and that there can be no oversubscription because discrete wavelength paths are being manipulated. Consequently, it is a dominant consideration that no two lightpaths which share a span in their backup routes can have anything in common on their working routes. [Figure 7-2](#) shows the basic principle for shared backup path protection from this orientation. In [Figure 7-2](#) each link in the primary path has a dedicated capacity allocation and carries traffic under normal conditions. The capacity allocated on the backup path is shared with backups for other working paths. (In contrast, 1+1 APS has dedicated capacity on both the primary and backup paths.) Any two lightpaths O1-D1 and O2-D2 can have spans in common, such as X-Y, and share channels on that span only if O1-D1 and O2-D2 are span-disjoint.

Figure 7-2. Optical layer viewpoint: If lightpaths L1 and L2 are physically disjoint, then their backup paths may share capacity.

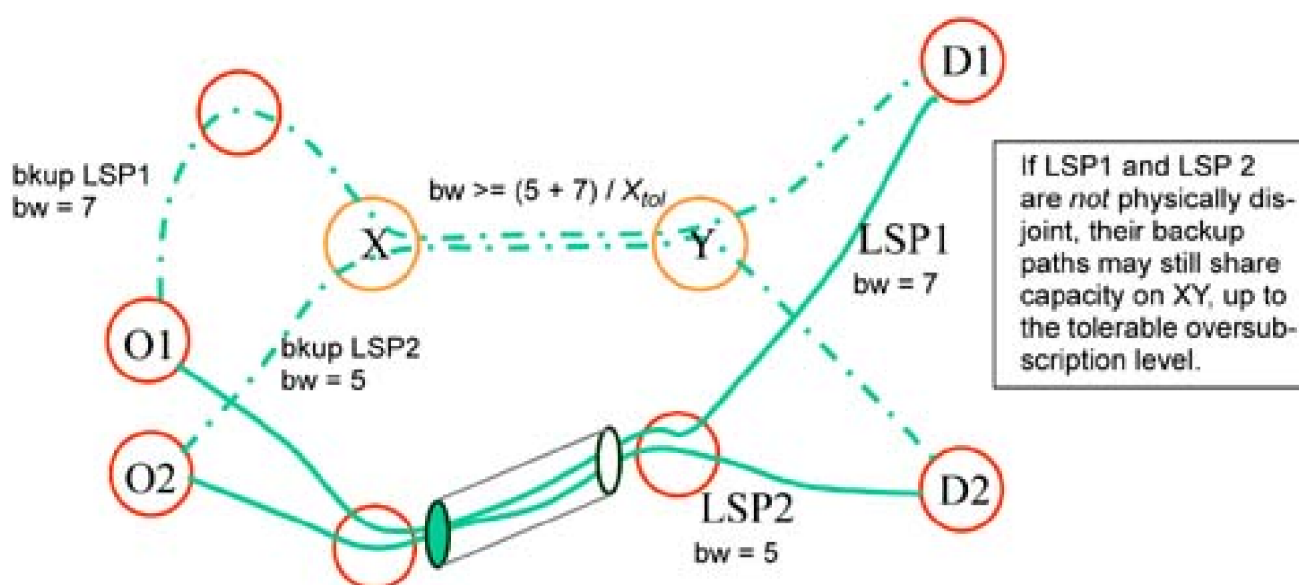


7.3 The Oversubscription Design Framework

Above we have two different viewpoints from which the same basic idea of arranging end-to-end backup paths using shared protection capacity has been approached. The mechanism for backup path activation is logically identical—only the capacity planning philosophies differ. Capacity planning with controlled oversubscription unifies these two viewpoints under a single more general framework. In the shared backup lightpath context it is a basic requirement that any two working paths with backup paths sharing the same spare capacity resources must be completely disjoint in the physical layer. In the oversubscription framework, this is the same as completely eliminating oversubscription—because there is always an assured match of restoration bandwidth to failed working path bandwidth. Conversely, in the KST context, the paramount idea is of finding a logical set of backup VP routes that maximally reuse protection capacity allocations over independent VP failures. In the oversubscription framework, this is the removal of *any* limit on the magnitude of oversubscription effects under span failure.

Our interest is in capacity planning that is at a controlled point between these extremes of *no* oversubscription (as in SONET, WDM) and *uncontrolled* oversubscription (as in GMPLS auto-reprovisioning). This approach will not provide for complete replacement of each working LSP or VP's normal bandwidth under all restoration scenarios, and also does not require strict disjointness between working paths, but it is mindful of the oversubscription effects. [Figure 7-3](#) shows how this idea of oversubscription-based design relates to the two previous principles for capacity planning. Here, LSPs 1 and 2 are not fully disjoint. They share a common physical layer span as shown. When this span is considered as a failure scenario the philosophy will be to allow both backup paths to be activated and to impinge on link X-Y as long as the total oversubscription factor on X-Y is not more than a tolerable design limit, X_{tol} . For example, if $X_{tol} = 1.33$ ("33% oversubscription"), link X-Y would require a bandwidth allocation of 9 (or more) in [Figure 7-3](#). The example considers however, only one failure scenario and disregards ordinary working capacity allocations on link X-Y and other details the rest of the chapter will consider.

Figure 7-3. Capacity planning view for controlled oversubscription.



A partial analogy to motivate this approach is found in the airline business where most flights are slightly overbooked as part of an overall policy for revenue maximization. Most often, the overbooking is unseen to users unless all the passengers show up. Therefore, in an ATM or MPLS network, could we not similarly overbook the protection capacity somewhat? Unless a failure occurs right when working path utilizations are simultaneously at their peaks, the oversubscription of restoration capacity may remain almost unnoticed by customers if the initial bandwidth allocations were not operating at their design utilizations right at the time of failure. Even the payment of penalties for *realized* overbooking situations, which is still part of an optimal revenue strategy for the airlines, has a corresponding aspect of a potential network services business model that employs oversubscription in its capacity planning strategy, although this is beyond the current scope.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

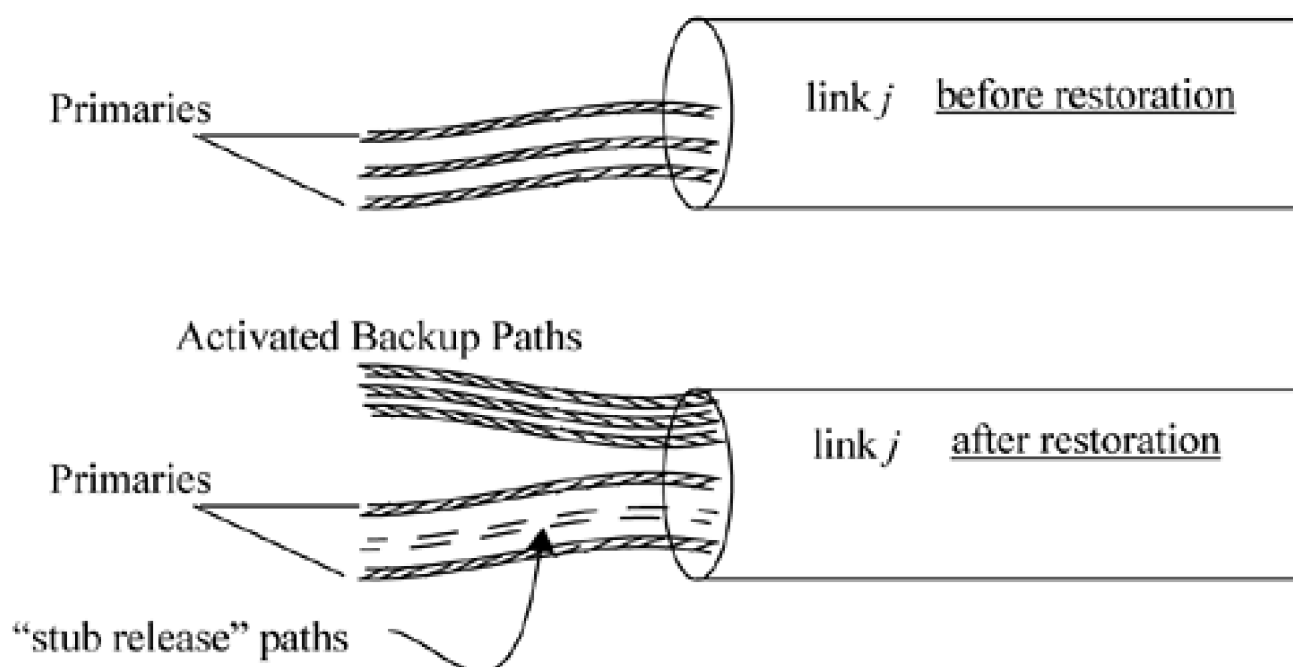
◀ PREVIOUS

NEXT ▶

7.4 Defining the Oversubscription Factor $X_{j,i}$

Let us now precisely define the oversubscription factor of a link in the general situation of an ATM or MPLS network in a restored state. Rather than carry forward the terminology of both LSPs and packets for MPLS and VPs and cells for ATM, let us henceforth also adopt the generic terms primary or working path, backup path, and traffic. In [Figure 7-4](#) assume that link j has a bandwidth allocation that is based on its nominal working load and an allocation of additional capacity for restoration. These working and spare bandwidth allocations are not integral transmission units and spare capacity is not physically separated from working. Rather, the total bandwidth is present for all traffic, although its determination may have depended on separate calculations of working and protection capacity. Upon failure of some *other* link i , all primaries going through link i are switched to their backup routes over other links. In the general case some of the primary paths on the failed link i use link j in their backup routes. These backup were logically present on link j prior to failure but consumed no bandwidth. Upon failure, however, redirected packet flows appear in each backup LSP imposing an additional load on link j .

Figure 7-4. Logical view of a surviving ATM or MPLS link in a restored network.

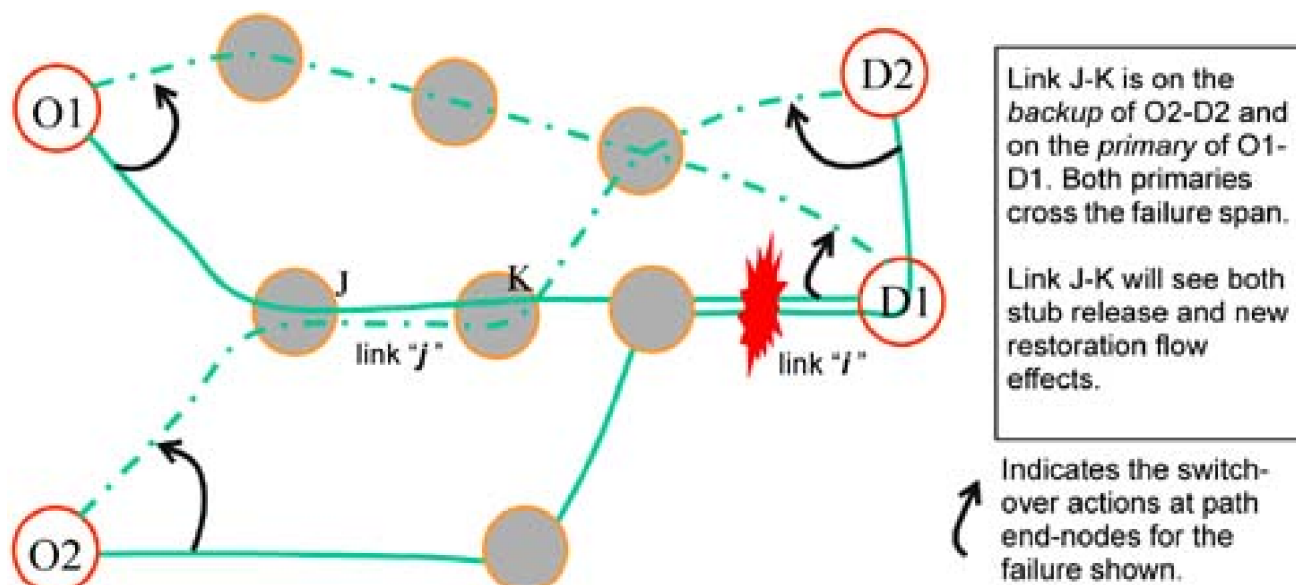


Additionally, when considering complete link or span cuts, there may also be a *reduction* in traffic on link j . If one or more of the normal primary paths crossing link j also traversed the failure link i (upstream or downstream of link j on its path) there is a "stub release" effect, in the same sense that we saw for path restoration in [Chapter 4](#). The stub release effect is not a choice or an option, however, in this case. Upstream and downstream of the failure, the path stubs are still logically present but now, because of the failure itself, they suddenly have no traffic on them. Therefore, a surviving link j may see both a disappearance of working flow from some of its primaries and a sudden onset of new flow from activated backup paths that traverse it. Thus, there are three types of traffic on a surviving link after restoration:

1. Baseline working traffic on the link j that is unaffected by the failure.
2. Disappearance of traffic from primary paths that cross the surviving link j but also crossed the failure link i , upstream or downstream of the current link.
3. Newly asserted traffic arising from activation of backup paths that cross the link j .

[Figure 7-5](#) gives an example showing how a failure on a certain link i may result in *both* a stub release effect and the imposition of new flows at the same time on another link j . When the failure shown occurs, path O1-D1 drops to zero load on link J-K, but at the same time the backup for path O2-D2 is activated and it traversed link J-K.

Figure 7-5. Illustrating stub-release and backup flow activation effects from a single failure.



We can therefore summarize the restoration-induced capacity oversubscription factor $X_{j,i}$ of a link j in response to failure of another link i as:

Equation 7.1

$$X_{j,i} \equiv \frac{W_j - Rs_{j,i} + Rr_{j,i}}{W_j + S_j} \quad \text{or} \quad X_{j,i} \equiv \frac{W_j - Rs_{j,i} + Rr_{j,i}}{T_j}$$

where:

- $Rs_{j,i}$ is the total allocated bandwidth of primaries that cross both link i and link j .
- $Rr_{j,i}$ is the total allocated bandwidth of primaries on link i whose backups cross link j .
- W_j is the total allocated bandwidth of primaries on link j . [2]

[2] Different symbols (W_j and S_j) are used in this chapter (as opposed to w_j and s_j elsewhere) to recognize that W_j and S_j are both real-valued capacity allocations made to a link (not integer channel counts) and that the distinction between them is not the same as between w_j and s_j . W_j and S_j are both allocations of bandwidth that are used by working flows during normal times. It is just that S_j allocations are made for a different *reason* than the W_j s—to provide the *extra* capacity that is needed to support restoration objectives.

- S_j is the total spare bandwidth allocation on link j .
- T_j is the total available capacity on link j .

Note that $X_{j,i}$ is based on *allocated* primary and backup capacity not the actual traffic intensities (or utilizations). Thus, an oversubscription of bandwidth due to restoration switching is not the same as an *overload* in actual traffic impinging on the link. Any cell or packet-level overload that occurs depends on the actual traffic intensity in each primary path involved, not their bandwidth allocations per se. The worst case traffic overload represented by a certain oversubscription factor would be the oversubscription factor value itself as a multiplier on the offered load to the link. This would occur only if the actual packet level utilization of all working paths involved was 100% at the moment of failure, i.e., each LSP or VPs offered load was exactly that for which its nominal bandwidth allocation was initially designed. Note that nothing requires that $Rs_{j,i} < Rr_{j,i}$ and that consequently it is quite possible for $X_{j,i} < 1$ to arise. This just means that the removal of load due to path failures upstream of span j exceeds the activated backup path allocations, leaving the link *undersubscribed* in failure scenario i .

The idea of working with capacity allocations is that, for network planning purposes, the over-*subscription* factor does represent the worst case packet-level over-*load* that could arise, but by manipulating only bandwidth allocations, we avoid the need to directly model time-varying cell or packet level traffic details. A separate set of modeling or simulation studies (and/or business judgements) can define a maximum tolerable oversubscription factor, $X_{tol} > 1.0$, based on the worst case packet-level congestion or overload that is considered tolerable under a restored network state.

As defined $X_{j,i} = 1.0$, $\forall (j,i)$ is a basic property of SONET or lightpath restoration because the total bandwidth of paths for replacement of failed transport signals is always equal to the failed capacity. There is no option of partly replacing one or more failed lightpaths. Either each is replaced exactly by a matching restoration path or all services on the given lightpath experience immediate total outage. In MPLS or ATM, however, $X_{j,i} > 1.0$ means only that link j 's total bandwidth is oversubscribed with respect to the initial capacity allocations to path involved when link i fails. This can be thought of as a logically restored state but in which there is a partial shortage of bandwidth. Service connections may be affected in terms of cell loss and delay performance, but none are immediately terminated or disconnected as they would be in STM if there was not 100% replacement of working capacity. Here there is no outage, but neither is the prefailure bandwidth of each primary 100% restored either. Whether packet level performance exceeds QoS requirements under $X_{j,i} > 1.0$ will depend on the actual primary path utilizations and traffic parameters at the time of failure. Actual congestion levels on link j can rise and fall in a time-varying way depending on how traffic varies in the rerouted primaries during the duration of the restored network state, but we know they will not exceed $X_{j,i}$ times the nominal load on the link if the load on the corresponding primaries stays within its normal maximum for their own allocated bandwidths.

Thus we have a capacity planning framework that deals with bandwidth allocations, not stochastic flows, which has the ability to guarantee limits on the worst case stochastic flow congestion effects. In fact because additional statistical multiplexing gain always arises through the confluence of flows in a larger shared bandwidth, the oversubscription factor is a somewhat pessimistic estimator of the effective worst case overload in the oversubscribed situation. For example, if the flows from two 1 Gb/s links at 20% utilization are merged on link at 1.5 Gb/s, the oversubscription factor would be 33%, but the actual cell level performance at 26.6% utilization in the merged link will be not as bad as it would be in either of the original links had their loads been simply increased 33%.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

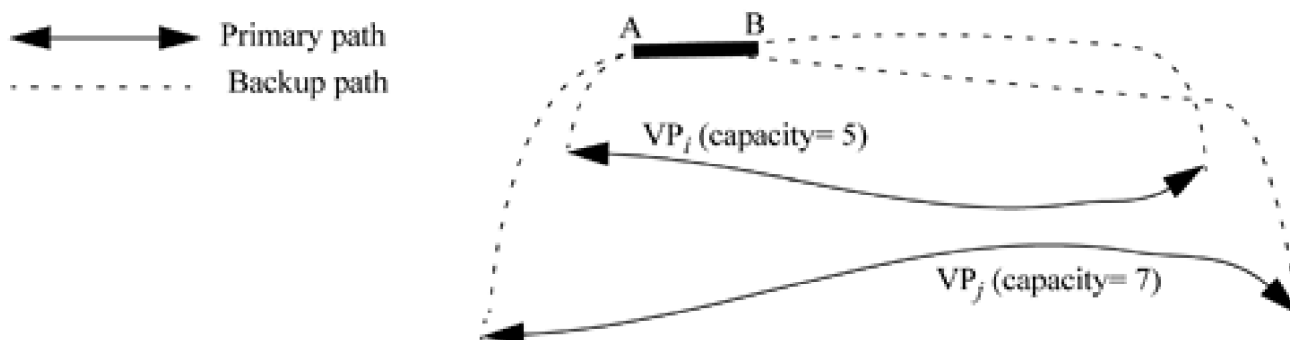
7.5 KST Algorithm for Backup Path Capacity Allocation

Let us now look at *KST-Alg* which provides backup path capacity placement that would efficiently guarantee 100% restoration of any *single* primary *path* failure. We will then use the oversubscription measure to analyze its effects under *single-link* (or span) failure scenarios. There are several reasons for studying this algorithm as a first step to what follows. It illustrates an effective general heuristic for optimization of capacity sharing relationships for protection and it has aspects evocative of both the "max-latching" idea in for span-restorable capacity design [GrRa97] and to simulated allocation in Section 6.16. Treating this algorithm is also instructive to show why we must consider backup path capacity allocations with respect to entire link or span failures, not just from single independent path failures. This is an important point in dealing with any scheme where backup path allocations may be made one at a time on a per working path basis at the time each path is provisioned. Unless the backup arrangements are dedicated, the backup route and capacity must always consider all previously provisioned service paths and their disjointedness and sharing relationships on the physical layer graph. This is an extremely important but easily overlooked consideration. The KST algorithm and its implications thus aide us to set up the next idea of formally designing for controlled maximum oversubscription levels. The application of KST to link failure recovery gives a striking illustration of how serious the implications can be if we plan for only a single path failure at a time, but then encounter correlated multiple path failures due to a common link or span failure. The results also help explain why the spare capacity ratios for restorability against link failure are not as low as was widely thought at first, following widespread publication of the "zero bandwidth" ATM backup VP scheme [KaSa94] [KaTo95] [KaO99].

In *KST-Alg* all primaries are initially routed via shortest paths. A shortest disjoint backup route is then assigned for one of the primaries to start with, creating an initial allocation of spare capacity on some links. The second and all subsequent backup path choices will then try to reuse the spare capacity allocations made in prior steps, by choosing among backup route options so that the least new spare capacity must be added to realize each additional backup path. More generally each eligible backup route is tested to find the one which requires the least (if any) new spare capacity, given the current state of spare bandwidth allocations already placed for previously decided backup routes. When all primaries have an initial backup route assignment, each primary is revisited in a second master iteration to see if a different choice from its set of eligible backup routes could yield a lowering in total spare capacity. The latter level of iteration repeats until no further reduction in total spare capacity is found.

Figure 7-6 illustrates the principle. In the example, link A-B is on the possible backup routes for primary *i* (capacity 5) and *j* (capacity 7). Assume *KST-Alg* has first considered primary *j*, selected the backup shown and accordingly assigned link A-B a spare capacity of 7 units. Once link A-B has 7 units of spare capacity, *KST-Alg* will recognize that in considering the next primary that if it uses link A-B in the backup path for *i*, there will be no spare capacity cost arising on A-B because primary *j* needs only 5 units of bandwidth and 7 are present. This reuse of link A-B in the example assumes that *KST-Alg* also finds that the rest of the backup path for primary *i* is suitably efficient on other links as well. A more complete statement of *KST-Alg* is given in the following pseudo-code specification where *k* indexes the demand pairs requiring primary paths:

Figure 7-6. KST-Alg: backups are chosen to reuse spare capacity placed for other primaries.



KST Algorithm (G(N,S), source and target pairs, demandsd(k)):

```

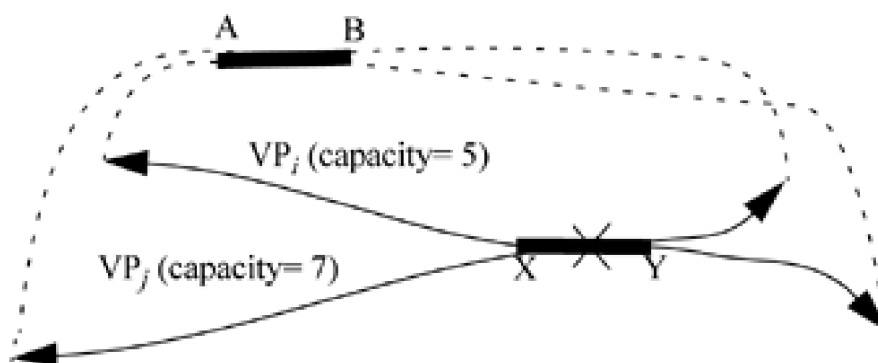
For every primary path, k:
    Find a set of eligible backup routes b in B(k) that are
    disjoint from the shortest route for primary k;
For every span i in S: s(i):= 0;
For every primary path k:
    For every eligible backup path b in B(k):
        back_cost := sum(over i in b): max(0, d(k) - spare(i));
    Choose backup b*(z) from B(k) s.t. back_cost is minimal;
    Update: for every i in b*(z) spare(i) = min(spare(i),d(k));
Repeat
old_tot_spare:=sum(over i): spare(i);
For every primary path k:
    For every eligible backup path b(z) in P(k):
        back_cost := sum(over i in b(z)): max(0,d(k)-spare(i));
    Assign new_b(z) from P(k) s.t. back_cost is minimal
    If new_b(z) not= b*(z):
        For every i in b*(z):spare(i):= max(0, spare(i)-d(k));
        For every i in new_b(z):spare(i) := max(spare(i),d(k));
        b*(z):= new_b(z);
        new_tot_spare:=sum(over i):spare(i);
Until old_tot_spare = new_tot_spare (no further reduction found);

```

So why does *KST-Alg* not generate a completely restorable network design in the sense of link or span failures? Every primary has a predetermined backup with adequate spare capacity allocated on each of its links to support 100% replacement of the bandwidth on the failed primary. While this would indeed protect against single path failures, the problem is that if primaries *i* and *j* happen to share the same physical link, for instance, X-Y, then in case of X-Y failure, primaries *i* and *j* will be rerouted simultaneously onto backups which traverse link A-B as illustrated in [Figure 7-7](#). Omitting any stub release effects for the example, and assuming $\sum W_j$ of 9 units on A-B, the result of link cut X-Y is an oversubscription of the allocated bandwidth on A-B. If the baseline working allocation for A-B was 9 units, then failure of link X-Y shifts another 5+7 units of required capacity onto link A-B. But A-B only has an actual capacity allocation of 16 units (9 working plus the spare capacity assignment of 7 above). Thus, the restoration induced oversubscription factor $X_{A-B, X-Y}$ is $21/16 = 1.31$. Clearly, what is missing are considerations on how to coordinate the set of backups for each physical link failure as a simultaneously instantiated group of paths.

Figure 7-7. KST-Alg: Backups chosen to reuse spare capacity placed for other primaries.

Link A-B:
 working = 9
 under KST, spare = 7



7.6 Oversubscription Effects with KST-Alg

KST-Alg was implemented in [\[Zhen97\]](#) to reproduce and test the predictions of quite low spare capacity made in [\[KaSa94\]](#), and to check for restoration-induced oversubscription effects as be expected from the reasoning above. Considering that the number of distinct routes in a network of $|S|$ links is $O(2^{|S|})$, the number of backup route for each primary was restricted in practice. (Although limited, when later comparing KST to results from later MIP formulations, the *same* set of backup route choices is used in both cases.) One approach is to enumerate all distinct eligible routes up to a prescribed hop count, but a limitation of that is that there may be thousands of backup routes for some primaries while few disjoint routes can be found for other primaries under the same hop limit. Consequently, a k -successive shortest distinct routes method was adopted to generate a set of distinct backup route options for each primary path. We select eligible routes from this subset in the results presented. An issue we have seen before ([Section 6.9.1](#) and Section 4.74) is that in some situations there are no disjoint backup routes if the shortest path is taken for the primary. To avoid this in results that follow, each shortest path is checked first to see if it has a feasible backup path. If not, we choose the next shortest route for the primary which does have a link disjoint backup. *KST-Alg* is then applied to choose the actual backup route set to use and to make the corresponding spare capacity allocations.

After *KST-Alg* has assigned all backup routes and spare capacity, we checked the oversubscription factors on all other links for each single span cut by actual experiments on each designed network. For each failure scenario, each simultaneously affected primary is rerouted on its preplanned backup. We then assess all traffic effects (including working, stub release and backup) in the definition in [Equation 7.1](#) to obtain the restoration-induced oversubscription on all links for each link cut. For a network containing $|S|$ links, each of $|S|-1$ other links has an oversubscription factor upon a link failure. Therefore, the total data set of oversubscription factors is an $(|S|-1)$ by $|S|$ matrix.

Five networks and demand matrices previously studied for STM restoration [\[IrMa96\]](#) were used to test *KST-Alg*. The characteristics are detailed in [Table 7-1](#).

Table 7-1. Test Network Characteristics

network	# nodes	# links	# demand pairs	# primaries
Smallnet	10	22	45	79
Net 1	15	28	67	68
Net 2	20	31	153	153
Net 3	30	59	263	271
Net 4	53	79	347	418

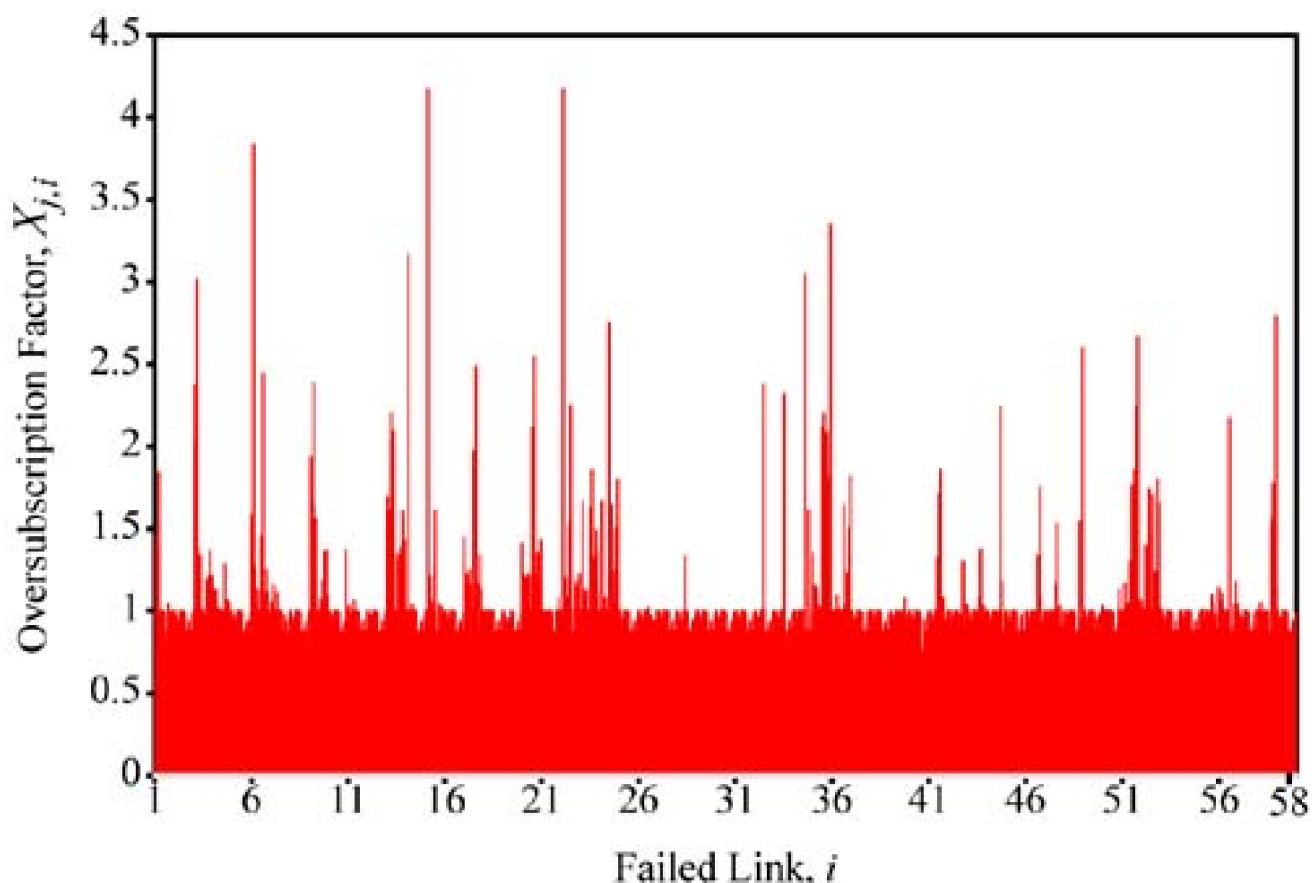
[Table 7-2](#) shows the distance-weighted redundancy requirement with *KST-Alg* and the corresponding oversubscription effects. Average oversubscription is the mean oversubscription value of all cases where oversubscription factors are greater than or equal to 1.0. Maximum oversubscription is the case with the largest oversubscription factor. The table shows that *KST-Alg* indeed generates low redundancy, as low as about 29% in Smallnet to only 30 to 40% in two other networks. These spare capacity predictions are indeed much lower than those generally required by STM networks. It is important, however, to note that these particularly low spare capacity levels are accompanied by uncontrolled oversubscription effects. With the levels of oversubscription reaching 3 to 10 times nominal traffic load, cell loss and delay in MPLS networks would be intolerable for many applications.

Table 7-2. Spare Capacity and Oversubscription in Designs with *KST-Alg*

Network	Redundancy	Average Oversubscription	Maximum Oversubscription
Smallnet	28.6%	1.14	1.42
Net 1	51.7%	1.39	3.00
Net 2	54.3%	1.37	3.32
Net 3	31.2%	1.37	4.16
Net 4	38.6%	1.46	10.00

[Figure 7-8](#) illustrates the detailed oversubscription factor analysis of Net 3. The diagram shows the oversubscription factor over all surviving links when each link is failed. For each link considered as the failure link i on the x-axis, the $(|S|-1)$ values of $X_{j,i}$ experienced by other links are plotted left to right with a vertical line for each value. The x-axis is labeled with the failure link names and in the fine scale the oversubscription factors of all the surviving links plotted. For example, in a network of 10 links, there would be ten clusters of nine $X_{j,i}$ values displayed side by side to form the plot. The plot shows that a large number of oversubscription cases are involved in *KST-Alg* design. The largest oversubscription is about 4.16 in Net3.

Figure 7-8. Oversubscription factors arising from KST backup capacity under link failures.



At this point we can see the problem of oversubscription effects in trying to share backup path capacity allocations. But we can also see an opportunity, inspired by *KST-Alg*. The capacity savings of *KST-Alg* relative to STM networks are attractive but the oversubscription implications are unacceptable. What if we could build on the idea of KST to formulate capacity allocation methods that will still gain much KST-like capacity savings but without exceeding an *allowed maximum* of restoration-induced oversubscription? These are the key ideas of the next sections.

7.7 Minimum Spare Capacity with Limits on Oversubscription

Let us now consider a first formulation to minimize the spare capacity of a shared-backup path protected network given a peak allowable oversubscription factor in the network. In the first formulation we assume primaries are routed over shortest paths except where the shortest path choice prohibits any disjoint backup paths. In the latter case, a separate program provides a different primary path specification and a corresponding set of disjoint backup route choices as inputs. The parameters of the model are:

- S = the set of all physical layer spans in the network, indexed by z .
- L = the set of all service layer logical links in the network, indexed by j .
- $F(Z)$ = the set of logical links routed over physical span z (i.e., the set of links that fail simultaneously if span z is cut). More generally this is the set of all simultaneous link failure scenarios to be considered. This set is indexed by i .
- C_j = the cost of link j per unit of capacity allocated (can include length dependency).
- W_j = the total working capacity allocation on link j . This is generated as an input to the model based on the shortest-path routing of primary paths prior to spare capacity minimization, but is also needed as part of the oversubscription calculation.
- P^r = the set of disjoint eligible backup routes for the primary path on relation r .
- $\xi_i^r = 1$ if the route of the primary on relation r crosses link i , zero otherwise ^[3].

^[3] By a *relation* here we mean a specific pair of communicating applications that terminate the logical path. This allows that each O-D pair may support more than one logical path at a time.

- $\delta_{k,j}^r = 1$ if backup path k for the primary path on relation r crosses link j , zero otherwise.
- R^r = the nominal target bandwidth replacement level of the primary path on relation r .
- D = the set of all relations to be served, indexed by r , demand magnitudes d^r .
- X_{tol} = an oversubscription factor deemed to be the maximum tolerable on any link under any restored network state.

The variables to be solved for are:

- $\alpha_k^r = 1$ if backup path k is chosen for primary path r , zero otherwise (binary).
- S_j = the spare capacity allocation on link j (real).

The objective function is:

Minimize

Equation 7.2

$$\sum_{j \in L} C_j \cdot S_j$$

Subject to:

1. Spare capacity allocations are sufficient to keep oversubscription effects below the design limit, X_{tol} , for all failure scenarios. This expression is expanded below into its dependence on the primary variables, but stated in its conceptual role here:

Equation 7.3

$$X_{j,z} \leq X_{tol} \quad \forall z \in S; \forall j \in L | j \notin F(z)$$

2. Upon failure of any primary path, its target bandwidth replacement level (which may not necessarily be 100% to allow for various classes of service) is provided for in its individual backup path allocation.

Equation 7.4

$$f_k^r = \alpha_k^r \cdot d^r \cdot R^r \quad \forall k \in P^r; \forall r \in D$$

3. Only one backup path may be used for each primary path, i.e., flows are not split:

Equation 7.5

$$\sum_{k \in P^r} \alpha_k^r = 1 \quad \forall r \in D$$

Several notes on this model will be helpful. First note that W_j values are not included in the objective function for minimization along with spare capacity because in this model all primary path choices, and hence working capacity allocations to logical links, are constants that can be determined as input prior to running the optimization of spare capacity. Secondly, for actual implementation, [Equation 7.3](#) needs to be put in terms of the primary constants and variables. This is done by recognizing that the general expression for oversubscription of link j in response to failure of span i , [Equation 7.1](#), is extended to cover all links in a more general failure scenario z , i.e., $X_{j,z}$ as:

Equation 7.6

$$X_{j,z} = \frac{W_j - \sum_{i \in F(z)} \left\{ \sum_{r \in D} d^r \zeta_i^r \zeta_j^r \right\} + \sum_{i \in F(z)} \left\{ \sum_{r \in D} \sum_{k \in P^r} f_k^r \delta_{k,j}^r \zeta_i^r \right\}}{(W_j + S_j)}$$

$R s_{j,z}$ —stub release traffic lost from link j after failure of all

$R r_{j,z}$ —restoration traffic crossing link j after failure of

links i in failure scenario z .

all links i in failure scenario z .

To put this into an implementable form for use in an LP solver, we rearrange as follows:

Equation 7.7

$$S_j \geq \frac{W_j \cdot (1 - X_{tot}) - R_{s_{j,z}} + R_{r_{j,z}}}{X_{tot}}.$$

After substitution for $R_{s_{j,z}}$ and $R_{r_{j,z}}$ simplification yields the following constraint system, in a final usable form:

Equation 7.8

$$S_j \geq \frac{\left(\sum_{i \in F(z)} \sum_{r \in D} \left\{ \sum_{k \in P^r} f_k^r \delta_{k,j}^r \zeta_i^r - d^r \zeta_i^r \zeta_j^r \right\} + W_j \cdot (1 - X_{tot}) \right)}{X_{tot}}$$

$$\forall z \in S; \quad \forall j \in L \setminus j \in F(z)$$

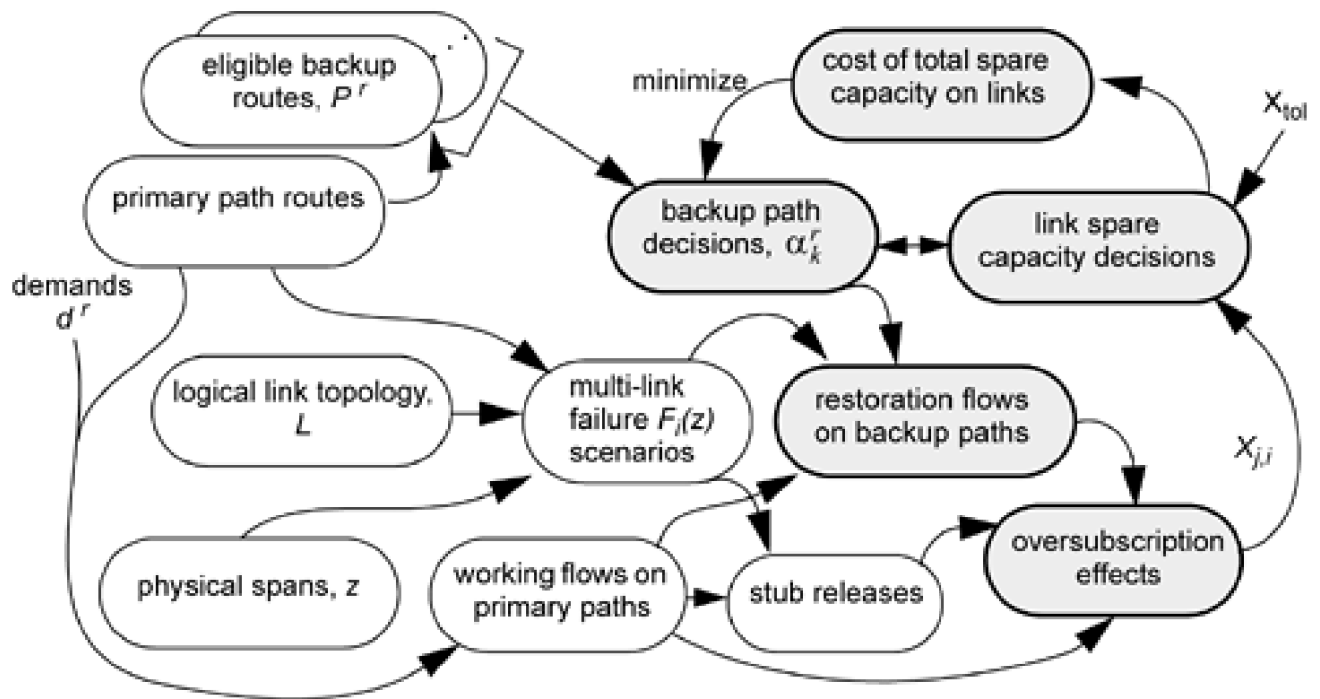
The main output variables are S_j , the spare capacity bandwidth allocation on each logical layer link, and the decision as to the routing of the backup path for each primary path, α_k^r . The physical span capacities that are required are easily generated from the solution information by reference back to the original information on the routing of primary paths over physical spans, i.e., . As expressed above, the intermediate variables f_k^r represent the restoration flow on each eligible backup path for each primary, but are entirely dependant on the backup route decision variables α_k^r and the bandwidth replacement target for the corresponding primary path. As such these intermediate flow variables can be eliminated from the actual implementation, by substituting [Equation 7.4](#) into [Equation 7.8](#), although they are included for clarity of presentation above.

In addition to eliminating f_k^r , a second tactic to speed solution of this model is to set the MIP solver priority or α_k^r according to its associated primary path demand volume, d^r , so that the backup paths for larger capacity primaries are decided first. Additionally, because only one backup route can be chosen for each primary, it is useful to set the branch parameter to "up" which causes the selected branching variable to be set to one, and then forces all the rest of the variables in the constraint to zero, which eliminates all the infeasibilities in that constraint. Note also that the spare capacity cost of *KST-Alg* itself is usually a lower bound on OS-1 for any finite X_{tot} .^[4]

[4] We say "usually" because the intent of *KST-Alg* is to minimize capacity with no limit on X_{tot} —which certainly positions it as a lower bound, conceptually. On the other hand *KST* is also a heuristic, so sub-optimality in achieving its goal could allow that for some suitably high X_{tot} values, OS-1 may require less capacity than *KST*.

[Figure 7-9](#) portrays the relationships between input information (unshaded), the decision variables and the internal calculations that are part of the optimization itself (shaded). Sample results obtained with OS-1 follow consideration of two further oversubscription based capacity planning models.

Figure 7-9. Understanding the relationships in the OS-1 design model.



7.8 Minimum Peak Oversubscription with Given Spare Capacity

Let us now consider controlled oversubscription design in another context, that in which there is an existing set of spare capacity allocations and the aim is to route and protect all demands such that over a stipulated set of failure scenarios, the worst case oversubscription factor is minimized. Parameters and variables are the same as in OS-1, except that now S_j is an input parameter, not a variable. The backup path choices are now the sole variables determining the extent to which we can curtail the worst case oversubscription effects due to restoration, for the given physical and logical topology, and primary routes and flows. This formulation can be used to minimize the worst potential impact of restoration in situations where there is not enough spare capacity for complete restoration without oversubscription.

The objective is:

Minimize:

Equation 7.9

$$\{ \max_{j,z} (X_{j,z}) \quad \forall z \in S; \quad \forall z \in S; \forall (j \in L | j \notin F(z)) \}$$

where $\max_{j,z}(X_{j,z})$ is the peak restoration-induced oversubscription resulting on any non failed link j over all stipulated failure scenarios, z . To achieve the minimization of a maximum value (which is not a linear objective function) an additional constraint and intermediate variable Y is defined. Y effectively acts as $\max(X_{j,z})$ through the added constraint will insist that all $X_{j,z}$ must be less than Y . We then put Y in the objective function. Thus, [Equation 7.9](#) is replaced by:

OS-2

Minimize:

Equation 7.10

Y

Subject to:

1. Ceiling constraint on oversubscription factors:

Equation 7.11

$$X_{j,z} \leq Y \quad \forall z \in S \quad \forall j \in L | j \notin F(z)$$

This is an example of the general technique pointed out in [Section 4.16.3](#) for handling a *min* (*max*()) type of objective function by creating an intermediate "ceiling" variable and putting the ceiling value itself under the minimization pressure. A nonlinear objective function is effectively replaced by a family of constraints only one of which will generally be active at any time. To complete [Equation 7.11](#) we need to substitute the detailed expression that mechanizes $X_{j,z}$ which in this case is the same as [Equation 7.6](#). The remaining constraints follow from OS-1:

2. [Equation 7.4](#): Primary paths are provided with their target bandwidth replacement level.
3. [Equation 7.5](#): Only one backup path may be used for each primary path.

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

7.9 OS-3: Minimum Total Capacity with Limited Oversubscription

One more design model remains to be defined. In both of the above models, we accept as input a fixed primary path for each relation, and a set of corresponding disjoint path options for each primary from which a backup path can be chosen. We now consider the extension of OS-1 to simultaneously optimize the primary routes *and* the backup spare capacity placement for minimum total link capacity subject to a maximum tolerable oversubscription ratio for any restoration event. This is technically an ambitious formulation, which can easily be made too large to run in a reasonable time, but which we used for research purposes to assess the potential benefit from attempts to fully solve or approximate the joint optimization problem. There are additional parameters required to describe the options in primary path routing:

- Q^r is the set of distinct eligible working routes for relation r , indexed by q .
- $c_i^{r,q} = 1$ if the route of primary path q for relation r crosses link i , zero otherwise.

The dimensionality of the parameters describing backup path options is increased as there must now be a set of eligible backups for each different primary path option:

- $\delta_{k,j}^{r,q} = 1$ if the k^{th} backup route available for primary path q on relation r crosses link j , zero otherwise.

A new set of decision variables will specify the choice of primary path routes:

- $b^{r,q} = 1$ if the q^{th} eligible primary route is chosen for relation r , zero otherwise.

The objective function now becomes:

OS-3

Minimize:

Equation 7.12

$$\sum_{j \in L} C_j \cdot (W_j + S_j)$$

Subject to:

1. Link j 's working capacity is sufficient to meet the prefailure demands of all primary paths which cross it:

Equation 7.13

$$W = \sum_{r \in R} \sum_{q \in Q^r} b^{r,q} \cdot d^r \quad \forall i \in I$$

$$r_j = \sum_{r \in D} \sum_{q \in Q} \beta_j^{r,q} \quad \forall j \in P \quad \forall r \in D \quad \forall q \in Q$$

2. Only one primary path can be used for each relation:

Equation 7.14

$$\sum_{q \in Q} \beta_j^{r,q} = 1 \quad \forall(r, q)$$

plus:

3. [Equation 7.8](#): Spare capacity sufficient to keep $X_{j,z}$ below the limit, X_{tol} , for all failures.
4. [Equation 7.4](#): Primary paths are provided with their target bandwidth replacement level.
5. [Equation 7.5](#): Only one backup path may be used for each primary path.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

7.10 Related Bounds on Spare Capacity

Two simple algorithms are presented here to calculate reasonably tight upper and lower bounds on the required spare capacity of a shared-backup path protected network involving oversubscription considerations. These bounds provide a check on the results to follow for OS-1 through OS-3 and they may also be used as relatively quick procedures yielding bounds that help speed the solution of the OS-1 and OS-2 MIP problems. The lower bounding procedure in particular may be useful to rapidly generate starting point designs for large networks, with the MIP-based optimization used subsequently to reach a final complete design.

7.10.1 Upper Bound Based on KST Algorithm

The upper bounding algorithm is based on *KST-Alg* with a simple modification to strictly eliminate any oversubscription. In *KST-Alg*, the spare capacity of a link is set as the largest requirement of all primary paths whose backup routes cross the link. If several primary paths simultaneously fail upon one link failure or span cut, the network may suffer from high oversubscription. Clearly if the spare capacity on each link were set to be the *sum* of all the capacities of primary paths whose backups traverse it, then the capacity would be adequate to support restoration of all primaries with exact capacity replacement and we would have an upper bound on the required spare capacity. For example, in [Figure 7-6](#), the upper bounding algorithm would say that link A-B needs $7+5 = 12$ units of capacity, rather than $\max(7,5) = 7$ units as *KST-Alg* does. This results in an overprovisioned design with a guaranteed maximum oversubscription factor of 1.0.

7.10.2 Lower Bound Based on the LP Relaxation of OS-1

The lower bound is based on OS-1 with the constraint in [Equation 7.5](#) relaxed to allow real valued a_k . This converts the MIP as presented into an LP which can be solved much more quickly in general. This particular LP relaxation can also be interpreted as representing a restoration scheme where primary paths would be arbitrarily decomposable for rerouting. This is usually not considered practical but conceptually, in ATM, it would correspond to letting individual VCs in a VP take different routes for restoration. The spare capacity achieved is a lower bound for the more practical case where only one backup path is available to restore each primary.

7.11 Illustrative Results and Discussion

For further insight and assessment of the oversubscription based design strategies, we draw on an illustrative set of test case results from [\[Zhen97\]](#).^[5] For purposes of comparing basic methods these results were produced with one unit of demand on each relation and the set of relations was all O-D pairs. In addition, the set of failure scenarios corresponded to every single-span failure, assuming a logical layer topology that was a clone of the physical layer, i.e., each link of the service layer corresponds to a single physical span.

^[5] [Sections 7.11](#) and [7.12](#) are based on results obtained in collaboration with Yong Zheng [\[Zhen97\]](#).

[Table 7-3](#) presents selected results summarizing the total spare capacity (normalized to total working capacity) and maximum oversubscription ratios arising in four test networks to which KST, OS-1, and the two bounding algorithms were applied. For the OS-1 designs the resulting $\max(X_{j,i})$ need not be listed as in all cases it was exactly equal to the design limiting X_{tol} value for the respective test case. The correctness of KST and OS-1 implementations was checked by independent programs that tested all possible failures and restoration rerouting experiments within each design, and functionally validated the adequacy (100% replacement) of the individual backup route and bandwidth assignments for all independent primary failures. It then calculated the experimentally observed $X_{j,i}$ values on every surviving link after each complete span and link failure scenario.

A striking feature of [Table 7-3](#) is how different the spare capacity results from OS-1 are from the capacity designs with *KST-Alg*. The *KST-Alg* results are however, quite consistent with the low levels of spare capacity for ATM backup VP restoration that were reported in [\[KaSa94\]](#). On the other hand, the OS-1 results are in much better general agreement with the results for path restoration (without stub release) in both [\[IrMa98\]](#) and [\[XiMa99\]](#), to which they are directly comparable in principle at $X_{tol}=1$. The explanation for the discrepancy between *KST-Alg* and OS-1 is explained by the oversubscription ratios $X_{j,i}$ values observed in the corresponding *KST-Alg* designs. Specifically, we see the uncontrolled effects of providing a backup path of equal bandwidth for each primary failure in isolation, but failing to coordinate the backup routes and capacity to take into account the set of backups that are simultaneously instantiated by a span cut. Thus *KST-Alg* designs will be functionally adequate against single random primary path failures but there can be uncontrolled oversubscription effects when many fail simultaneously in a common-cause scenario. This clearly accounts for why the spare capacity in [\[KaSa94\]](#) is so much lower than in either [\[IrMa98\]](#) or [\[XiMa99\]](#) and here. Note also that OS-1 results falls nicely between the two bounding algorithms which as a pair provide a relatively tight estimate.

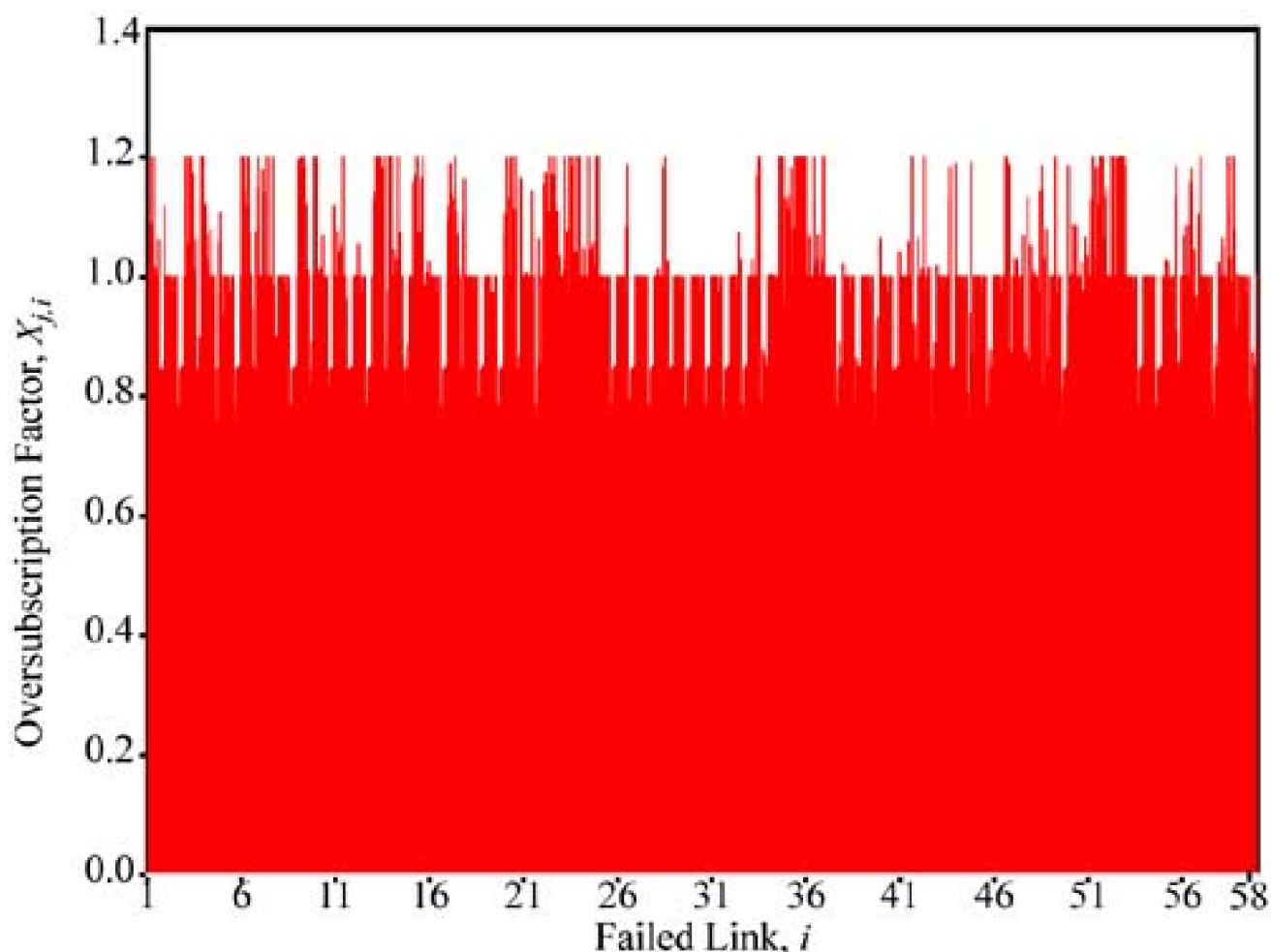
Table 7-3. Comparative Spare Capacity and Oversubscription Effects

Net	# nodes	# spans	KST Algorithm		OS-1			
			Spare	$\max(X_{j,i})$	Spare @ $X_{tol}=1$	upper bound	lower bound	Spare @ $X_{tol}=1.2$
1	15	28	51.7%	3.00	74.8%	78.4%	71.4%	68.0%
2	20	31	54.3%	3.32	82.5%	88.4%	76.9%	69.0%
3	30	59	31.2%	4.16	81.5%	86.9%	78.7%	65.9%
4	53	79	38.6%	10.0	91.9%	92.9%	91.4%	65.4%

[Figure 7-10](#) shows the oversubscription analysis in Net 3 (corresponding to [Figure 7-8](#)) using OS-1 with $X_{tol}=1.2$. The absolute clamp on $X_{j,i}$ values of 1.2 is quite evident, validating OS-1 for its intended properties. The OS-1 design at $X_{tol}=1.2$ has about 50% more spare capacity than the *KST-Alg* design but also 30% less spare capacity than the equivalent STM ($X_{tol}=1.0$) design. Note that in the $X_{j,i}$ line plot of [Figure 7-8](#), and to a greater extent in [Figure 7-10](#), there are many combinations of failure link i and other surviving link j where a link j actually experiences a *reduction* in total imposed traffic, i.e., an undersubscription effect in the protected-failure network state. In fact, in [Figure 7-10](#) undersubscription effects of ~ 20% appear about as often or even more so than oversubscription effects. The reason is the stub release effect of the failure. Restoration rerouting can only impose additional loads on network links, but the failure itself actually

lightens the load on many links because of the stub release effect. And because backup paths are disjoint from the primaries they protect, it tends to be links on the failed primary paths that are undersubscribed in the failure state.

Figure 7-10. Oversubscription factors with OS-1 at $X_{tol} = 1.2$ in net 3 (65.9% redundancy).



7.11.1 The Design Trade-off between Spare Capacity and X_{tol}

Using OS-1 it is possible to explore how the total spare capacity of the network responds to increasingly aggressive oversubscription strategies. Note that even in the most aggressive strategies, oversubscription effects remain only theoretical risks until such time as a failure arises. Even when a failure occurs, the worst case oversubscription factors will generally arise only for a certain failure span, on a particular surviving span, and even then only actually manifests itself as a corresponding overload if the traffic utilization of all primary paths is at its full value for the allocated working bandwidths. With this in mind [Table 7-4](#) summarizes design results (originally from [\[Zhen97\]](#)) for six test networks considering X_{tol} up to 2.0. $X_{tol} = 2.0$ is considered highly aggressive in the sense that if the oversubscription factor is applied to bandwidth allocations at full utilization, it implies a corresponding *traffic* overload factor of 2.0 which would generally cause severe loss and delay in queuing nodes. For comparative presentation, all spare capacity totals are normalized to that of $X_{tol}=1.0$ case for each network.

We see that the total spare capacity decreases rather quickly as the design tolerance for restoration induced oversubscription increases. Allowing only 10% maximum oversubscription, i.e., $X_{tol}= 1.1$, spare capacity is reduced by a range of 17% to 23%. At a more aggressive $X_{tol}= 1.5$, a full 60% to 70% reduction of the spare capacity is obtained.

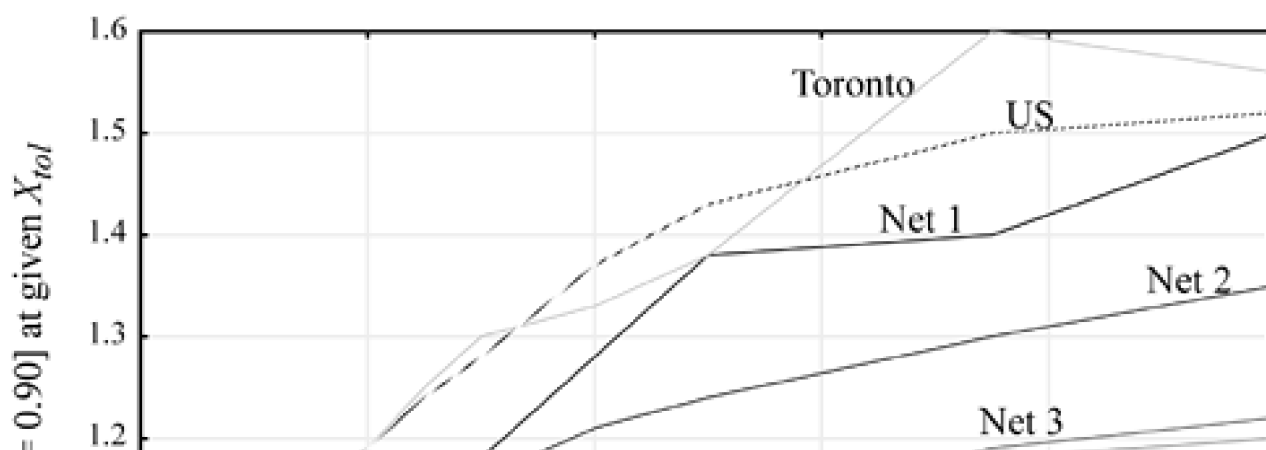
Table 7-4. Spare Capacity Requirement vs. Allowable Oversubscription

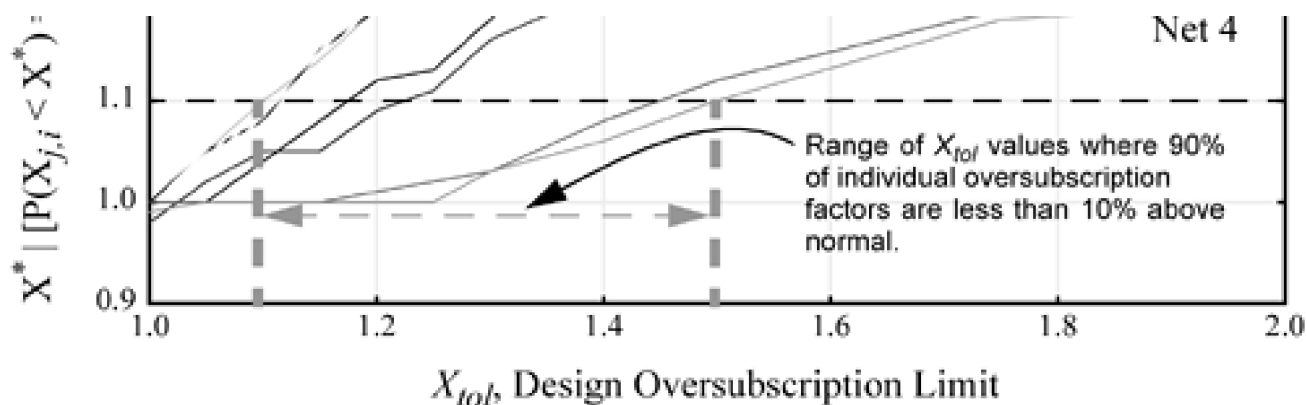
Design X_{tol}	Net 1	Net 2	Net 3	Net 4	Toronto	U.S.
1.00	100%	100%	100%	100%	100%	100%
1.05	91.5%	90.0%	90.0%	90.1%	87.5%	87.7%
1.10	82.9%	82.3%	80.6%	81.1%	76.3%	79.0%
1.15	75.0%	75.5%	76.9%	72.7%	66.7%	65.9%
1.20	68.0%	69.0%	65.9%	65.4%	57.3%	57.1%
1.25	62.0%	65.0%	59.3%	58.3%	50.4%	49.1%
1.30	57.3%	58.9%	53.6%	52.1%	40.5%	42.6%
1.40	48.6%	49.9%	43.7%	40.9%	30.4%	32.1%
1.50	40.9%	43.3%	34.9%	31.1%	23.0%	24.7%
1.75	26.5%	30.4%	22.7%	13.8%	7.1%	11.3%
2.00	17.3%	23.2%	14.2%	5.4%	1.9%	5.5%

7.11.2 Statistics of Individual $X_{j,i}$ Values under OS-1 Design

As pointed out, X_{tol} is the strict maximum oversubscription ratio that we will tolerate in any combination of circumstances in the OS-1 designs. This worst case where an individual $X_{j,i} = X_{tol}$ may possibly only occur for one specific combination of failure and surviving link in a whole network design. Thus, it is worth analyzing the statistics of actual values oversubscription levels experienced on links within a design produced with limiting value of X_{tol} . To do this [Figure 7-11](#) presents the 90th percentile of *actual* oversubscription levels experienced by links over all link cuts versus the design X_{tol} . The data shows, for example, that at $X_{tol} = 1.4$, 90% of the surviving links experience individual oversubscription factors no greater than 1.08, 1.06, 1.21 and 1.28, 1.33, 1.36 in Nets 3, 4, 2, 1, U.S. and Toronto respectively. Conversely we see that depending on the network characteristics we could have designs based on X_{tol} values from 1.1 to as high as 1.5 without having more than 10% of individual $X_{j,i}$ values exceed 10% oversubscription. This adds to the expectation that fairly significant capacity savings could be possible in practise without severe restoration induced side effects because most events will engender $X_{j,i}$ values that are well below the single limiting extreme chosen for the design of the network.

Figure 7-11. 90th percentile actual oversubscription levels versus tolerable design maximum.



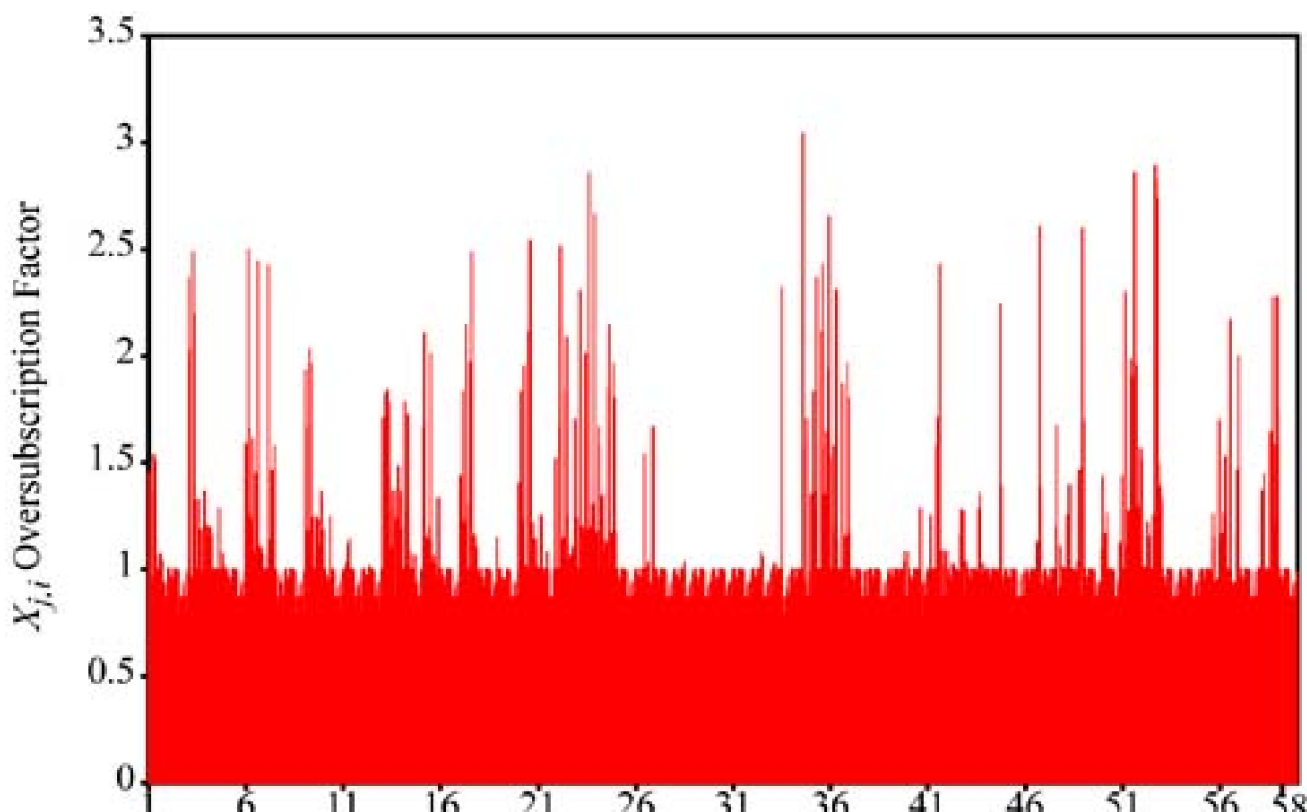


And even for those combinations of failure scenario and surviving link that do manifest $X_{j,i} \sim X_{tot}$ in the design based on path capacity allocations, we stress again that this only translates into the corresponding congestion effects if traffic was fully at the nominal utilization levels on each path during the failure.

7.11.3 OS-2 Minimization of Oversubscription with Given Capacity

Since *KST-Alg* allocates the least total spare capacity, it is of interest to see how the peak oversubscription might be capped within this spare capacity environment if OS-2 is applied to the *KST-Alg* spare capacity. Used in this way, OS-2 can be thought of as simply rearranging the backup path choices to improve their coordination against common-cause failure scenarios to reduce the peak oversubscription factor. [Figure 7-12](#) shows this result for Net 3. By rearranging the backup path assignments OS-2 managed in this case to reduce the peak oversubscription of the *KST-Alg* design to 3.04 from 4.16 (in [Figure 7-8](#)) while retaining 31% spare capacity. Not surprisingly a side effect of reducing the worst-case oversubscription is that there are more individual cases of $X_{j,i} > 1$. When we squeeze the maximum oversubscription down by applying OS-2, the restoration flows are distributed more extensively over all links and so more links suffer from a moderate oversubscription exposure, although the peak factor is lowered.

Figure 7-12. Oversubscription factors in OS-2 design with the spare capacity from *KST-Alg*.



Failed Link, i

7.11.4 OS-3 Benefit of Jointly Optimizing Primary and Backup Paths

OS-3 is a computationally ambitious formulation to solve to optimality due to the expansion of the solution space in terms of primary path choices and the need to consider a set of disjoint backups for each possible primary. Consider that if a network has D relations to serve, W possible primary routes on average for each relation, and B possible backup routes for each primary, then the solution space is $O(W^D B^D)$. Nonetheless, for research purposes results were obtained for three test networks, including the relatively large Net 4. The additional capacity savings attributable to joint optimization were from 3 to 9%. This is of interest as it indicates that when considering path protected design, there is little practical value to struggling with the jointly optimized approach. These findings are also consistent with those for failure-specific path-restorable networks where the benefit from joint capacity optimization, relative to spare-only optimization in path-restorable networks ([Section 6.8](#)), was found to be at most about 8%.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

7.12 Determining the Maximum Tolerable Oversubscription

In the previous sections, we saw that significant capacity efficiencies can be obtained if even a modest oversubscription factor is allowed in restoration spare capacity design. In choosing an X_{to} value as a guideline for network planning, the central issue is how oversubscription effects may translate into traffic-level performance degradations. In the worst case where all primaries are fully loaded, the oversubscription factor implies a corresponding cell or packet-level *overload* factor on a link. By "fully loaded" we do not mean that the primaries are at 100% utilization, only that their actual utilization is at the assumed level for the bandwidth allocation they were given. Notionally, *overload* is a function of *oversubscription*, and details of the cell or packet level flows themselves:

Equation 7.15

$$O_{j,i} \equiv f(X_{j,i}, U, T, P)$$

where:

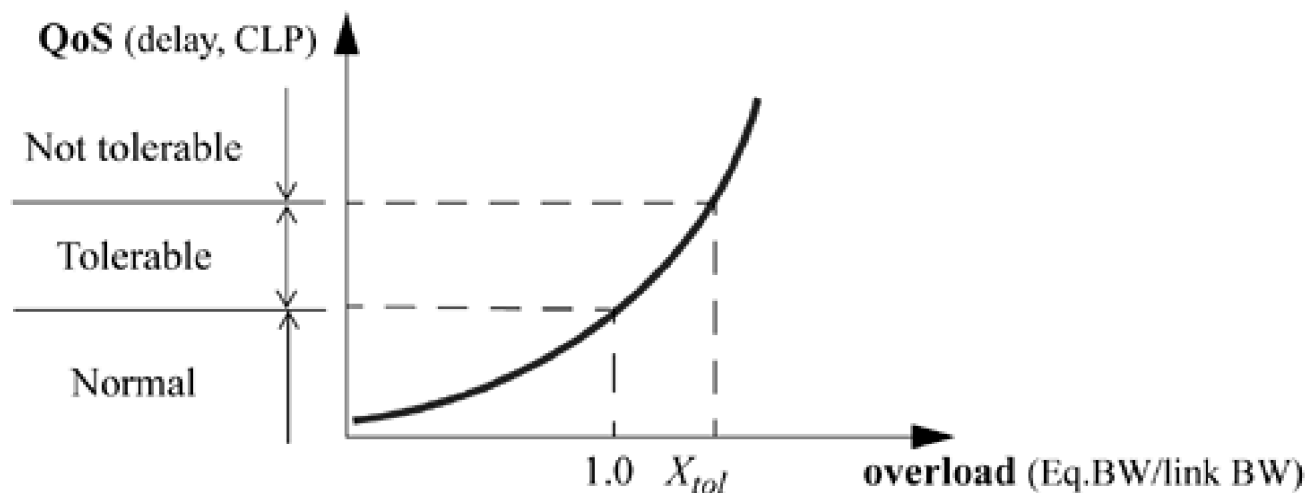
- $O_{j,i}$ is defined as the cell or packet level overload factor of a link in terms of equivalent bandwidth of the aggregated flow presented to the link in question.
- $X_{j,i}$ is the oversubscription factor of the link.
- U is the utilization of primary paths in the possible failure cases.
- T is a vector of traffic types, e.g., CBR, VBR, ABR, UBR in VPs or LSPs.
- P is a vector of traffic descriptors, e.g., rate, burstiness.

A basic problem in characterizing the effects of merging statistically multiplexed flows in MPLS or ATM is that there is a strictly infinite number of combinations of stochastic traffic descriptors and models to have to consider. One practical method for characterizing the overall load factor from a group of merged stochastic flows, all of different types and individual intensities is the concept of *Equivalent Bandwidth* developed in conjunction with extensive work on Call Admission Control (CAC) algorithms for ATM. (See for example [\[Onvu94\]](#), [\[GuAh91\]](#), [\[AnMi82\]](#) or [\[PeE196\]](#) for more detail.)

Equivalent bandwidth (EB) is defined as the effective bandwidth requirement of stochastic flows when statistically multiplexed into one link whose capacity is allocated so as to meet some specific requirement on QoS. In the CAC context, the network can decide to accept or reject a new flow connection based on its equivalent bandwidth and the currently available unallocated bandwidth. We can, however, adapt the use of equivalent bandwidth methods to approach the question of acceptable performance degradation in choosing an oversubscription design guideline.

To illustrate, consider a group of traffic flows that are aggregated into one link. Say that the combined equivalent bandwidth of the flows is 110 Mbps. This does not mean that the average bit rate is 110 Mbps, but rather that if we statistically multiplex the combined flows into a link whose capacity is 110Mbps (or more), the stipulated QoS goals on delay and cell loss probability (CLP) will be met. While the delay and loss response are generally nonlinear in response to congestion effects, the simple ratio of EB to actual link capacity provides a linear measure of *overload*, against which delay and CLP will at least respond monotonically. Thus, if the composite flow of 110 Mbps EB is imposed on a link of 100Mbps, we would say the overload factor is 1.1. In general, the performance degradation due to a certain overload factor can then be studied in terms of CLP or delay response to EB/link bandwidth as portrayed conceptually in [Figure 7-13](#).

Figure 7-13. How QoS measures respond to a restoration-induced overload factor.



By assigning limits to a delay or CLP increase that would just be tolerable in a restored-failure network state, one can then work backwards through studies to produce curves such as in [Figure 7-13](#) to assign a limit on tolerable (EB/link bandwidth) overload. Once the tolerable overload guideline is established, it can either be adopted directly as the tolerable limit on oversubscription, or, some higher X_{tol} value could then be set based on acceptable risk considerations that are not as quantifiable. The logic is that setting X_{tol} equal to the tolerable overload factor directly is rather conservative because it assumes that the worst case (j,i) failure combination will arise at the peak busy hour time of day and when the network is fully grown out.

In this framework, a series of cell level simulation studies was conducted to gain insight as to what might be a reasonable oversubscription factor. The approach was to find initial conditions with a given buffer depth and ensemble of simulated sources that just produced a cell loss probability (CLP) of 10^{-9} . In each simulation this established a nominal full utilization baseline. The EB of the source ensemble at these conditions is denoted $c-g$. Then in a variety of simulation trials, parameters of the source pool such as number of sources, utilization per source and peak rate per source were increased to find conditions at which a degradation to a CLP of 10^{-5} resulted. The EB of the source pool in this state is denoted $c-5$. The ratio $c-5 / c-g$ is then a measure, for the given traffic types and buffer depth assumed, of the *oversubscription ratio that would correspond to a worst case degradation of CLP to 10^{-5} during a restored state, if it was at 10^{-9} before restoration.*

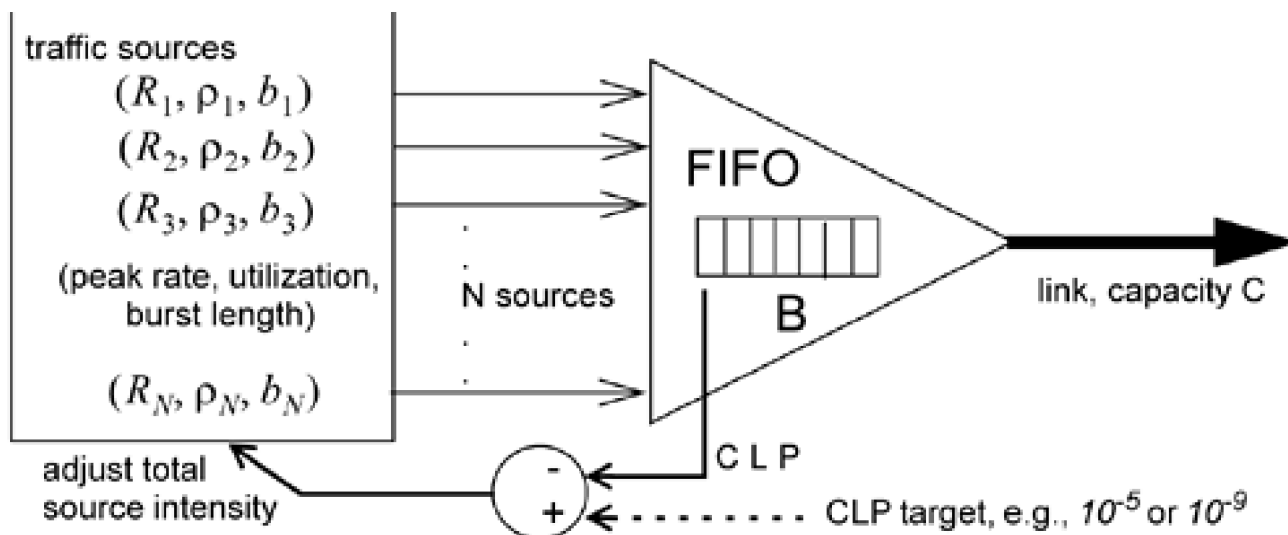
It is important to understand why the ratio is expressed as $c-5 / c-g$ (as opposed to the other way around). The key is to remember that EB is referred to the composite source equivalent load characteristics. Although it is calculated with a notion of asking "how much output bandwidth would be required to take this load at the given CLP?", the point is that when that question is answered it serves as a descriptor of the composite group of source as an equivalent *input* to the actual link present. So a group of sources that as a whole produce a CLP = 10^{-5} (i.e., $c-5$) are more (not less) bandwidth intensive than a group of sources characterized by g . Thus $c-5 / c-g$ is also always greater than one on any given link of fixed actual capacity.

7.12.1 Simulation Design for Equivalent Bandwidth

The system model used in simulation consists of a group of traffic sources, a finite queue and a transmission link, as illustrated in [Figure 7-14](#). Cells arrive asynchronously to the queue from the sources. They are multiplexed on a FIFO basis and transmitted out onto the link. In this simulation, an event-driven model is used in which the simulation of the queue is not driven by individual cell arrival events. Rather it is structured so that an event is any state transition in a traffic source model. The transition can either move from idle to active or from active to idle. Between any two adjacent events, the state of all traffic sources is unchanged. During these events the aggregated traffic rate of all sources is constant. A series of transition events is generated for all sources over the whole simulation time. The length of active and idle periods conforms to the respective Poisson process mean values. After verifying all transition events and traffic in all periods, we can calculate the buffer fill, total traffic throughput, cell losses, and hence CLP from data counts accumulated during the simulation run.

Figure 7-14. Simulation queuing model for determining X_{tol} guidelines.





Besides the parameters of the traffic model, the following factors are also involved in the equivalent bandwidth calculation:

- c , the link bandwidth.
- B , size of the buffer.
- The cell loss ratio (CLR).

To illustrate the event-driven simulation process consider N traffic sources with (R_i, r_i, b_i) $i = 1 \dots N$ being multiplexed into a link with a queue size B and link bandwidth c . In general, let Z be a random variable denoting the aggregate bit rate of all sources. Then, the dynamics of the queue in the system are defined as follows:

1. If $Z < c$ and
 - a. the buffer is empty, then it remains empty.
 - b. the buffer is not empty, then its content decreases at a constant rate of $c - Z$.
2. If $Z = c$, then the buffer content does not change.
3. If $Z > c$ and
 - a. the buffer is not full, then the buffer content increases at a constant rate of $Z - c$.
 - b. the buffer is full, then cells are lost at a constant rate of $Z - c$.

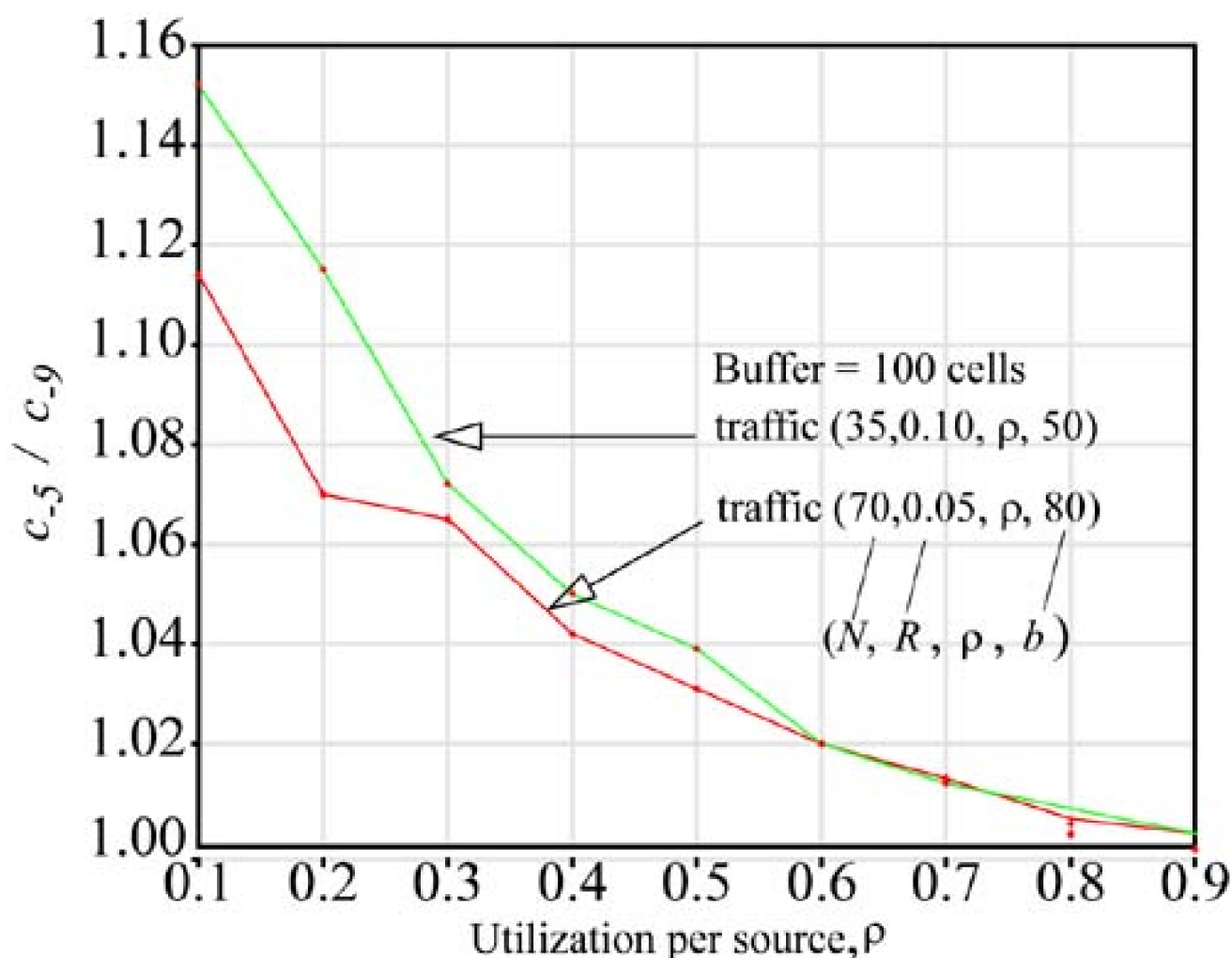
The Equivalent Bandwidth (EB) is defined as the minimum link bandwidth c , where the probability of cell loss is just less than the target Cell Loss Ratio. The observation window of the whole simulation is set large enough to hold at least 200,000 transitions of any source. It should be emphasized that in practice, the simulation can only obtain the CLR if a link capacity is given. The opposite is not feasible by running only one simulation (i.e., to obtain the equivalent bandwidth with a given CLR requirement). Consequently, to get the EB of a certain mix of sources for a given CLR, the simulation is run several times with differing link capacity values. The link capacity of the simulation that produced the CLR nearest to the target level is then regarded as the Equivalent Bandwidth of that particular mix of traffic sources.

Based on this approach we can obtain the EBs of the source composition for target CLR of 10^{-9} and 10^{-5} . We denote these c_{-9} and c_{-5} , respectively. The physical explanation is that a traffic volume c_{-5} is routed to a link whose nominal design capacity is c_{-9} . The ratio $O = c_{-5}/c_{-9}$ can then be taken as the tolerable oversubscription factor. Stated the other way around, $\overline{\kappa}_{tol}$ is set based on the c_{-5}/c_{-9} ratio for an assumed representative set of merging source traffic types, then it implies that under no circumstances would restoration result in CLR on a link degrading from 10^{-9} to more than 10^{-5} for the assumed traffic mix.

7.12.2 Illustrative Results

Figure 7-15 is a sample result from these studies of various source mixtures to determine what a reasonable design X_{to} value would be based on the overload factors that cause increases in CLP from 10^{-9} to 10^{-5} . Here overload and oversubscription factors are the same because we assume each source in the simulation represents the full load for some corresponding capacity allocation. Figure 7-15 shows the oversubscription ratio that leads to the above CLP increase in a buffer of 100 cells with an on/off fluid flow traffic model described by the values shown in the figure inset for number of sources N , peak rate per source R , utilization per source ρ , and mean burst length b , respectively. In this case the utilization per source is the simulation variable. From these studies, it appears that a capacity design with at least 5 to 10% oversubscription of restoration bandwidth would be feasible in many circumstances, under the above CLP increase criterion. Figure 7-15 shows that about 20% savings in spare capacity for restoration could thus be obtained.

Figure 7-15. Overload factors at which CLP degrades from 10^9 to 10^{-5} as a function of nominal primary path utilization for 35 and 70 merging primary paths.



With the increase of peak bit rate and burst length, the tolerable overload becomes larger. With the increase of utilization and buffer size, the overload factor decreases. The number of sources does not significantly affect the tolerable overload. From our analysis, and first principles, the tolerable overload also tends to decrease with a larger number of sources because overall traffic becomes more smoothed, offering less to be gained through further statistical multiplexing. It should be emphasized that this discussion only gives a general range for factors of tolerable overload. The practical value of a tolerable worst case overload (and hence design X_{to} value) for a particular network cannot always be determined exactly.

7.12.3 Other Comments on Determining X_{tol}

But the determination of a design X_{tol} value may also be in part a policy or business issue, as much as a matter of pure queuing-theoretic considerations. If there is to be strictly no degradation upon restoration, then $\max(X_{j,i}) = X_{tol} = 1.0$ must be the operator's policy, and the restoration capacity planning is equivalent to STM (i.e., perfect bandwidth replacement). But an alternate business point of view might be that all paths could be allowed to suffer slightly during a restored network state. In addition, the actual QoS impact also depends on the time of the failure relative to the busy period and the equipment provisioning interval, and this is not something the pure queuing simulations recognize. All things considered, a relatively high X_{tol} and hence low spare capacity allocation might actually be workable. In practice the aggressiveness of each network provider in designing MPLS restorable networks would vary in this regard. An aggressive startup network operator or one serving a great deal of best-efforts type IP data services, may want to allow large oversubscription to save valuable capacity in the network. On the other hand an operator positioning itself at the high quality service end of the market would strive to use the lowest oversubscription factors possible in capacity planning.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

7.13 Extension and Application to Multiple Classes of Service

Oversubscription-based capacity planning can also lend itself to providing differential levels of oversubscription for multiple service classes. In the treatment of oversubscription-based design to this point, we have inherently treated all primary path flows as being of equal importance or class of service. The simplest sense in which oversubscription-based capacity designs naturally accommodate multiple QoS in restored network situations is that some ATM and IP connection types automatically back off or suppress themselves if noticeable congestion is faced. ATM provides for *Unspecified Bit Rate* (UBR) and *Available Bit Rate* (ABR) traffic types, both of which conceivable tolerate a larger oversubscription factor because the protocols that support UBR/ABR traffic are designed to increase network utilization in normal times, and, if congestion is encountered, to back off on sourcing UBR/ABR traffic. UBR/ABR traffic sources are thus adaptive to the network load. If actual congestion does arise from entering an oversubscribed network state, UBR/ABR traffic sources will inherently give up bandwidth to the full priority services. In the IP / MPLS context, a large volume of ordinary Internet IP flows are under TCP control and will similarly back off if congestion arises from oversubscription. It is not necessarily desirable for all TCP / IP flows to be caused to back off in this way. Both TCP/IP and UBR/ABR transmission protocols therefore offer one natural form of "best-efforts" service support.

More structurally, however, the basic formulation (OS-1) includes a single scalar factor R^f that indicates how much of the primary bandwidth allocation for each relation we will even try to restore upon failure, and this can be directly the basis of a simple form of multi-service class design. One approach to multi-service capacity planning is therefore simply based on setting $R^f = 1$ for priority services and $R^f = 0$ for best-efforts services. In the corresponding network, primary paths associated with $R^f = 0$ would not be given any backup paths and would be prohibited from switching their primaries upon failure. If a significant fraction of working demands are of this best-efforts nature in OS-1 it will significantly lower the oversubscription effects without modifying OS-1 at all. In this approach, the oversubscription based capacity design effectively denies protection to best-efforts traffic if it fails, but allows non-failed working flows to persist on the non failure links, and hence still enter the oversubscription calculation in that sense (i.e., the first term in the numerator of [Equation 7.6](#)). $R^f = 0$ services can still obtain a meaningful best-efforts treatment, however, if at the time of failure, each surviving link assesses its actual cell level utilization following activation of backup paths for $R^f = 1$ services. End nodes of best-efforts primaries could then request (using ordinary CAC protocols) activation of backup paths for lower service class paths and these would be granted only if each link along the way agrees that its utilization is low enough to support the request even in the current restored state.

A further strategy would be to include an additional capacity "credit" term in the numerator of [Equation 7.6](#). The idea is to give an effective reduction in the W_j term, but only in the numerator of [Equation 7.6](#). The reduction is to dismiss or discounts any surviving working paths of $R^f = 0$ status in the oversubscription view of $\text{span } j$ following failure of another $\text{span } i$. The W_j in the denominator is *not* altered because $(W_j + S_j)$ in the denominator still represents the total installed capacity of link j . The revised oversubscription metric in the OS-1 model would then be written as follows where W_j in the numerator is replaced by the sum only of flows over link j that have priority status, i.e., $R^f=1$. This gives us:

Equation 7.16

$$X_{j,z} = \frac{\sum_{r \in D} d^r \zeta_j^r R^r - \sum_{i \in F(z)} \left\{ \sum_{r \in D} d^r \zeta_i^r \zeta_j^r + \sum_{r \in D} \sum_{k \in P^r} f_k^r \delta_{k,j}^r \zeta_i^r \right\}}{(W_j + S_j) \leftarrow \text{i.e. equivalently, total link capacity}}$$

where other terms are as previously defined. Assuming only $R^f=1$ or 0 for the moment the effect of this is to cause OS-1 to disregard any

surviving load from the low priority service when assessing oversubscription, in addition to only providing protection for the high priority class.

The operational meaning of this would be that the lower priority paths are actually *preemptible*. The best-efforts traffic neither obtains protection nor is assured of ongoing support on its working paths over surviving links where the bandwidth is needed for priority flows. Such a preemptible best-efforts philosophy is one in which flows of that class are inherently pushed out of the way when bandwidth is needed. This would allow even further reductions in spare capacity, while retaining assurances about oversubscription for all priority services, and corresponds inherently to the backoff behavior that will happen automatically if ABR or TCP control protocols encounter a bandwidth shortage, i.e., they inherently suppress themselves, vacating the scene when necessary, consistently allowing the $R^f = 1$ flows not to have to back off.

Another aspect of the OS-1 design model that could be used to support multi-service class design is X_{tol} itself. In OS-1, $X_{j,i}$ values pertain to each link as a whole, not to individual flows. We cannot therefore simply set a path-specific X_{tol} value to reflect the individual priority of paths. One could, however, group demands into more than one X_{tol} design class and solve instances of OS-1 to design separate logical capacity layers for each class of service. In practice, if three priority layers were defined, using three lightpaths on each logical edge between LSRs, the defining difference in X_{tol} /service classes would arise from the extent to which each layer could be loaded, since all the capacity allocations are the same. Far more load could be admitted to the $X_{tol} = 1.5$ layer than to the $X_{tol} = 1$ service class layer. [Table 7-5](#) offers a summary of these strategies.

The ideas above are partly inspired as an extension of the proposal in [\[YaKa98\]](#) which considers three service classes and provides more details of the signaling to seize and release spare and preemptible working capacities. The highest service class (excluding dedicated 1+1 APS realizations) is guaranteed its normal peak cell rate (PCR) in the restored state. In the framework above this is the $R^f = 1$, $X_{tol} = 1$ (i.e., "gold") service class. The middle service class is guaranteed its PCR if not itself failed (i.e., it won't be preempted) but is guaranteed only some minimum non-zero cell rate (MCR), less than its PCR, when it is restored. This corresponds roughly to silver where $MCR = PCR / (1+x)$ here. The lowest service class is only ever guaranteed an MCR, in either working or failed states, implying that a bandwidth allocation of $(PCR - MCR)$ of these paths can be preempted to assist in restoration of higher priority paths as required. This is like the preemptible service class in the scheme presented here, but with a form of MCR assurance more like the bronze scheme here. In general, there are many slightly different multi-service priority schemes that can be defined with the types of capacity design models we give above and implemented with signaling solutions such as in [\[YaKa98\]](#).

Table 7-5. Adapting the Oversubscription Design Model to Multiple Service Classes

Service Class	Quality of Protection (QoP) (service description)	Implementation
Platinum	1 + 1 physically reserved diverse capacity.	External to OS-1: Solve with shortest disjoint path pair and dedicate 100% backup bandwidth.
Gold	1:1 complete bandwidth replacement guaranteed for single path failures. Protection capacity is shared but designed for no oversubscription on span or link failure.	Use OS-1 with $R^f = 1$, $X_{tol} = 1$ to design the capacity for this service layer.
Silver_(x)	Service protected against any single link (or span) failure with guarantee of no more than x% worst-case oversubscription on any link in a restored network state.	Use OS-1 with $R^f = 1$, $X_{tol} = (1+x)$ Several virtual service class capacity layers can be defined for $x=0.5, 0.75$, etc.
Bronze_(x)	Non-protected service. Protection is not provided but working path bandwidth is also not preempted at any time and there is a guarantee of max oversubscription $< x$ %.	Use OS-1 with $R^f = 0$ within an otherwise normal OS-1 capacity design for $X_{tol} = 1+x$.
Preemptible	Non-protected, preemptible, service.	Use OS-1 with $R^f = 0$ and replace OS-1, Equation 7.6 , with Equation 7.16 to compute $X_{j,i}$ only over $R^f = 1$ service class flows.

7.13.1 Adaptive Link-Based Implementation of Priority Schemes

The benefit of the oversubscription-based capacity planning for 1:1 path protection is that it allows a network operator to determine an acceptable worst case "stress level" on cell or packet level performance during a restored network state and then design a network (or a network service layer) that exploits this tolerance for soft degradation to require the least possible total of spare capacity allocations. The multi-QoS schemes above make the overall efficiency of such networks even greater, but for the lowest two classes they assume outright suspension of service if any failure directly affects such paths, i.e., $R^f = 0$ means the path end-nodes do not even attempt restoration for those paths. In addition, the implementation of a silver service class with $(1+x)$ design X_{TO} that operates concurrently with a gold service class at $X_{TO}=1$ requires operational segregation of the capacity pools that each service layer operates within. If there is enough demand in each category to create good link utilizations, a separate lightpath or OC-n carrier may be justified to create physically separate logical links for each X_{TO} /layer and this remains quite efficient overall. If, however, the total demand on each layer is much lighter, such segregation implies extra queuing, interface, and software complexity for the routers or ATM switch nodes and a loss of stat-muxing efficiency and peak burst bandwidth available to any one application.

In the latter circumstances, one could still use OS-1 for capacity planning, but accompany it with a link-based adaptive scheme to differentiate between service classes within a single capacity pool, and without necessarily suppressing the lower service classes entirely. The OS-1 capacity and backup path design would produce a single capacity layer but using an X_{TO} calculation of the type in

[Equation 7.16](#) with $R^f = 0$ for the lowest two classes. Service discrimination below the gold level would then be redefined for this mode of operation to be a more continuous type of priority scheme. In particular, silver does not get its own separate X_{TO} guarantee but is redefined simply as a pure best-efforts category. The key ideas are that:

- Each working path and associated logical protection path bears a service classification. In either ATM or MPLS this could be implemented by partitioning the label space itself, or, at the time of service establishment, explicitly recording the service class of the path associated with the labels it is assigned at each node.
- At any time following a failure each surviving link would assess its actual cell level utilization after allowing enough time for backup path activation. This scheme is logically oriented around links monitoring themselves, but of course this service is implemented for them by the nodes on which they are incident. A network restored state could be explicitly notified or implicitly detected by each link from sensing the presence of flows on labels or VPIs that are known to correspond to backup paths.
- Based on the actual utilization (or other queuing measures) observed at each link following activation of all backup paths, each link begins "tagging" the paths traversing itself in a prioritized way that sustains the maximum the number of paths possible subject to the QoS guarantees given to the gold class primary paths and backup paths. If a path is "tagged" by any link along itself, its end-nodes are instructed to throttle it.

The link-state-driven control logic is:

[\[View full width\]](#)

```
Define Boolean QoS_ok := f(link_utilization, gold_primary+backups);
Boolean QoS_exceeds := f(link_utilization, gold_primary+backups);
Define Stack priority := {a stack of the labels of working paths and backup paths
  currently traversing the link};
Define stack killed := {a stack of labels of paths tagged};
```

```
While(not QoS_ok):
  {label_to_kill:=pop(priority);
  tag path (label_to_kill);
  observe and update (QoS_ok)}
```

```
While(QoS_exceeds):
  {label_to_allow:=pop(killed);
  admit_path (label_to_allow);
```

```
push(priority):= label_to_allow,  
observe and update (QoS_ok);
```

```
While (QoS_ok and not QoS_exceeds):  
  {observe and update (QoS_ok);  
  observe and update (QoS_exceeds)}.
```

`QoS_ok()` provides a true/false answer to the question: Is the link utilization low enough to ensure QoS guarantees for all gold primary and backup paths currently on this link? `QoS_exceeds()` is a related measure of whether conditions are actually significantly better than needed for gold QoS guarantees. The priority stack is maintained in the following order: {preemptible working service paths}, {activated backup paths for silver (best-efforts) services}. `Admit_path(label)` produces an explicit signaling message to the end nodes of a tagged path indicating it is readmissible from the given link's standpoint.

The overall effect is that in restored network situations where a link utilization is high, lower priority paths traversing the link are marked by a throttling indication to be acted upon by the path end-nodes and/or neighboring nodes as well. But the tagging only progresses up the priority scheme as required to ensure the gold class QoS guarantees from the standpoint of utilization on that link. Thus despite the number of paths traversing the link after restoration, all paths may enjoy a transparent continuation of service if *actual* loads in the restored state permit it. But if not, the priority paths are protected from a QoS reduction by the throttling of lower priority paths. In this way the benefits of design to exploit restoration induced oversubscription of capacity can be pursued with a protective mechanism to ensure QoS for selected services, while still granting all services a best-efforts treatment whenever actual circumstances permit.

The readmission aspect of the simple adaptive protocol above could be considered optional. Its main benefit is that it allows a way to compensate for possibly overaggressive tagging initially after the failure. The point is that paths are tagged locally by each link, not knowing immediately of path suppressions that will appear, relieving the local link conditions, as a result of tagging happening at other links. Once the tagging at other links has occurred, it may be apparent that the local link QoS has been greatly improved making it possible, at least in principle to undo some of the working path preemptions that may have occurred or to expand the extent of silver class restoration. A significant hysteresis or gap between the criteria for `QoS_ok` and `QoS_exceeds` is intended to avoid hunting back and forth between tagging and readmitting paths, stabilizing the system other than for ongoing updates to track normal call admission activities while in the restored state. Readmission of a tagged path by the end nodes does not occur, however, unless a separate ordinary call admission control (CAC) request succeeds end-to-end on the requested path. This is necessary because the path may have been tagged by one link, causing its suppression before it would have been tagged by other links as well.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

Chapter 8. Dual Failures, Nodal Bypass and Common Duct Effects on Design and Availability

When we say a transport network is "100% restorable" we generally mean that it has a sufficient amount and distribution of spare capacity so that any *single* span failure can be withstood without service outage. It may be tempting to think that once we have taken measures to enhance physical protection, designed a spare capacity plan, and embedded a real-time restoration or protection mechanism that the network availability must be just about perfect at that point. In principle only dual failures can then cause any service outage. In fact, measures to ensure single failure restorability *are* enormously effective in that they routinely convert "three nines" of physical unprotected availability (99.9% or ~9 hours of outage/year) into the region of "five nines" or better (99.999% or 5 minutes/year). But there are increasing numbers of "mission critical" services calling for as little as 30 seconds of unavailability per year. This may actually require the ability to withstand certain types of dual failures. At the least it motivates analysis of the effects of dual failures on single failure restorable designs. An availability figure is also only a long-term average characterization of a large pool of similar service paths. Regardless of how low the availability, it is still possible for any one dual failure to have enormous impacts on one user. It is more important than might be thought at first, to therefore ask what happens in the event of dual failures in a mesh-restorable network?

In fact there are several variations on the kinds of logical or physical dual failures that can arise. A dual failure situation is the most likely class of failure state after a single failure, and is therefore the most relevant set of considerations affecting service availability. But there are at least three different types of dual failure states. It is possible that before physical repair is completed on a first occurring (single) failure some other span fails independently, putting the network in a dual failure state. But where an express route or "bypass" logical span is co-routed on the same cable as other logical spans that terminate at the bypassed node, a single cable cut can have the effect of dual logical span failures. Another situation is where two logically distinct spans are actually routed through a common duct or other physical structure for part of their travel, called a "shared risk link group" (SRLG) situation. This chapter treats all these forms of physical or logical dual failure considerations in mesh networks based on span restoration but with some comparative data on the dual failure unavailability of SBPP. Finally, certain types of maintenance or in-service upgrade actions on one span may have network-wide effects that are similar to a failure-related use of spare capacity, so that if a true failure occurs while in the maintenance state, there may be a loss of restorability. However, that aspect of dual failure design considerations was already covered in [Section 5.11.2](#).

8.1 Are Dual Failures a Real Concern?

It seems initially that dual failures might be a rather academic concern. How likely is that in practice? Perhaps surprisingly it is a practical issue of importance. Would we, however, ever really expect to design for complete dual failure restorability? Probably not. We will see that this is indeed expensive in capacity, not to mention that it would require all nodes to be at least of degree three. Nonetheless there is quite a bit we know and can do to understand and mitigate the impact of dual failure situations. An example is understanding how typically 20% or more of all demands could actually receive dual failure restorability, even though the network as a whole is designed only for single failure restorability. This kind of advance opens the door to actually providing premium services with under 30 seconds annual unavailability, in an economic way.

But let us consider several reasons why dual failure issues are in fact of considerable importance to network operators and customers in need of high availability service. Each of these are points to which we return in further depth:

- 1. Sheer mileage and independent failure rates:** In some networks the total of fiber cable kilometers and the rate of failures on a per-km basis can, through basic statistical consideration alone, imply that the several dual failure situations will arise per year. With 30 to 50 thousand route miles of cable in a regional or long-haul carrier network, cable cuts turn out in practice, to be an almost weekly affair. With an average of one cable cut every four days [\[Fons98\]](#) and an 8- to 12-hour MTTR, it is clear that one can expect to face several time-overlapping independent failure situations each year.
- 2. Express route nodal bypasses:** Each time an express route bypasses the cross-connect at a node, but shares the same physical cables as other paths which transit the cross-connect, a specific type of logical dual span failure situation is set up. This is explained further, with rather surprising implications for spare capacity design.
- 3. Common ducts or other SRLGs:** It is administratively complex and expensive to be certain of the physical layer diversity of spans as cables are newly laid each year joining previously placed cables in a limited system of trenches, ducts, tracksides and bridge crossings and so on. Although every effort is usually made to achieve a diverse physical realization of spans, SRLGs do arise, causing dual span failures from single physical cuts.
- 4. Availability guarantees (SLAs):** Customers increasingly want service level agreements (SLAs) stipulating availability performance. To get the most out of the investment in single failure restorability an operator therefore needs to know how dual failures, which become virtually the only remaining outage contributor, affect the SLA guarantees that can be given.

8.2 Dual Failure Restorability Analysis of Span-Restorable Networks

Let us start with a basic question of analysis related to dual failures.^[1] The question is: How well does a mesh network, designed for 100% restorability to single failures, actually stand up to *dual* span failure situations? We consider this for span-restorable networks, but later extend the analysis to SBPP. The question requires us to define the *dual failure restorability* of a network. When we have completed a definition and appreciation of the dual failure restorability, we then develop the link between path availability and dual failure restorability. Later sections go on to consider how the design process itself can be altered to enhance the dual failure restorability, without necessarily adding any significant amounts of extra capacity.

^[1] This section includes adaptations of previously published material [\[ClGr00\]](#) [\[ClGr02\]](#) arising from the collaboration with M. Clouqueur.

We have previously defined the restorability of a network as the average fraction of failed working span capacity that can be restored by a specified mechanism within the spare capacity that is provided in a network. The average is taken over some stipulated set of failure scenarios, most often the set of all complete single-span failures. The restorability of a specific failure scenario has a corresponding definition. For each single failure scenario i , some w_i units of working capacity are disrupted. It makes sense to now distinguish explicitly between single and dual failure restorability statements. We therefore denote the fraction of w_i that is restored under single failures as the " R_1 " restorability of the individual failure scenario, specifically denoting it $R_1(i)$. The corresponding restorability of the network as a whole to single failures is denoted R_1 . We can thus similarly define the dual failure restorability of a network and individual spans as follows. Subsequently, unless otherwise mentioned "[restorability](#)" continues to refer to R_1 .

The "dual span failure restorability" $R_2(i,j)$ of a given pair (i,j) of span failures is defined as the fraction of the total failed working capacity of spans i and j that can be restored in the case of the simultaneous outstanding failure states on spans i and j . If $N(i,j)$ is the total number of *non*-restorable working channels under the failure of spans i and j , bearing w_i and w_j working links respectively (and $w_i + w_j$ is not zero), then:

Equation 8.1

$$R_2(i,j) = 1 - \frac{N(i,j)}{w_i + w_j} = 1 - \frac{N_i + N_j}{w_i + w_j}$$

where N_i and N_j are the individual numbers of unrestorable channels on spans i and j , respectively. Once we have the $R_2(i,j)$ values of each individual (i,j) failure combination there are two ways that average dual-failure restorability can be defined for a network as a whole R_2 . One is to define R_2 simply as the average of all $R_2(i,j)$ over all ordered (i,j) dual failure combinations. This leads to:

Equation 8.2

$$R_2 = \frac{\sum_{(i,j) \in S^2 | i \neq j} R_2(i,j)}{|S| \cdot (|S| - 1)}$$

where $|S|$ is the number of spans in the network.

However, this is an average of $R_2(i,j)$ ratios which gives as much importance in to a dual failure with $w_i + w_j = 1$ as it does to a dual failure involving spans with hundreds of working channels. A weighted R_2 measure that is more reflective of the overall demand-impact from dual failures can be stated as:

Equation 8.3

$$R_2 \equiv \frac{\sum_{(i,j) \in S^2 | i \neq j} (w_i + w_j) \cdot R_2(i,j)}{\sum_{(i,j) \in S^2 | i \neq j} (w_i + w_j)}$$

which, following substitution [Equation 8.1](#) for the basic definition of $R_2(i,j)$, simplifies to

Equation 8.4

$$R_2 \equiv 1 - \left(\frac{\sum_{(i,j) \in S^2 | i \neq j} N(i,j)}{\sum_{(i,j) \in S^2 | i \neq j} (w_i + w_j)} \right).$$

[Equation 8.4](#) winds up being in line with intuition: It says the dual failure restorability is unity less the fraction of total failed working capacity over all failure scenarios (dominator) that is not restored (the numerator). A further simplification is possible, however, by recognizing that the sum of all ordered " $w_i + w_j$ " pairs, excluding the duplicate pairs (on the denominator of [Equation 8.4](#)) is equivalent to $2(|S|-1)$ times the sum of all w_i values, because each span's w_i is added in the total for the $|S|-1$ times that it is not one of the failure spans itself, and because each (i,j) (j,i) ordering is counted separately. Thus, we get:

Equation 8.5

$$R_2 = 1 - \left(\frac{\sum_{(i,j) \in S^2 | i \neq j} N(i,j)}{2 \cdot (|S| - 1) \cdot \sum_{i \in S} w_i} \right).$$

8.2.1 Canonical Dual Failure Scenarios

When considering a span-restorable mesh network there are four logical categories that describe dual failure combinations:

- Failures that are spatially independent (the restoration route sets are disjoint).
- Failures with individual restoration route sets that interact with each other on the graph and may contend for needed spare capacity.

- c. Cases where the second failure damages one or more of the restoration paths of the first failure.
- d. Cases where the two failures isolate a degree 2 node or effect a cut of the graph.

Figure 8-1 illustrates the first three categories. In these illustrations, only the reserve network is shown and working capacities are indicated only for the failed spans. The bold lines show the restoration paths formed to restore the indicated number of working channels. Where the order of occurrence is relevant to the discussion, the first failure to occur is associated with w_1 . Figure 8-1(a) shows a case with no spatial interaction between the individual restoration route sets. Both failures therefore remain individually fully restorable, so $R_2(i,j) = 1$ for the scenario. Whether the restoration route sets interact or not depends not only on the failure spans involved but also on the rerouting mechanism and the working and spare capacity of the graph. Figure 8-1(b) portrays a case in which there is a spatial interaction between the routes of the respective restoration path sets. Depending on the spare capacities outage may or may not result from contention between the two failures. As drawn in the example, after failure and restoration of w_1 , only one restoration route remains feasible for the second failed span. Consequently, if $w_2 > 1$, then the second failed span will not be fully restored. Moreover, if $w_2 = 2$ we would say that for the particular ordered pair of failures (i,j) the restorability on span 2 is $1/2$ and over the two failures together $R_2(i,j)$ for the scenario is $3/4$. In general if the order of the failure pair had been different, the restorability for each span and possibly for the scenario as a whole can differ. This is why each dual span pair is considered as two (assumed equally likely) ordered sequences, (i,j) and (j,i) , in computing R_2 for the network as a whole with Equation 8.2.

Figure 8-1. The three canonical cases of dual failure in a span-restorable network ([CIGr02]).

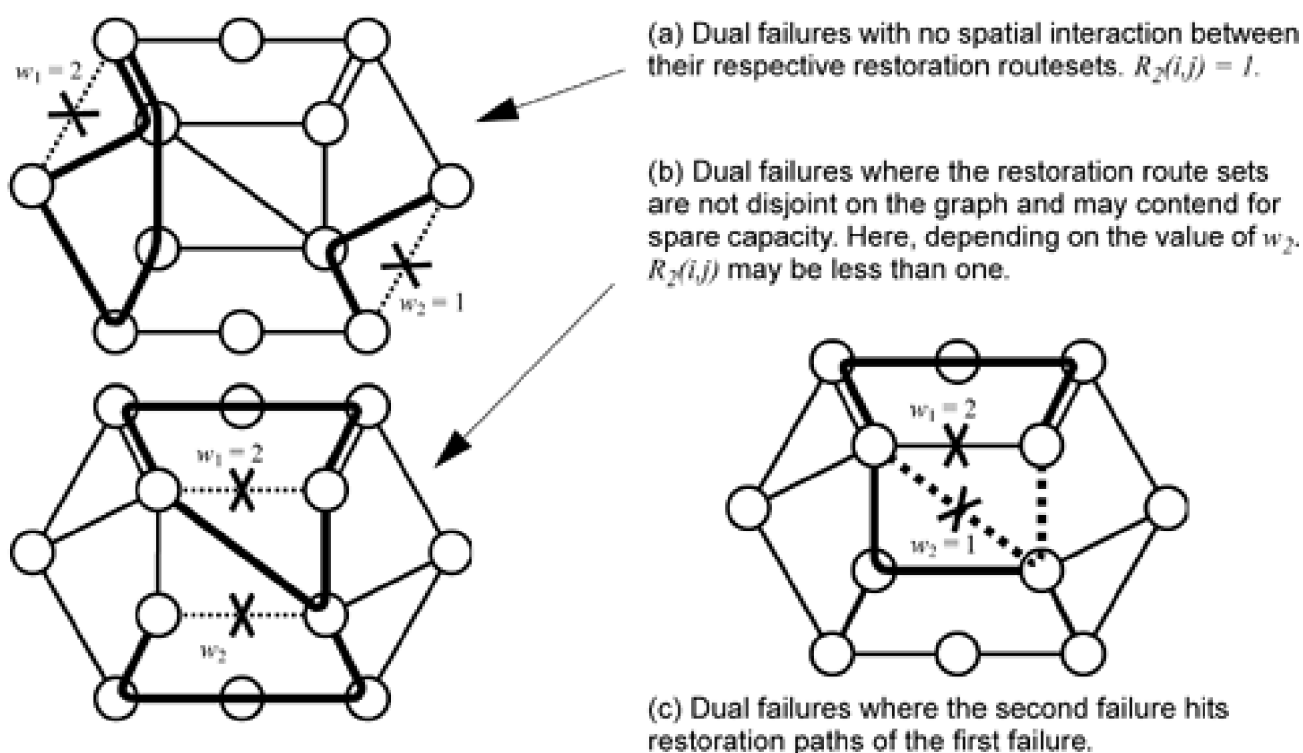


Figure 8-1(c) is a case where the second failure directly damages restoration paths deployed in response to the first failure. In this case the number of restored channels for the second failure depends on the remaining spare capacity after the first failure and on the "secondary" adaptability of the restoration process, i.e., whether the restoration mechanism is capable of (or allowed to) act again, when a previously deployed restoration path is severed by a second failure. The result also depends on whether sufficient spare capacity remains for the mechanism to repair the damage to its first path set by an updated restoration response.

Finally, (but not illustrated) a degree 2 node may be isolated by the failure of both its adjacent spans, creating a fundamentally unrestorable situation for which $R_2(i,j) = 0$ under any restoration scheme or capacity plan. Somewhat more generally, if there is any cut of the network graph that contains only two edges, then there are two ordered failure pairs that disconnect the graph. In this case the amount of spare capacity and the adaptability of the restoration mechanism also have no influence on the restorability. $R_2(i,j) = 0$ for any such dual failure scenario because the graph is disconnected.

8.3 Determining the Network Average Dual Failure Restorability, R_2

8.3.1 Computational Approach

It may be appreciated from [Figure 8-1](#) that it is difficult to predict $R_2(i,j)$ analytically because the $R_2(i,j)$ of each (i,j) failure pair depends in detail on:

- The exact positions of i and j in the graph.
- On the failure sequence.
- On the exact working and spare capacities.
- On the complete graph topology.
- On the assumed restoration dynamics.

Some workers have attempted various average-case approximations or other simplifying assumptions with regard to these details to facilitate analytical models. Typically this involves assuming aggregate network-wide probabilities for transition to and from failure states so that Markov modeling can be applied (e.g., [\[CaNa97\]](#)), making a specific assumption of 1+1 APS to model the protection mechanism (e.g., [\[ArPa00\]](#)), or doing comparative availability analyses on small fully-regular networks where rerouting possibilities are limited. These approaches all lead to useful comparative results and insights, but do not take fully into account, in a detailed way, all specific aspects above including interactions between topology, capacity and rerouting mechanism. It is overwhelmingly difficult for a purely analytical approach to capture all such details. And yet it is actually not that difficult to obtain $R_2(i,j)$ by direct experiment with routing programs that model the restoration process and take all considerations of the type in [Figure 8-1](#) exactly into account. This is the approach we take, leading to a partly analytical and partly computational approach to the overall problem of availability analysis of mesh restorable networks. The method is practical to use but does not simplify out any of the important details of exact topology, capacity distribution, and restoration mechanism.

8.3.2 Models for the Restoration Process

When implementing our computer-experimental approach to R_2 computation we realize there are at least three different models, varying by the level of adaptability, that could describe the restoration process or mechanism. Each of the following functional models corresponds to different technical options for engineering the restoration mechanism. We now define these restoration process models and then look at some experimental test results showing how the statistics of individual $R_2(i,j)$ and overall R_2 vary with the adaptability of the restoration process.

(a) Static restoration preplans

The first model for restoration behavior is intended to represent the use of centrally computed single failure preplans. In this model,

restoration of each span failure in a dual failure sequence follows a predetermined plan, trying to restore both spans as if each was an isolated failure. If not enough spare capacity exists to support the superposition of both static preplans, restoration paths of the second failure are suppressed to conform to what is feasible within the spare capacity remaining after the first failure, without changing any routes of the restoration preplan for either failure. When this is necessary, restoration paths for the second preplan that cross one of the spans on which spare capacity is in shortage are deleted in order of the longest path first.

(b) First event adaptive

The second model of restoration dynamics assumes that after a first failed span has been restored (but repair is not complete), the restoration mechanism of any second failure is aware of, and adaptive to, the changes in available spare capacity resulting from the first failure. Moreover, if any restoration path for the first failed span is routed over the span that then fails second in the sequence, the restoration mechanism will combine the working requirements for the second span failure with the failed restoration paths of the first failure. New restoration paths are not sought between the end nodes of the first failed span. In other words, restoration paths directly failed by the second failure are referred into the second failure's w_j quantity and restoration for the combined w_j quantity proceeds intelligently within the surviving reserve network using only spare capacity not already used by the first failure, and between the custodial nodes of the second span failure. This particular model is an exact reflection of the characteristics obtained when the SHN protocol [Grov97] operates in its usual way a network that sustains two failures in succession.

(c) Fully adaptive behavior

The third model is of a completely adaptive restoration mechanism including both spare capacity awareness following a first failure and a coordinated re-restoration effort for the first span *from its original end-nodes* for any damage that the second failure causes to previously deployed restoration paths. In almost all cases the behavior is identical or equivalent to that of (b) above, except when a second failure directly hits restoration paths of the first failure. The difference then is that the fully adaptive restoration mechanism falls back to the original end nodes of the latter paths, and attempts to re-restore them from that starting point, rather than referring that secondary damage into the span restoration effort for the second failure. This includes a release of surviving spare capacity on restoration paths of the first failure that were severed by the second, and repetition of the first span's restoration effort for the newly outstanding unrestored capacity. This is done with awareness of the spare capacity now withdrawn by the second failure. In detail, the functional sequence (from onset of the second failure) is: (i) restoration effort for the second failure (in light of all spares used by the first failure) (ii) fallback of the damaged restoration paths and release of their spare capacity, (iii) re-restoration effort for failed restoration paths of the first failure in light of all spare now used and of the release in (ii), and (iv) a further attempt at any remaining restoration paths required for the second failure. This behavioral model could be achieved in practice through centralized control or if using a DRA such as the SHN, through partly centralized control in which the Sender node holding paths of the first failure that are directly hit by the second failure is told to release those paths and reattempt them only after the SHN response for the second failure has proceeded.

Each of these two-failure restoration behaviors can be precisely implemented with adaptations of the basic k -shortest paths (ksp) routing program ([MaGr94] or see Section 4.5). Regardless of implementation details, the performance of real networks faced with dual failures is not likely to be worse than (a) nor better than (c). In fact, (c) amounts to a procedural approximation of the optimal problem of two-commodity path restoration, i.e., PRR-1 from Chapter 6 for the specific case of two "O-D pairs" in the reserve network following both span failures.

8.3.3 Experimental Results

Using programs that implement each of the above restoration models, work in [ClGr00] [ClGr02] considered all possible ordered dual failure experiments in several test networks and recorded the number of non-restorable working channels $N(i,j)$ for each failure pair (i,j) . From this $R2(i,j)$ and $R2$ were calculated according to Equation 8.1 and 8.2. The test networks were capacitated with only the theoretical minimum of spare capacity for an assurance of $R_1 = 100\%$ in modular and non modular capacity environments. The non-modular test cases were generated with the SCA design formulation (Section 5.3.4) with complete sets of eligible restoration routes up to a hop limit of

$H = 5$, inclusive. The modular test case designs also use $H = 5$ and module sizes of $M = \{12, 24, 48\}$ capacity units with a corresponding economy of scale of $c_j^m \in \{30, 40, 50\}$ under the "modularity aware" MSCA model of [Section 5.6.1](#). Demand patterns were generated using the gravity-based demand model ([Section 1.5.5](#)). A summary of the test networks is given in [Table 8-1](#).

Table 8-1. Test Networks for R_2 Dual Failure Restorability Experiments

Network	Nodes	Spans	Standard Redundancy (integer)%	Standard Redundancy (modular)%	Number of weight-2 cuts
Bellcore	11	23	55.9	92.1	3
EuroNet	19	37	50.6	78.3	5
6n14s	6	14	44.1	59.4	0
11n20s1	11	20	91.6	105.8	0
11n20s2	11	20	47.5	73.2	0
12n18s1	12	18	99.8	112.1	0
12n20s1	12	20	104.4	130.1	3
12n24s1	12	24	48.4	89.6	0
15n28s1	15	28	58.1	84.7	2
16n26s1	16	26	78.8	83.0	0
22n41s1	22	41	53.5	73.0	2

Here, redundancy is the ratio of total spare capacity to the total working capacity used in shortest-path routing of the working demands, i.e., the "standard redundancy" ([Section 1.5.3](#)). Standard redundancy is used here, not because working path routes or capacities are different as they might be in jointly optimized designs, but because we want to compare modular and non-modular designs on a fair basis, recognizing all provisioned capacity in the modular designs that is not working capacity as a form of redundancy. The redundancy of the test cases is relevant because if the $R_2(i,j)$ trials were repeated with arbitrarily high redundancy, correspondingly higher R_2 levels would result. The spare capacity available in the "integer" test networks is stringent: each has only the minimum amount of spare channel capacity sufficient to yield $R_1 = 1$. There is, however, a significant redundancy increase associated with modular capacity placement. This is a realistic side effect of minimum cost modular capacity design, however, and is therefore a practical effect which is important to reflect in assessing the R_2 level of these R_1 -designed networks. The last column of [Table 8-1](#) records the number of degree 2 cuts that exist in the test network graphs. In all cases but one these corresponded to degree 2 nodes.

[Table 8-2](#) shows the R_2 results for each restoration process model. These results include the penalizing effects of any degree 2 node or graph disconnections, where they arise in the averages. Under integer capacity R_2 ranges by network from 0.60 to 0.80 for the fully static behavior, 0.61 to 0.82 for the partly adaptive behavior, and 0.62 to 0.83 for the fully adaptive behavior. Thus, with the non-modular minimum capacity networks there seems to be relatively little benefit to the more adaptive restoration behaviors. In the same networks with modular minimum cost capacities, however, the more adaptive behaviors show R_2 values that are about 10% higher on average compared to the static behavior.

What is most remarkable overall is that networks designed only to withstand single-span failures have such high inherent levels of dual failure restorability. Even with fully static preplans about 70% of failed capacity is restorable on average under double failure scenarios. And in the best combination of a fully adaptive restoration process working in a modular capacity environment, we see R_2 values that are 90% and over.

Table 8-2. Dual failure restorability of test networks depending on restoration process model and modular or integer capacity

Network	(a) Static preplans		(b) First failure adaptive		(c) Fully adaptive	
	integer	modular	integer	modular	integer	modular
Bellcore	0.67	0.79	0.72	0.89	0.73	0.91
EuroNet	0.76	0.81	0.80	0.92	0.81	0.94
6n14s	0.60	0.70	0.61	0.92	0.62	0.99
11n20s1	0.79	0.79	0.81	0.82	0.83	0.83
11n20s2	0.60	0.75	0.61	0.80	0.62	0.82
12n18s1	0.66	0.70	0.70	0.74	0.71	0.76
12n20s1	0.70	0.72	0.73	0.77	0.76	0.80
12n24s1	0.63	0.80	0.67	0.87	0.67	0.90
15n28s1	0.73	0.84	0.77	0.88	0.78	0.90
16n26s1	0.75	0.80	0.76	0.81	0.77	0.81
22n41s	0.80	0.86	0.82	0.90	0.83	0.91

Figure 8-2 and Figure 8-3 consider the EuroNet test case above (and portrayed in Figure 1-14(c)) as a sample for closer inspection of the effect of the restoration model on the distribution of individual $R_2(i,j)$ values. The histograms of Figure 8-2 apply for the more stringent in the non modular design and shows that even though the overall R_2 is not much higher on average with the fully adaptive than with the static behavior, the adaptive effects are seen to be active and significant in terms of raising $R_2(i,j)$ of the worst cases of individual cases of low $R_2(i,j)$ under the static model. Whereas 30% of dual failure scenarios had $R_2(i,j)$ levels of 50-60% under static restoration, only 5% are that low under fully adaptive restoration. In fact the data of Figure 8-2(b) imply that if all dual failures were equally likely, 95% of the time the network could support more than 60% dual failure restorability. In Figure 8-2 five entries in the 0-10% $R_2(i,j)$ bins are not shown but correspond to dual adjacent span cuts at the five degree 2 nodes that this network model contains. These $R_2(i,j)=0$ cases are included, however, in the averages of Table 8-2.

Figure 8-2. Histograms of individual $R_2(i,j)$ dual failure restorability (non-modular, Euronet).

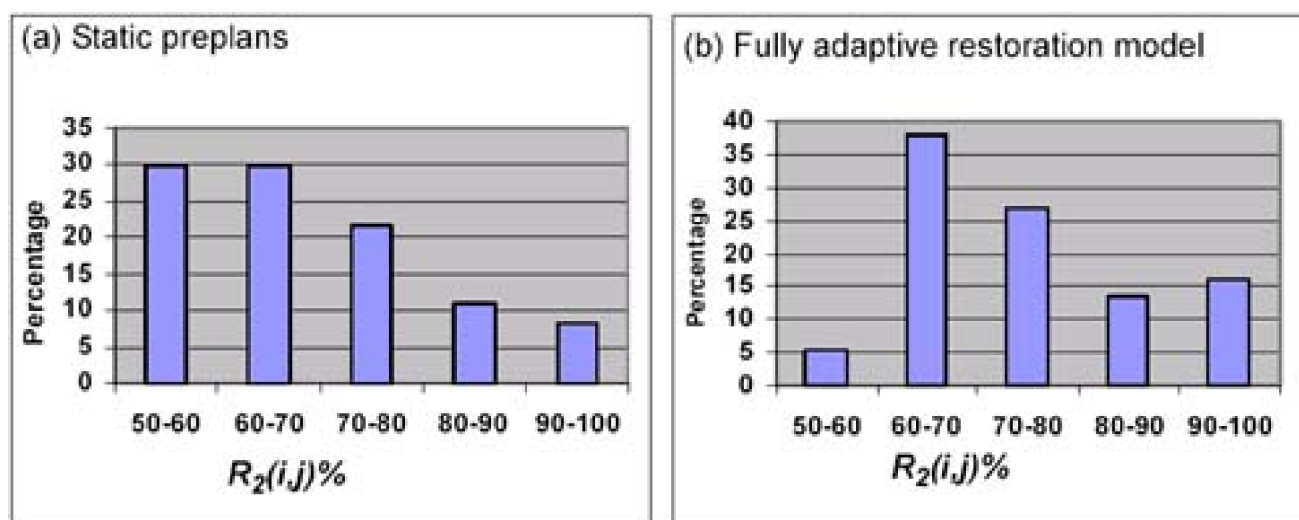


Figure 8-3. Scatterplots of network R_2 levels versus total network redundancy.

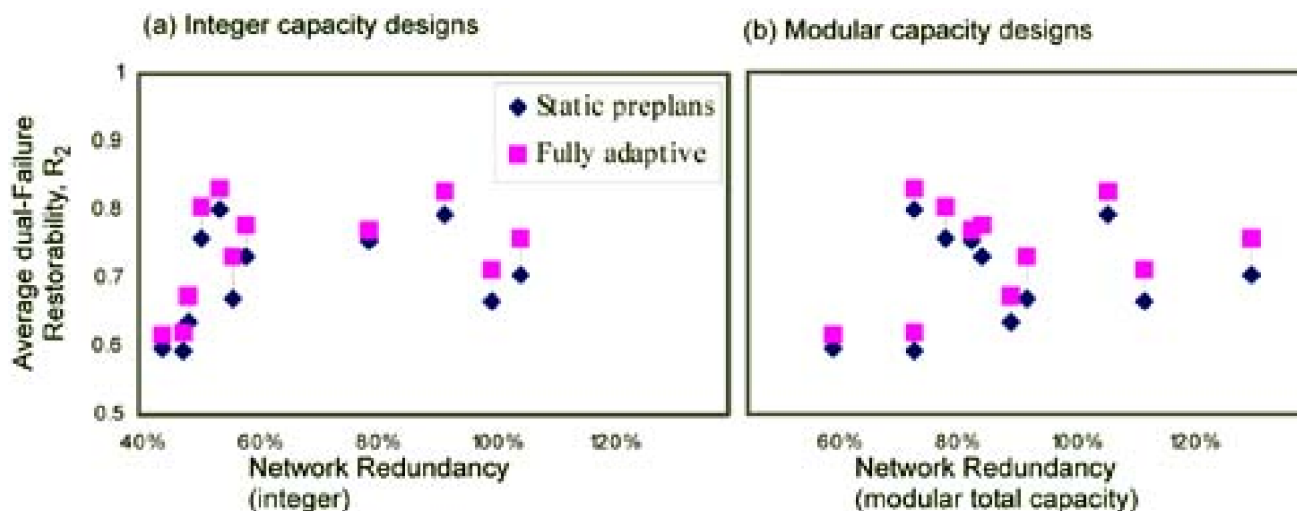


Figure 8-3 presents the R_2 level of each network plotted against its corresponding redundancy. Each network is represented by the R_2 levels corresponding to the extremes of fully static and fully adaptive restoration models on the y-axis. The purpose is to test the extent to which the R_2 level correlates with the relative spare capacity in the network. This is reasonable to check because it is a fairly widespread assumption that added redundancy by itself always corresponds to higher availability. While both plots in Figure 8-3 show a slight tendency toward greater R_2 with increasing redundancy, it is much less than might be expected, and by no means a systematic progression. The almost flat nature of the scatter means that R_2 (and hence the availability) actually depends much more on the individual network, demand pattern and recovery mechanism involved than on the simple bulk redundancy of the network. In other words, some networks can have high redundancy but still be less able to withstand dual failures than another network with a lower percentage of spare capacity. This is an important insight to keep in mind when comparing ring networks to mesh-based networks in general. In fact work completed just before the press time for this book [CIGr03] has found that although rings have the greater bulk redundancy, this does not automatically imply a higher availability because that redundancy is specifically arranged and locked up in rings. The greater flexibility of response in using shared spare capacity lets the mesh achieve the same or higher availability, even though using several times less spare capacity in total.

[Team LiB]

◀ PREVIOUS NEXT ▶

8.4 Relationship Between Dual Failure Restorability and Availability

In opening this chapter we made allusions which seem clear enough in principle that a higher dual failure restorability would contribute to higher availability in general. We now develop that linkage in a precise technical sense, and in doing so, we also show how the $R_2(i,j)$ characterization of network can be fed into the calculation of its service path availability using any number of different data or assumptions for the actual failure rates and repair times of network components. To do this we need to start with some background on availability analysis as it pertains to the problem of calculating availability of a path through a restorable network. [Chapter 3](#) included an introduction to the basic concepts of availability and reliability, and that remains required background, but now we extend those considerations to more specifically pertain to service path availability.

8.4.1 Axioms and Role of Availability Analysis of Networks

Recall that the most common equation for availability is $A = \text{MTTF} / (\text{MTTF} + \text{MTTR})$ where A is the availability, MTTF is the mean time to failure, and MTTR is the mean time to repair. Related to this is the *unavailability* $U = 1 - A$. In the telecommunications industry there is a long-used framework for approximate availability analysis of communication systems or signal paths, based on the following assumptions:

- a. A two-state "working/failed" model describes the status of all elements.
- b. Elements fail independently (aside from specific common-cause failure "scenarios" that may be identified for consideration).
- c. The in-service times (or times to failure) and repair times are each independent memoryless processes with constant average values.
- d. The repair rate is very much greater than the failure rate. Equivalently, the MTTR is much smaller than the MTTF.

It is worth discussing each of these to an extent because availability analysis is an important and useful technique.

Network or Path Availability?

Point (a) does not ignore that in general systems may degrade or only partially fail. Rather it requires that we make a clear definition of a threshold or other criterion that we agree constitutes the dividing line between "up" and "down" states. An important aside in this regard is that it is practically meaningless to talk about "*network* availability" per se, although the term is commonly used as a shorthand form for overall notions of characteristic or average path availability. The point is that a network as a whole is virtually *never* entirely available, nor entirely failed. By the strict definition of availability (all required system components up), network availability would always be nearly zero. Attempts to set a specific threshold at which a network as a whole is deemed to be either up or down (such as, say, 90% of all components working) is essentially arbitrary and still does not describe the availability of the *services* provided by the network. A user's perception of the network can be quite uncorrelated to such measures. 95% of components could be up (= "network up state") but a specific user's path is down. Conversely 20% of network components could be failed (= "network down state") but the same user's path is up. So conceptually the idea that we can have a single meaningful number for the "availability of a network" is problematic.

What is far more meaningful and practical to characterize is the *availability of specific paths* or the characteristics of specific sets of reference paths through a network. Although networks as a whole cannot really be defined as being either up or down, every specific path has a clear up/down state. It is either working at the prescribed capacity and bit error rates, or not. The signal is present and service is operating for the customer, or not. There is essentially no arbitrariness to this at all. The paths of interest to specific customers are their own. The paths of interest to network planners may be a specific set of hypothetical reference (digital) paths (HRDPs) on which various comparative technology selection or equipment specification studies can be done. An HRDP can be thought of as a "sentinel" path model.

It may or may not actually exist, but it is an agreed path model used for calculating availability. HRDPs may be defined for metro transport, metro access, long haul backbone, transoceanic, and other planning contexts. By focusing on the availability of paths (not whole networks per se) one also has the option in principle to calculate the availability of every possible path, and let the grand average, i.e., the network-average path availability stand to give meaning to the "network availability."

Independence of Failure Scenarios

Returning to the points above, point (b) touches on a sometimes misunderstood or oversimplified complaint about availability modeling. That is the argument that "it only deals with independent failures." This overlooks that in fact we do not disregard known correlated failure scenarios, such as when two cables share the same duct. We would include that as a single physical failure scenario whose occurrence is independent but whose impact is the simultaneous loss of both cables. This is the concept of failure scenarios as opposed to simple single element failures. We have to assume that the probability of each *scenario* is statistically independent but scenarios may include one or more simultaneously failed elements. Independence is only assumed between elements that are not obviously linked under a functional understanding of the system. In other words, only where we can see no scenario where they fail together due to a common cause do we model elements as subject only to independent failures. Thus while statistically independent dual span failures is one type of failure consideration, we do not assume that such failures can *only* arise through independent single failure mechanisms. The whole point of the SRLG concept is to recognize that certain failures are possible where a single event (that itself is assumed as being statistically independent) leads directly to correlated failure of more than one system element. Inclusion in a correlated failure scenario also does not exclude the element from having its own independent failures as well.

Statistical Equilibrium

Point (c) is strictly only true regarding failures over periods of statistical equilibrium. In practice, we know empirically that a cable cut is more likely during seasons and hours of work that correspond to greater construction activity, and so on. Nonetheless, the aim in availability analysis is usually to be able to gain comparative insight between alternatives, not to predict the future or the actual onset of failures or outages per se. For comparative studies of alternative service path designs and network technologies it suffices therefore to use a constant rate memoryless arrival process. The implication from the memoryless Poisson process assumption that repair times will be negative exponentially distributed is even less accurate in practice, primarily because any real repair has some clear minimum time before it can be completed but the Poisson model implies some repairs completed in almost zero time. Only the mean repair time really corresponds to practice. Nonetheless it is a property of calculations on mean measures (such as availability) that they do not depend on the distribution shape (here of repair times), only on the mean values themselves.

Adding Unavailabilities

Point (d) is abundantly evident from experience and it is the main reason that "network reliability" (in the sense of graph disconnection — [Section 3.13](#)) is not of greater practical concern. To illustrate, To and Neusy [[ToNe94](#)] give data for 100 miles of optical cable, the component of highest failure rate, for which MTTF = 19,000 hours and MTTR = 12 hours. The argument that MTTF >> MTTR is also the basis for a useful simplified form of mathematical treatment because it implies that:

Equation 8.6

$$\prod_{i=1}^n A_i \approx 1 - \sum_{i=1}^n U_i$$

where A_i is the availability of the i^{th} set of N elements in a series relationship, and U_i is the corresponding *unavailability*, $1-A_i$. This says we can "add unavailability instead of multiplying availabilities" for elements, or subsystems, in series. Freeman [Free96b] illustrates the accuracy of this approximation with an example of six elements in series having elemental unavailability from 10^{-5} to 10^{-3} . The accuracy of the approximation is better than 0.5% which "is typically far more precise than the estimates of element A_i 's". A simple way to appreciate this intuitively is to consider just two elements in series for which $A_{\text{exact}} = A \cdot A = (1-U)(1-U) \approx 1-2U-U^2$. We see that the approximation omits only the square of an already small number. With the data above, $U = 6.3 \times 10^{-4}$ so the error in ignoring terms of $O(U^2)$ is clearly trivial.

"Most Likely Paths to Failure" Concept

Another important concept for practical availability analyses, but not one of the axiomatic assumptions listed above, is the concept of the "most likely paths to failure" [WiSh97] as dominating the unavailability. For clarity in our context we could reword this as the "fewest steps to outage." The principle is that in any system with redundancy, the simplest combinations of elemental failures that lead to an outage state will dominate the overall availability. The system availability is approximated by summing effects of these dominant classes of failure combinations.

The direct application of this principle in our case is that dual failures need to be the focus of the availability analysis. Any single failure is fully restorable by design. Dual failures are therefore the next most likely situations to arise and are expected to almost fully determine the availability. This is especially true as individual element failure rates are small. The reasoning is that if there is a way for two failed elements to bring down the service, this is much more likely than a three-element scenario. The validity of this can be seen in a simple

argument for a network of, say, 32 spans. There are $\binom{32}{2} = 496$ double failure combinations. Although there are ten times as many triple failure combinations (4960), the U^3 term weighting such combinations means that, even if every triple failure was outage-causing, their total contribution would remain minor. As an example, again using the elemental unavailability value for the 100 miles of cable, $496 U^2 = 1.97 \times 10^{-4}$ while $4960 U^3 = 1.25 \times 10^{-6}$, which is ~160 times below the double failure contributions.

Philosophy of Availability Analysis

In summary, it is important to understand that availability models do *not* claim to be exact predictors of what will happen in the future. Availability analysis is more like a formalized system for comparative reasoning about the inherent quality of alternative system architectures. Examples of how it can guide the system design process are questions such as: All other factors being equal is it inherently better for a service path to rely on adaptive span restoration or 1+1 APS? The extent to which availability analysis applies to the real world is limited by the axiomatic assumptions above and practitioners need to be aware of this. Generally, however, the assumptions above are accepted for comparative analysis assuming a large number of system instances operating over a long time. Availability analysis does not predict what will be the experience with any *one* system instance over any specific time interval; it strictly characterizes only the whole population of identically designed systems operating in identical conditions over an indefinitely long time.

8.4.2 Average Case Availability of Service Paths

Let us now consider the availability of a service path through a span-restorable mesh network that is designed for 100% single failure restorability, i.e., $R_1=1$. Based on the arguments above, we expect dual failure combinations to dominate the unavailability. In addition a path through a network in general has an obvious series-element nature. A path through a span-restorable network has the same series-element structure and it also has a form of parallel redundancy too, representing the combination of spare capacity and an active restoration mechanism. The problem is, however, that it is unclear how to explicitly represent this parallelism in an availability block

diagram for a given path. In contrast to situations of clear series-parallel structure and explicit means for analysis (such as [Section 3.12.4](#)), the sense in which an adaptive restoration mechanism embedded in a pool of shared spare capacity represents parallel redundancy for a path is much less clear. One approach to this problem is, however, to model the path as a pure series element system, but where the individual element unavailabilities in the series model are somehow transformed to represent the actual net effect of the pooled and shared redundancy that is actually present in the network environment. This we do through the concept of an *equivalent channel unavailability*.

If we first imagine a completely passive and unprotected network over which a path p is provisioned over n spans, its exact availability expression would be:

Equation 8.7

$$A_p = \prod_{i=1, \dots, n} A_i$$

where A_i is the availability of the i^{th} span in the path. We know, however, that we can closely approximate this as:

Equation 8.8

$$A_p = 1 - \sum_{i=1, \dots, n} U_i$$

or equivalently:

Equation 8.9

$$U_p = \sum_{i=1, \dots, n} U_i^{\text{phy}}$$

where U_i^{phy} is the physical unavailability of the i^{th} span in the path.

One way of thinking about the path through a span-restorable mesh is that it is of the same form as [Equation 8.9](#), which is for the non-protected path, but that the action of span restoration is that it is a *transformer* of physical span unavailability U_i^{phy} to a lower equivalent unavailability U_i^* of channels on the span. The viewpoint is that from the standpoint of an end-to-end path, there are two equally acceptable ways in which a channel along the path can be in "up" state: either it is physically working, or it is physically "down" but transparently replaced or repaired by a restoration path. Thus, if we can define or compute the equivalent unavailability of a channel, then the path availability has the same form as [Equation 8.9](#) but is based on the summation of U_i^* values not U_i^{phy} , i.e.,

Equation 8.10

$$U_p = \sum_{i=1, \dots, n} U_i^*$$

This reduces the problem of calculating path availability to determining the equivalent unavailability of channels in a span-restorable network based on the capacity in the network and the particulars of the restoration mechanism.

Clearly, with no spare capacity or no restoration mechanism, there is a 1:1 conversion of physical unavailability to equivalent channel unavailability. But with a restoration mechanism and a given reserve network, the fraction of U_i^{phy} that comes through to U_i^* when span i fails depends on the states of other spans in the network at the same time. Not all working channels on span i may be restorable if there are failures outstanding on other spans. In principle, U_i^* also depends on the reconfiguration time for channels that are restorable, although in practice, outage due to reconfiguration time itself is milliseconds, at the most a few seconds, and is a very small effect compared to outage due to multiple failure states that are not fully restorable. (In the latter case, unrestored channels may expect to see an outage time of half the MTTR on average). It is now fairly easy to show how the dual failure restorability is involved in determining the equivalent channel unavailability and, hence, the path availability.

Consider a network of S spans, each with its own elemental probability of being in a failed but not yet repaired state, i.e. U_i^{phy} . The probability of being in a dual failure state in which spans i and j are both (independently) failed is thus $U_i^{phy} \cdot U_j^{phy}$. But the probability that a given channel that is disrupted under the i,j scenario and is restorable is exactly the previously defined $R_2(i,j)$. Therefore the probability that an individual channel (at random) on a span i is *not* available due to a independent dual span failure in the network is:

Equation 8.11

$$U_i^* = U_i^{phy} \cdot \sum_{j \in S | i \neq j} U_j^{phy} \cdot (1 - R_2(i,j)).$$

In the case where all spans have the same elemental physical unavailability U_s and we average over all dual failure scenarios, we get a single equivalent unavailability due to independent dual failures that simplifies to:

Equation 8.12

$$U^* = (S - 1) \cdot U_s^2 \cdot (1 - R_2)$$

where U^* is a single equivalent average unavailability for channels in the network.

Let us now use this simple result to illustrate how R_1 restorability affects availability. Consider a path comprised of 4 hops in a 20-span network having $R_1=1$ and an average dual failure restorability of $R_2 = 0.7$ (which is relatively conservative given the experimental results above). Also assume a physical span unavailability $U_s^{phy} = 3 \times 10^{-4}$, which would correspond to about 50 miles of cable given To and Neusy's data above. In a non-restorable network the equivalent channel unavailability is no different from U_s^{phy} so the unavailability of the path would be $U_p^{phy} = 4 \cdot U_s = 1.2 \times 10^{-3}$ or about 10.5 hours of outage per year. In comparison, with $R_1 = 1$ and $R_2 = 0.7$ the equivalent channel unavailability is

Equation 8.13

$$U^* = (3 \times 10^{-4})^2 \cdot (20 - 1) \cdot (1 - 0.7) = 5.13 \times 10^{-7}$$

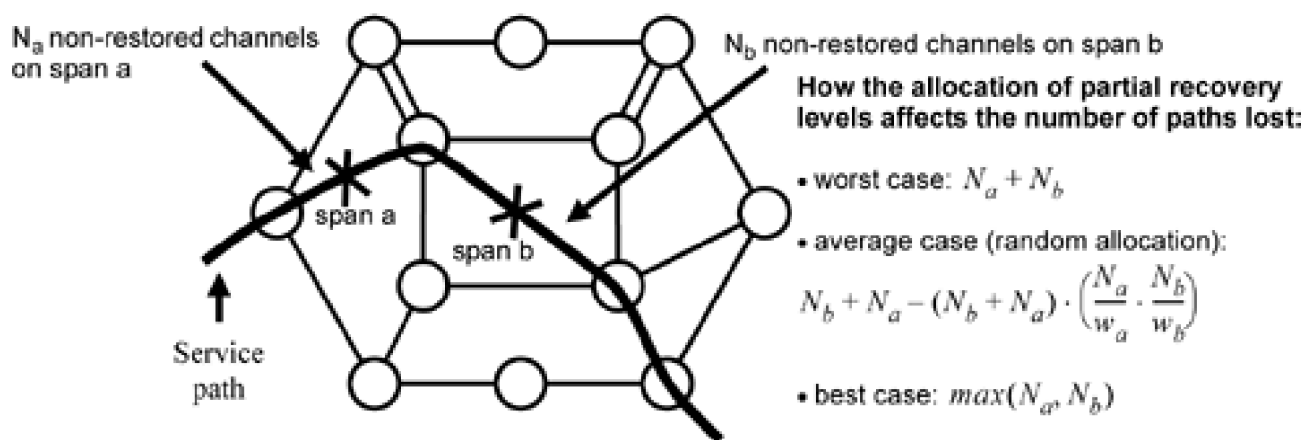
which implies a path unavailability of $U^* = 4 \cdot U = 2.1 \times 10^{-6}$ or about *one minute* of outage per year. Thus, in this example the improvement brought on by the investment to achieve $R_1 = 100\%$ is a reduction by approximately 585 times in the unavailability of the 4-hop path. An interesting thing to note at this point is how this compares to perfect 1+1 parallel redundancy as a conceptual reference. If the physical path is taken as the element and given dedicated 1+1 parallelism, the unavailability becomes $(U_p^{phys})^2 = 1.44 \times 10^{-6}$, which is only slightly better than the unavailability achieved by the mesh-survivability scheme with shared spare capacity. We see in later sections that if the fractional R_2 restorability levels are allocated to failed channels on a priority basis, that it is possible for selected paths through the mesh-restorable network to have availabilities *better than* that with dedicated 1+1 APS.

8.4.3 Availability Calculation for a Specific Path

The availability model so far developed is an average case model in the sense of how the $R_2(i,j)$ "asset" is used to benefit the availability. This is because it assumes that under any given failure scenario where $R_2(i,j) < 1$, that available restoration paths are allocated to failed channels without priority (i.e., effectively at random). Its use in [Equation 8.11](#) $R_2(i,j)$ is averaged over all possible dual failure scenarios to produce the R_2 that is used. [Equation 8.11](#) (or [8.12](#)), when used in [Equation 8.10](#), is thus an overall characterization of the expected availability of all paths on the route considered. This is consistent with the general intent of an availability analysis in that it speaks to the long-term average over a population of identical systems, in this case paths over a given route. In other words, it could be used to address a question such as "What is the availability to expect of the average path five hops long in this network?" But it does not say anything about the actual availability of any *one specific* individual path.

One of the first differences in considering a specific path rather than an average case path is that there may be situations where the unavailability of a path may strictly be overestimated by [Equation 8.11](#) or [8.12](#). This arises in the specific circumstances of an (i,j) failure scenario for a path that crosses both spans i and j , and both spans i and j have $R_2(i,j) < 1$. Under these circumstances, which only pertain when a specific path is stated, the end-to-end availability of the path will be slightly pessimistically estimated by [Equation 8.11](#) (or [8.12](#)) because it sums the individual channel unavailabilities on each span as independent contributors to the path outage. But if both failures of some scenarios leave unrestored channels that are both allocated to the path of interest, then in reality such a scenario contributes only once to the unavailability of the path. [Figure 8-4](#) shows how the independent channel unavailability contributions can in effect be counted twice in the unavailability of a specific service path if the allocation of restoration paths to failed channels is done in some coherent or coordinated way, rather than at random.

Figure 8-4. Example of how the sum of average case channel unavailability can overestimate the unavailability of a specific service path if the allocation of recovery resources is not random.



For simplicity of discussion, assume that there is only a partial recovery level on each span in [Figure 8-4](#) and that $\min(w_a, w_b) > N_a + N_b$ so that there would be enough continuous paths through spans a, b , to support the following arguments. If so, the fate of the through paths, in terms of how many suffer outage, depends on how the partial recovery resources are allocated or, equivalently, how the restoration shortages N_a, N_b are allocated. If it is done randomly, then in the worst case of "mismatch" allocation could result in $N_a + N_b$ paths suffering

outage (if shortages are allocated to disjoint sets of paths on each span). This worst case is actually what the general model assumes under [Equation 8.11](#) or [8.12](#). On average, however, exact analysis of the case for random allocation would predict $N_a + N_b - (N_a N_b / w_{ab})$ where the last term accounts for the expected number of paths that wind up being randomly allocated a shortage on *both* spans *a* and *b*. The best case would arise with an intelligent allocation of the restoration shortages in a way that each failed path absorbs a shortage from *each* span. Of course, if $N_a > N_b$ then only N_b paths can sacrificially absorb *two* shortages each, leaving the difference $N_a - N_b$ of paths that must each also absorb a shortage on span *a*. Overall, the number of paths suffering outage in this best case of coherent allocation is reduced to $\max(N_a, N_b)$.

We can make two useful observations from this: first, we realize that if restoration paths are allocated randomly, then the error involved in [Equation 8.11](#) or [8.12](#) is in the direction of slightly *overestimating* the path unavailability, so it is conservative in engineering terms. It will not usually be significant numerically however. Such error is after all only a manifestation of the approximation that we identified and accepted above in [Equation 8.6](#) that for a series system, the product of availabilities can be approximated by a sum of unavailabilities. In addition, of all dual failure combinations only those where both failures fall on the same path can possibly have this effect, and only if both of the spans in those scenarios also individually has less than a full restoration level. If either individually has full restoration then the overestimation does not occur. For dual span failures several spans apart on the same path this is likely often to be the case due to the spatial separation of their individual restoration patterns. Where the overestimation will be most likely is for adjacent span failures as shown in [Figure 8-4](#).

But secondly, these considerations tell us that it will be advantageous in practice to make a coordinated allocation of the limited restoration paths available for each failed span. The natural way to do this is through a priority scheme involving a class of premium service paths. In

fact it is interesting to observe that as long as $R_2(i,j) > 0$ on both spans, some number of the top priority paths could effectively see $U_i^* = 0$! Any time $R_2(i,j) > 0$ it means that there is at least one restoration path available for span *i* in the (i,j) dual failure scenario. Thus, if we think of a single topmost priority path crossing span *i* (but not span *j* for the moment) which is always preferentially allocated the available restoration path, then that path is not only immune to any single failure on span *i* (by virtue of $R_1(i) = 1$) but it is also immune to the (i,j) dual failure scenario by virtue of $R_2(i,j) > 0$. More generally, if $R_2(i,j) > 0$ for all other spans that the path crosses (where *j* is not on the path) then the path is also immune to all dual failures not involving dual cuts on the path itself. So it follows that if for all (i,j) where *i* and *j* are both on the path and $R_2(i,j) > 0$, $R_2(i,j) > 0$ then the specific top priority path we have considered is *assured of restorability against all dual failure combinations!*

Put another way, we can take the view that the availability of the individual priority path is still estimated through [Equation 8.11](#), but that

under priority allocation there is a null contribution to U_i^* anywhere $R_2(i,j) > 0$ because if there is even only one restoration path, it will be allocated to this priority path. In effect the specific path enjoys an *effective* $R_2(i,j) = 1$ for its needed any time the overall $R_2(i,j) > 0$. Thus, in the context of a specific path whose route and priority are known, the path availability can be more accurately estimated as:

Equation 8.14

$$U_p^* = \sum_{i \in \{p\}} U_i^{phy} \cdot \left\{ \sum_{j \in S \setminus \{p\}} U_j^{phy} \cdot (1 - \delta_i^p(i,j)) + \sum_{j \in \{p\} \setminus \{i\}} U_j^{phy} \cdot 1 - ([\delta_i^p(i,j) \cap \delta_j^p(i,j)]) \right\}$$

where:

- $\delta_i^p(i,j) = 1$ if the *individual* channel on span *i* that carries the path *p* was allocated a restoration path on span *i* from the set of restoration paths that were feasible under the (i,j) dual failure scenario, zero otherwise.
- $\{p\}$ is the set of spans in the specific path *p* of interest.

[Equation 8.14](#) can be thought of as walking along the length of the specific path of interest, considering each span *i* on the path as failing following the prior failure of: (i) in term 1: each other span that is not on the path, and (ii) in term 2: each other span that is on the path. In the first category of failure scenarios, path *p* survives if it has the priority to warrant allocation of one of the restoration paths available for

span i under the particular dual failure scenario, i.e., $\delta_i^p(i, j) = 1$. If not, then the physical unavailability of span i contributes (product-wise against the physical unavailability of span j) to the total unavailability of the path. In the second category, path p survives only if it has the priority to warrant a restoration path under the particular dual failure scenario on *both* failure spans (because each is in the path). Note that in [Equation 8.14](#) U_j^{phy} cannot be pulled out as a common factor because in the general case U_j^{phy} of each span differs and j is indexed differently on the two summations.

8.4.4 Implications for an Ultra-High Availability Class of Service

There is a striking practical implication that can be realized from the above. Assume for a moment that experimental dual failure trials such as those above ([Section 8.3.3](#)) showed that for a given network and spare capacity design $R_2(i, j) > X$; $\forall (i, j) \in S^2 \setminus \{i = j\}$. Then, if we have less than $X\%$ of all demands on each O-D in a predefined high priority service class, then the question of whether $\delta_i^p(i, j) = 1$ in [Equation 8.14](#) is equivalent to asking simply if path p is a *priority* service path. If we reconsider the experimental results of [Section 8.3.3](#) in this light it becomes apparent that a significant fraction of all demands could be in a *priority service class that would never experience any outage due to dual failures*. The only exception would be for dual failures isolating a degree-2 node, although that is already strictly recognized when we say we are discussing a network where $R_2(i, j) > X$; $\forall (i, j) \in S^2 \setminus \{i = j\}$ and $X > 0$. If there was even *only one* restoration path feasible under the (i, j) scenario, then the *top* priority path, which we imagined above, would get it and suffer no outage. But the results actually show far higher $R_2(i, j)$ recovery levels than this are possible. In fact [Figure 8-2](#) shows that 95% of the time the particular test network could support more than 60% dual-failure restorability. For all intents and purposes this implies that 60% of all service paths in that network could be given dual failure survivability simply by making them the priority group for consistent first allocation of any available restoration paths. Although not also shown, when the data underlying [Figure 8-2](#) are further analyzed to eliminate dual failures adjacent to degree-2 nodes, a minimum of ~20% of all paths—routed through degree-3 and higher nodes—could be given strictly 100% immunity against dual failures.^[2] It would take a triple failure to affect such priority service. But these interpretations are still conservative because they assume that any priority path is equally likely to be routed over any span of the graph. This means that, just as we might avoid routing an otherwise dual-failure protected service over any degree 2 nodes, in practice we might also alter the routing of priority paths to specifically exploit high $R_2(i)$ on some spans, or to avoid low $R_2(i)$ on others, or to avoid using specific combinations of spans for which $R_2(i, j)$ is low. By $R_2(i)$ in this argument we mean the average restorability of a specific span i given its own failure in the presence of all possible other single-span failures (and assuming span i 's working channels have equal priority to access restoration paths as do those of each other co-failed span, i.e.,

^[2] In other words, in the plot corresponding to [Figure 8-2](#) with degree-2 node effects removed, the histogram bin at $R_2(i, j) = 1$ is over 20% in both cases.

Equation 8.15

$$R_2(i) = 1 - \frac{\sum_{j \in S | i \neq j} N(i, j)}{(|S| - 1) \cdot \sum_{i \in S} w_i}$$

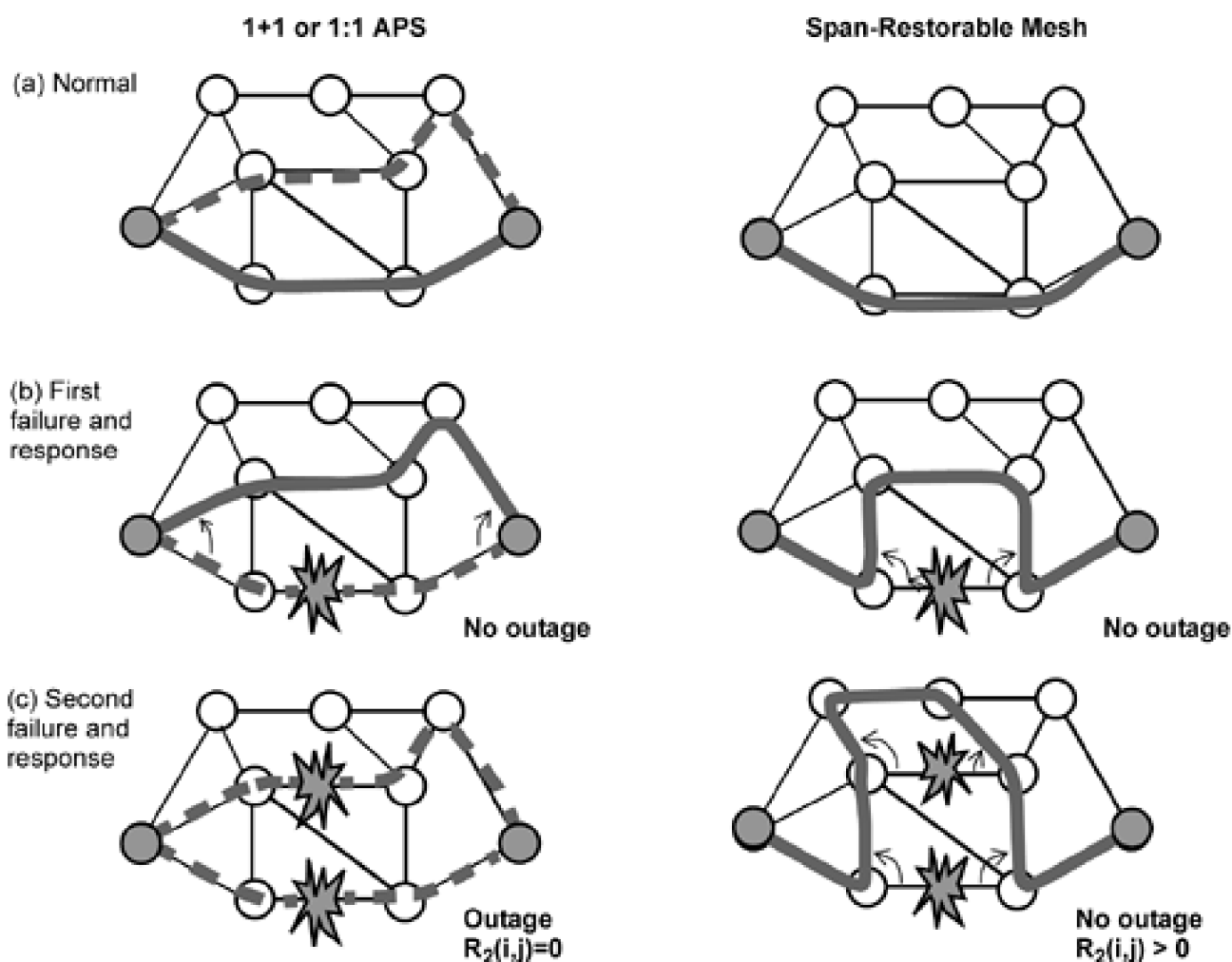
This has remarkable implications for how extremely high the availability of a priority service class could be in a span-restorable mesh network. It directly implies that such priority paths would see availability that is categorically *better* than that with dedicated 1+1 APS. This may initially seem to be a surprising claim because 1+1 APS usually stands in our minds as virtually the best service you could ever provide. And yet it follows directly that the priority service paths in the mesh get even better availability because the 1+1 APS path pair has a definite set of (i, j) pairs for which most assuredly, $R_2(i, j) = 0$ (and not involving degree 2 nodes). Any dual failure combination that hits both arms of the 1+1 path is directly outage-causing. This is the specific set of $H_A \times H_B$ combinations of physical span failures falling on each arm of the 1+1 path pair where H_A , H_B are the number of spans in the A and B signal feeds of the 1+1.

Another way to see this is that even though the backup path of a 1+1 path arrangement is fully *dedicated*, it is also *fixed*. In contrast, the spare capacity of the mesh is *shared* but its configuration is flexible and *adaptive*. It is as if the backup path that is effectively provided by span restoration against a normal single failure, itself has a kind of hidden further backup, should *it* fail too. In effect, the meaning of a network having $R_1 = 1$ and $R_2(i,j) > X\%$ is that all paths are protected against a single failure, and, if a double failure arises at least $X\%$ of affected paths can be given *yet a further backup path*. Ultimately this is attributed to the adaptive nature of the span-restorable mesh in being almost always able to come up with at least partial $R_2(i,j)$ in almost all scenarios. It is the secondary response ability that in effect acts like a 1+1 against single failures, but is most often able to find yet another backup a priority path when the 1+1 itself is hit on both primary and backup.

Comparing Mesh and 1+1 APS or Rings under Dual Failures

Figure 8-5 illustrates this with a side by side comparison of a 1+1 APS setup and a path through the mesh that has the same route as the "A" feed of the 1+1. A first failure hits that path. The 1+1 APS switches to its "B" signal feed and the mesh deploys its first failure restoration reaction, for which $R_1 = 1$. So far so good. Both services survive. But next, let a second failure hit the "B" feed of the 1+1 APS setup, and assume the same failure hits the mesh restoration path of the first failure for the path of interest. Now, the 1+1 service is out. But if the $R_2(i,j)$ of the dual failure scenario in the mesh is even slightly above zero (and we see that it is often 60% or more), then the path of interest is restored again, and the service in the mesh continues without outage.

Figure 8-5. Why priority paths in a span-restorable mesh will have higher availability than even a dedicated 1+1 APS service can provide.



Because a path through any type of ring (BLSR, UPSR or FDDI are alike in this regard) is also "down" in any state where there is a failure on the working path and on the reverse direction through the ring [Gro99],^[3] the same argument of how mesh can outperform 1+1 APS for a priority service class also serves as an existence proof that a certain number of priority services in a mesh can always enjoy higher availability than in a corresponding ring-based network. Experimental comparisons have also validated this reasoning in carefully controlled studies of dual failure related path outage in comparable ring and mesh based transport networks [ClGr03]. Thus, the high R_2 of a mesh restorable network designed for $R_1 = 1$ suggests that a commercially significant fraction of all service paths could actually be given ultra-high availability. Such "platinum" service customers would receive a guarantee that their service path would only be affected by a triple span failures (or dual failures adjacent to a degree-2 node, which no scheme can protect against). This takes their availability guarantees one order of magnitude higher, directly into the 30 seconds or less (per annum) range of unavailability.

^[3] A common misunderstanding is that rings cannot withstand dual failures at all. But the ring as a whole is not the entity to consider. Rather, for any given dual failure on a ring some individual *paths* through the ring will be protected, while others paths will not survive. For more details, see the derivation of the $Td1()$ function in [Gro99].

8.4.5 Link to the 1FP-2FR Concept

Finally note the link between this discussion and that of [Section 5.2.7](#) that proposes how an adaptive dual failure restoration response can be coupled with a faster preplanned protection reaction to single failures. We call this the "first failure protection, second failure restoration" (1FP-2FR) concept. Only in the event of a second failure that interacts with the restoration path set of the first, resulting in less than 100% immediate restoration, is the truly adaptive, even if possibly slower, restoration protocol executed directly in real-time. The adaptive restoration protocol is most of the time serving as the engine for background preplanning of protection reactions against single failures. But any time the protection level is not 100%, it executes directly to produce the non-zero $R_2(i,j)$ that gives a high recovery level to priority paths.

Part of understanding why mesh-restorable networks can achieve such high availability with the 1FP-2FR concept is that contrary to what might first be thought, the speed of response to a single failure does *not* matter greatly to the availability. Far more important is the probability that a dual failure is restorable or not, even if this takes seconds. From an availability standpoint the debate about switching speed of rings being faster than mesh is basically irrelevant if the mesh has a higher $R_2(i,j)$. We can illustrate this with a simple example. Let the ring technology be granted a 50 ms switching time in response to a single failure. Its response, however, to failure on a span of the forward path of a signal through the ring, and other span on the back path through the ring is $R_2(i,j) = 0$ (whether UPSR or BLSR). This means on average that the affected signal path sees MTTR/2 of outage assuming the two span failures were independent. To be extremely generous regarding switching times, let us say that the mesh takes $t_r = 2$ seconds to restore a single failure and 5 seconds to deploy its dual failure response, which has $R_2(i,j) > 1$, and we are considering the availability of a premium path.

Both schemes have $R_1=1$ for single failures, so the unavailability due to single failures is only due to reconfiguration times. If the MTTR is

12 hours for physical repair, then $\lambda_i = U_s^{phy}/MTTR$ is the physical rate of failures on a single span i (refer to [Section 3.12](#)). The

unavailability due to single failure reconfiguration times on one span is then $U^* = \lambda_i \cdot t_r = U_s^{phy} \cdot t_r / MTTR$. If we again

assume $U_s^{phy} = 3 \times 10^{-4}$ from above we find that the equivalent unavailability of channels on restored spans is 3.5×10^{-10} for

ring-protected spans and 3.5×10^{-8} for mesh-protected spans, both of which are so small as to be irrelevant in practice. Even the higher mesh figure is 1 s per year. On the other hand, when we find ourselves in a dual failure scenario on the specific (i,j) pair of spans for this argument, the ring will engender a six hour outage (average overlap time for two independent failures at MTTR = 12 hours) and the mesh will have an outage of $t_r = 5$ s. Therefore, in comparing alternative networks that both have $R_1 = 1$, *what is most important to the availability is not the speed of restoring single failures, but the likelihood of being able to mount an effective recovery in the face of dual failures*, even if the latter takes relatively more time in reconfiguration. An example of this insight is to consider the availability of rings relative to mesh in a generic way. Despite a faster switching time for rings, and the lower redundancy of the mesh, the generality of the restoration process can lead to a high level of dual failure restorability, which counts a great deal toward restorability. But it hardly counts at all if in response to a single restorable failure the ring switches in 50 ms while the mesh takes 1000 ms. The higher R_2 of a mesh can be of greater importance and used to provide relatively high availability to all paths in general, or an ultra-high level of availability to premium service paths. The extra redundancy that rings embody is, in effect, used to buy high speed in response to single failures, but its fixed structuring limits the generality with which such redundancy can be used to overcome dual failures.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

8.5 Dual Failure Availability Analysis for SBPP Networks

In [Chapter 6](#) we covered SBPP-based network design including control of the design to limit the maximum number of sharing dependencies on any on spare channel.^[4] The limiting technique was motivated by a general recognition that doing so would enhance the dual failure restorability and hence availability. Now we take the availability-related consideration of SBPP further by identifying the different scenarios that can lead to SBPP outage.

^[4] An extended version of this section was submitted for publication in *Optical Networks Magazine* [\[DoCI03\]](#).

A case-by-case analysis of how dual failures affect SBPP is summarized in [Table 8-3](#). The categories are based on the location of each span failure relative to the primary path P1, and backup route P2, of an SBPP service path setup. There are three possibilities for the location of any single-span failure relative to a specific service path. The failure can be on the primary route of the service path, it can be on the backup route, or it can be on neither the primary route nor the secondary route (denoted "other"). The last two columns list the effects of each combination on the availability of the service path considered for both SBPP and, for comparison, 1+1 APS.

Simple reasoning can explain most of the cases, except the seventh category, which is also important because it is the only one for which the effect differs between SBPP and 1+1 APS. The situation is that an initial failure affects neither the primary service path nor its backup path, but a second span failure strikes the primary path. In 1+1 APS, the backup path is dedicated to the protection of the primary path, so as long as it is not itself affected by a failure it is guaranteed to be available. But for SBPP, because spare capacity is shared, there is no guarantee that the spare capacity will be available along an entire backup path if there has already been a span failure elsewhere. For instance, the initial failure might have caused some other service path A1 to switch to its own backup path A2, which might share some capacity with the backup path B2 used to restore failure of primary service path B1. In this case, a detailed inspection of all the paths affected by the first failure, and of the capacity that their backup paths are subsequently consuming, is needed to conclude whether a primary service path affected by the second span failure is left exposed to outage or not. To do this exactly requires a computer-experimental approach, which we describe next. Note in passing, however, that clearly the availability of 1+1 APS is an upper bound on the availability of paths with SBPP.

Table 8-3. How Dual Failures Affect Shared Backup Path Protection (SBPP)

Failure Category	Failure 1	Failure 2	Outage causing? (SBPP)	Outage causing? (1+1 APS)
1	P1	P1	No	No
2	P1	P2	Yes	Yes
3	P1	other	No	No
4	P2	P1	Yes	Yes
5	P2	P2	No	No
6	P2	other	No	No
7	other	P1	depends on sharing details	No
8	other	P2	No	No
9	other	other	No	No

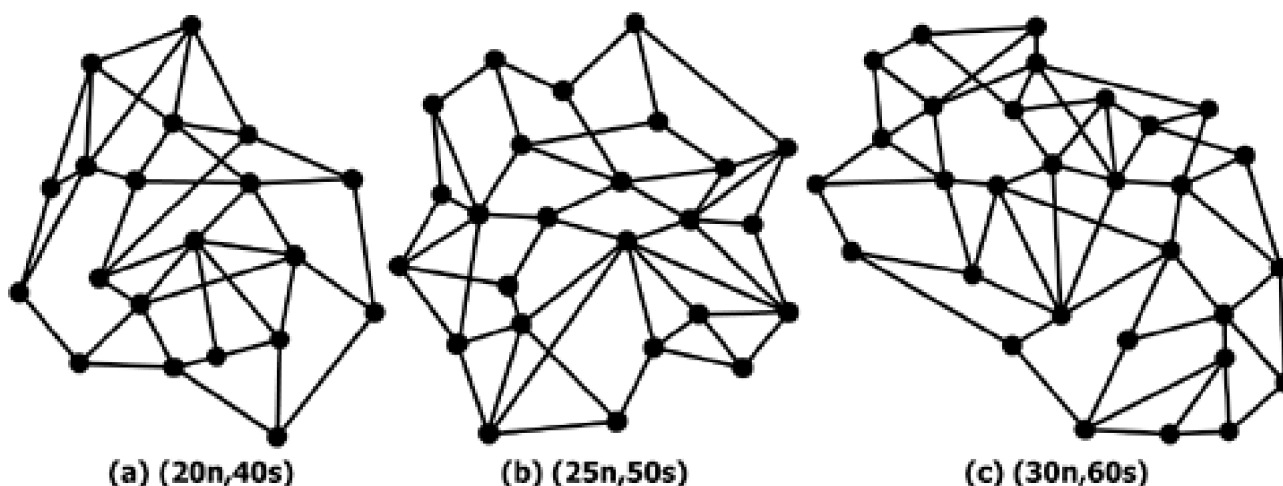
8.5.1 Experimental Comparison of SBPP and Span Restoration

The determination of dual-failure restorability for SBPP requires as input the set of demands, the routes for primary and backup for each demand, and the total spare capacity on each span (arising from the SBPP design model). The two failures are denoted a and b . Step 1 checks the restorability of service paths to the failure of the first failed span. For each primary affected by the first failure, we activate the corresponding backup route and seize the required spare capacity. Step 2 considers activated backup routes that may have been hit by the second failure. Under SBPP, such demands are failed. The next step (3) is to consider whether the backup paths for primaries affected by failure b can activate their backup paths given the spare capacity usage from step 1.

Let us now look at sample results from a study comparing SBPP and span-restorable networks using the dual failure restorability analysis methods covered so far [DoCl03]. Tests were conducted using three test network families, each family consisting of a master network and a series of progressively lower degree networks derived from the master by random elimination of nonessential spans.^[5] The network families were produced by starting with a master network and incrementally removing one span at a time (by random selection), subject to retaining biconnectivity. The master networks of each family are shown in Figure 8-6. The master networks are all of average nodal degree 4.0 and provide sample networks down to about degree 2.6.

^[5] Nonessential refers to spans that may be eliminated without the resulting graph losing biconnectivity. The method allows study of networking phenomena over a series of systematically related networks varying in nodal degree but having the same scale, node locations, and demand matrix for all test cases. See [DoGr01][GrDo02].

Figure 8-6. Master networks for comparison of dual failure restorability with SBPP and span-restorable network designs.



All networks were tested with all-pairs demand magnitudes being uniform random in $[1, \dots, 10]$. Demands were generated once for each master network and the same demands for each O-D pair were reused for the other members of each set. The SBPP capacity designs were formulated with the set of five shortest distinct eligible route choices (by span-length) for possible backups on each O-D pair with a sharing limit of five. The span-restorable designs serve the identical demand patterns with shortest-path working routes and use of the set of eligible routes for restoration of each span failure up to 5 hops long. The second failure response of the span-restorable network is adaptive to the spare capacity usage of the first, but if the second failure hits restoration paths of the first failure they are not re-restored. With the fully adaptive model for span restoration (model (c) of Section 8.3.2) $R_2(a,b)$ is even higher, but the partly adaptive model seems like the fairest comparison to make against SBPP.

Figure 8-7 shows the average $R_2(a,b)$ for both schemes for all the test cases plotted against the corresponding network average nodal degree. With SBPP about 70-80% of all service paths would withstand a dual failure that affected them. The restorability of the same dual failures is approximately 20% higher still with span restoration. We can also see a modest but clear trend that as the connectivity increases the dual-failure restorability improves for both schemes. The relatively constant nature of the curves is thought to be due to two counteracting factors. One is the shortening of the working routes. With more spans in the network, there is a decrease of the number of service paths crossing each span and each path is exposed to fewer spans, so the average number of affected service paths for dual failures decreases. By itself this tends to improve the average $R_2(a,b)$. However, as nodal degree increases, both forms of

mesh-restorable network also become increasingly efficient, embodying less total spare capacity, hence tending to decrease R_2 . Evidently the shortening of primary routes seems to win out slightly overall.

Figure 8-7. Average dual failure restorability with SBPP and span-restorable network designs.

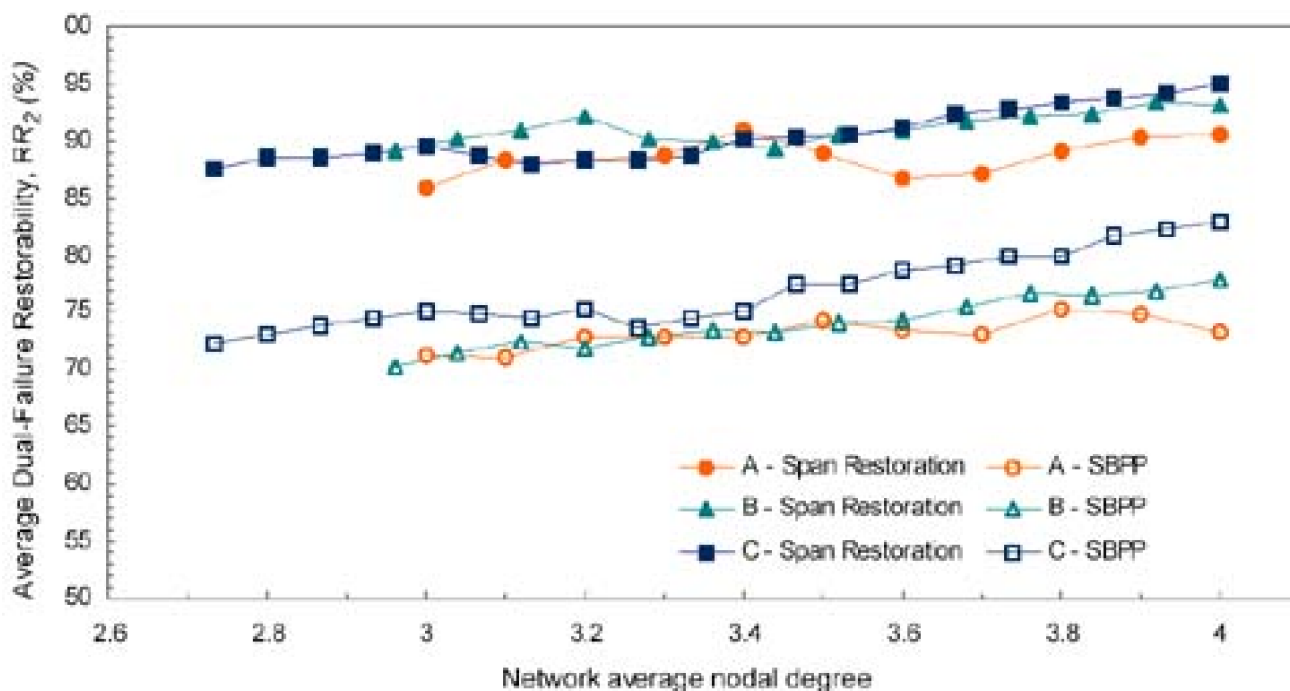
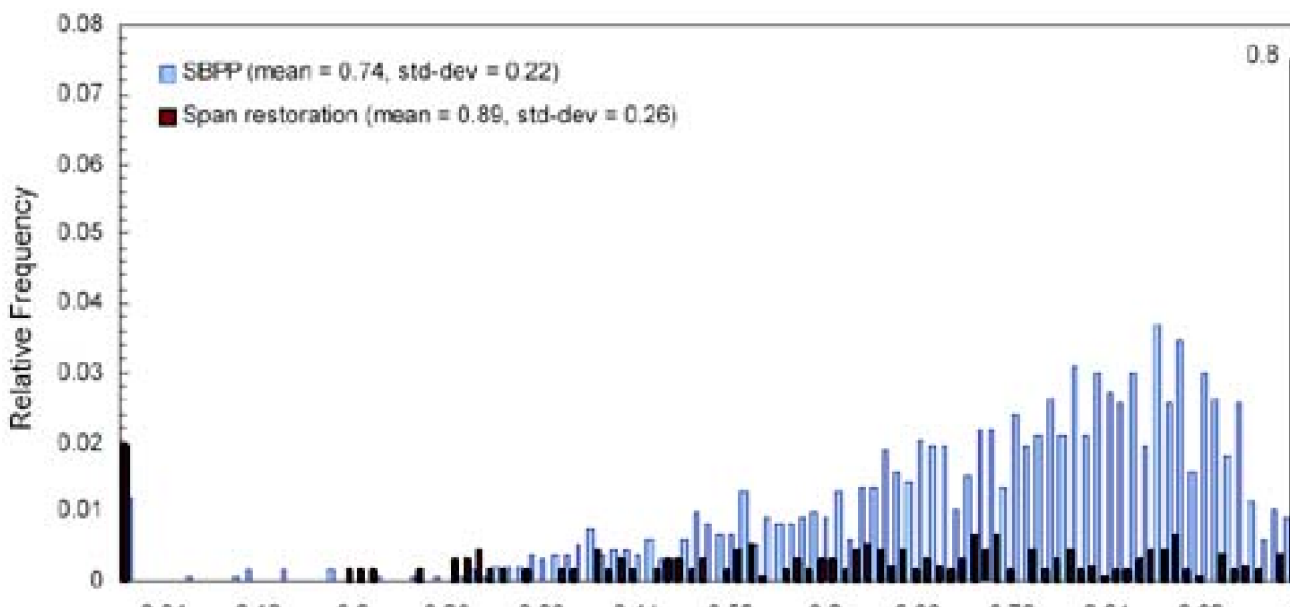


Figure 8-8 is a compares SBPP and span-restorable designs on the basis of individual failure pair $R_2(a,b)$ values. (The mean values of the data in Figure 8-8 are the points of Figure 8-7.) The particular result is from the 37-span derivative from master network B but the distributions are representative of all the other test networks studied. The right-most bin of $R_2(a,b) \sim 1$ show that under span restoration roughly 80% of dual failures have no impact at all. In addition, the only cases where $R_2(a,b)=0$ under span restoration is where dual failures isolate a degree 2 node. In contrast a dual span failure is more likely to cause demand outage in SBPP. This is consistent with SBPP's basic similarity to 1+1 APS. 1+1 APS experiences hard outage for any (a,b) failure combination where $a \in primary \cup b \in backup$ or vice versa. Actual SBPP performance is spread out below the 1+1 dedicated APS performance bound by the detailed sharing-related outage from effects in row 7 of Table 8-3.

Figure 8-8. Histogram of individual $R_2(a,b)$ levels under SBPP and span restoration.



0.08 0.12 0.2 0.28 0.36 0.44 0.52 0.6 0.68 0.76 0.84 0.92 1

Dual failure restorability, $R_2(a, b)$

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

8.6 Optimizing Spare Capacity Design for Dual Failures

In this section we move from the analysis of dual failure restorability of a given network, and how that affects path availability, to consider ideas about synthesizing networks with certain enhanced dual failure properties.^[6] A natural first exercise is to see how much it would cost to simply design in enough spare capacity to withstand all dual failures, i.e., $R_2 = 1$ design. We refer to this as the *dual failure minimum capacity design (DFMC)* problem. The results of DFMC are somewhat surprising: although we saw above that the average $R_2(i,j)$ may be relatively high, it turns out that strictly achieving $R_2(i,j)=1$ for all (i,j) is quite expensive. A next question is therefore to see how high an R_2 can be achieved with a given limit on additional capacity expenditure over the $R_1=1$ condition. This is called the *dual failure maximum restorability (DFMR)* problem. Finally we look at how to place capacity at minimum cost to serve certain paths only at $R_2=1$ and others at only $R_1=1$. This is called *multi-restorability capacity placement design (MRCP)* and relates directly to how one would plan capacity and routing for a defined premium service class that has an $R_2=1$ guarantee by design. In the first two problems demands are first shortest-path routed, generating the w_j values which are inputs to the problem. The last, MRCP, is a type of joint optimization problem. We first introduce the three design models, followed by a discussion of sample results.

^[6] During development of the material in this section, a preliminary version of it was presented at [QIGr02b](#).

8.6.1 Minimum Capacity to Withstand All Dual Failures (DFMC)

The DFMC formulation finds a minimum total spare capacity assignment that guarantees full restorability of all dual span failure scenarios. This formulation tells us the price for reaching $R_2 = 1$. For obvious reasons, a feasible solution to this problem cannot be found for a network with degree 2 nodes or 2-edge cuts of the network graph. Therefore our later tests of DFMC are limited to graph topologies that qualify. In practice, however, transport networks often have degree 2 nodes. DFMC could therefore be considered a formulation for the assurance of $R_2 = 1$ on the "mesh backbone" component, also called the "meta-mesh" abstraction ([Section 5.8](#)) of an overall network. In this context DFMC can be applied by logical removal of a degree 2 node between spans i, j and assertion that $w_i = \max(w_i, w_j)$ where w_i applies to the single logical edge arising from the degree 2 node removal. Alternately DFMC capacity design could be applied to a connected subnetwork of the original graph that excludes all degree 2 nodes. The idea would be that this subnetwork is used for provisioning all " $R_2 = 1$ " (i.e., ultra-available) service paths. A more basic reason for interest in DFMC is, however, that it serves as a benchmark and satisfies our curiosity about how much spare capacity would be needed to literally protect all demands against all possible dual failures.

For DFMC w_j and s_j have the usual meanings but we now also define:

- $f_{i,j}^p$ = the restoration flow assigned to the p^{th} eligible route for restoration of span i when span j is also failed (integer).
- C_∞ = a constant representing an arbitrarily high amount of restoration flow, higher than the largest restoration flow assigned to any route in an actual design solution.
- $\delta_{i,k}^p = 1$ = 1 if the p^{th} eligible route for restoration of span i crosses span k , zero otherwise.

In addition, for the models involving dual-failure scenarios, note that we employ the extra index variable k for greater clarity. This allows us to use the usual i, j indexes both in context of referring to a span failure (together, i, j specify a particular dual-failure scenario). This leaves k to be used (as j normally otherwise is) to refer to any other span in general in the context of a surviving span or as a general span index without implication that it is part of the failure scenario.

DFMC

Minimize

Equation 8.16

$$\sum_{k \in S} c_k \cdot s_k$$

subject to:

- Dual failure restorability:

Equation 8.17

$$\sum_{p \in P_i} f_{i,j}^p = w_i \quad \forall (i,j) \in S^2 | i \neq j$$

- Dual failure routing limitations:

Equation 8.18

$$f_{i,j}^p \leq C_{\infty} \cdot (1 - \delta_{i,j}^p) \quad \forall (i,j) \in S^2 | i \neq j, \forall p \in P_i$$

- Spare capacity required:

Equation 8.19

$$\sum_{p \in P_i} f_{i,j}^p \cdot \delta_{i,k}^p + \sum_{p \in P_i} f_{j,i}^p \cdot \delta_{j,k}^p \leq s_k \quad \forall (i,j,k) \in S^3 | i \neq j, i \neq k, j \neq k$$

- integer capacities and either integer or relaxed flows.

[Equation 8.16](#) asserts the restorability of each failure span i under all dual span-failure scenarios. In conjunction with the $\delta_{i,j}^p$ indicator parameters, [Equation 8.18](#) uses the "infinity constant", C_{∞} , to ensure that span j can support any required amount of restoration flow over itself when it is *not* part of the failure scenario, but that it cannot be used for restoration of span i in the (i,j) failure scenario. [Equation 8.19](#) ensures that there is enough spare capacity on each span to support the largest simultaneously imposed restoration flow on each span, over all dual failure scenarios. Here, every surviving span k is considered with respect to the restoration flows asserted over it for span i (the first sum) plus any simultaneously imposed flow that is for restoration of span j in the same scenario (the second sum). No

explicit statement of single failure restorability is needed as this is implicitly feasible under design for all *dual* failures.

Note that DFMC can be easily adapted to generate the minimum spare capacity to support restoration against any specific *subset* of dual failures as well. To do this one can generalize $\forall (i, j) \in S^2 | (i \neq j)$, which strictly generates all dual failures, into $\forall f \in F$ where F is a set of *specific* failure scenarios of interest. In this case, however, any spans that do not appear as part of dual failure scenarios then must be asserted directly as single failure scenarios in addition to restorability of the specific dual failures of concern. This is similar to how we later design for a specific set of SRLG hazards.

8.6.2 Dual Failure Maximum Restorability (DFMR)

An obvious concern with DFMC is that the spare capacity costs for strict restorability of *all* dual failure combinations might be extremely high. In fact this is confirmed by results that follow. And yet results have already shown us that R_2 tends to be relatively high on average in networks that are only designed for $R_1=1$. The implication is that relatively few specific dual failure scenarios may be responsible for the large multiplier on spare capacity needed for $R_2=1$ relative to the $R_1=1$ condition. It makes sense therefore to try turning the problem around and asking instead what is the highest achievable level of R_2 with a given limit or a "budget" on total spare capacity investment. To do this we need to add:

- B = a budget limit for total spare capacity cost (input parameter).
- f_i^p = the restoration flow assigned to the p^{th} eligible route for restoration of span i as a single isolated-span failure scenario (integer).
- $N(i, j)$ = number of non-restored working units under dual failure of spans (i, j) (a new integer variable). Note that $N(i, j)$ is the same quantity as found experimentally in R_2 analysis above. Here, however, it is involved directly as a design variable.

DFMR(B)

Minimize

Equation 8.20

$$\sum_{(i, j) \in S^2 | i \neq j} N(i, j)$$

subject to:

- Single failure restorability:

Equation 8.21

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

- Unrestored paths:

Equation 8.22

$$N(i,j) = w_i + w_j - \left(\sum_{p \in P_i} f_{i,j}^p + \sum_{p \in P_j} f_{j,i}^p \right) \quad \forall (i,j) \in S^2 | i \neq j$$

- Dual failure restoration flow maximums:

Equation 8.23

$$\sum_{p \in P_i} f_{i,j}^p \leq w_j \quad \forall (i,j) \in S^2 | i \neq j$$

- Dual failure routing limitations:

Equation 8.24

$$f_{i,j}^p \leq C_\infty \cdot (1 - \delta_{i,j}^p) \quad \forall (i,j) \in S^2 | i \neq j, \forall p \in P_i$$

- Essential spare capacity for R_1 :

Equation 8.25

$$\sum_{p \in P_i} \delta_{i,k}^p \cdot f_i^p \leq s_k \quad \forall ((i,k) \in S^2 | i \neq k)$$

- Additional spare capacity to enhance R_2 :

Equation 8.27

$$\sum_{p \in P_i} f_{i,j}^p \cdot \delta_{i,k}^p + \sum_{p \in P_j} f_{j,i}^p \cdot \delta_{j,k}^p \leq s_k \quad \forall (i,j,k) \in S^3 | i \neq j, i \neq k, j \neq k$$

- Budget restriction:

Equation 8.28

$$\sum_{k \in S} c_k \cdot s_k \leq B$$

[Equation 8.21](#) ensures that $R_1 = 1$ for every single-span failure. [Equation 8.22](#) defines $N(i,j)$, which is the number of working capacity units that are *not* restored in case of a dual failure on spans i, j . The logic is that by minimizing the sum of $N(i,j)$ over all dual failure scenarios, one is indirectly maximizing R_2 through [Equation 8.1](#) and [Equation 8.2](#) (and thereby also maximizing the availability through [Equation 8.11](#)). Constraints of [Equation 8.23](#) ensure that the number of paths assigned to the restoration of a span in a dual span failure scenario is at most equal to the number of working channels to be restored. Without this constraint system, under minimization of $N(i,j)$, restoration flow variables could be otherwise driven above the actual requirements of the problem so as to gain credit under [Equation 8.22](#). [Equation 8.24](#) is identical to [8.18](#), forcing the exclusion of restoration routes for span i from using span j and vice-versa during the (i,j) scenario, while allowing use of span j for all other scenarios. [Equation 8.25](#) ensures adequate spare capacity for every single failure case. Finally [Equation 8.27](#) is identical in form to [Equation 8.19](#) in DFMC, but in this context it does not assert full dual failure restorability (as it does above). It now serves only to permit such dual failure restoration flows as are feasible under a given distribution of spare capacity amounts. It is through the action of [Equation 8.27](#), working under the total budget constraint of [Equation 8.28](#), that the solver effectively chooses the extent to which it will cover each dual failure scenario using the available budget. The aspect of selective coverage is also why an explicit assurance of single failure restorability is needed here ([Equation 8.21](#)), but not in DFMC. We will soon look at sample results using DFMR as a vehicle through which, by varying B , we can study how R_2 increases with total capacity.

8.6.3 Pure Redistribution of Spare Capacity to Enhance R_2

A special case of interest for application of DFMR(B) should be noted. If we assume that a basic solution to SCA (min spare capacity for $R_1 = 1$) has already been obtained, resulting in a vector of s_j^0 initial spare capacity values, then:

Equation 8.29

$$DFMR\left(B = \sum_{k \in S} c_k \cdot s_k^0\right)$$

produces a pure redistribution of the initial spare capacity that maximizes the dual failure restorability to the extent possible with only the capacity of the initial $R_1=1$ design. This provides another instance of a kind of bicriteria optimization ([Section 4.16.4](#)). In networks where the s_j distribution of spare capacity is obtained from SCA (or JCA) we follow those designs with an application of DFMR at the same total spare capacity to secondarily optimize their ability to withstand dual failures as much as possible.

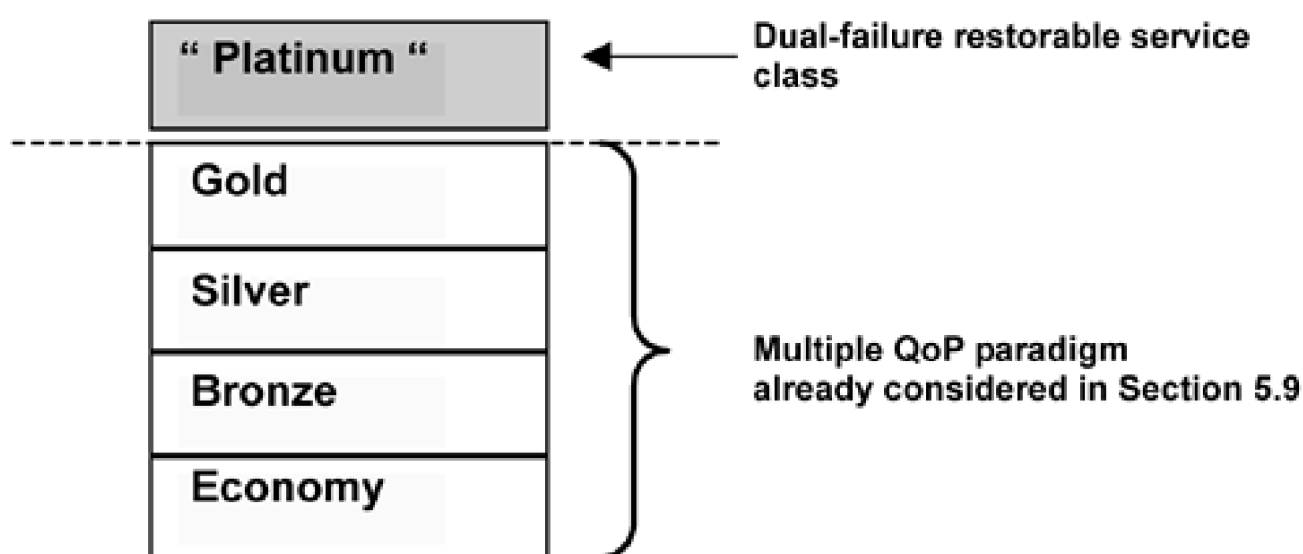
8.6.4 Multi-Service Restorability Capacity Placement Design (MRCP)

In a similar vein to the discussion of [Section 8.4.3](#), we note that DFMR maximizes the overall R_2 that is achievable for the network as a whole, but we have no control over which individual service paths benefit the most or least in the resulting design. Any R_2 enhancement produced by DFMR is a general benefit to the integrity of the network as a whole, but it is not targeted in any specific way to individual paths. Building on previous ideas of multi-QoP design, however ([Section 5.9](#)), we could consider the establishment of a "platinum" service class of the highest priority nature defined by the guarantee of $R_2 = 1$ by design.

In the prior multi-QoS considerations, there were no considerations of withstanding dual failures. Gold (the top service level) was defined as the $R_1=1$ service class. All other classes were defined "down" from this ceiling, such as best-effort or R_1 , no restorability, and preemptible. But now we can consider a design strategy to create a super-high priority class of service that is actually *assured by design* of 100% restorability for all single *and* dual failure scenarios. For simplicity we treat the design problem only for the addition of an $R_2=1$ subset of demands to a network where all other demands require $R_1=1$ or no provision for restorability.

This may not necessarily take that much extra spare capacity because we have already seen that R_2 is relatively high to start with. The problem is that such R_2 levels as intrinsically exist in an $R_1=1$ network are not coherently structured as might be desired onto individual platinum-class service paths. Therefore while we may still have to add capacity, we also expect to obtain some of the desired R_2 paths simply by restructuring the inherent R_2 resource of an R_1 -capacity network. Results to follow show that MRCP provides an economical way to design-in support for this platinum service class that stacks *above* the existing multi-QoS paradigm of [Section 5.9](#), in the sense illustrated in [Figure 8-9](#). The new super-high availability service class can be added to the range of possible QoS classes with remarkably little additional cost within span-restorable networks that are already efficiently designed to only support single failure restorability.

Figure 8-9. How the MRCP capacity design model extends the multi-QoS hierarchy.



The multi-restorability capacity placement (MRCP) design model allows us to target and structure the dual failure restorability specifically to the intended services or customers. Every demand receives a specific class of restorability guarantee on every span, end-to-end over its route. To each demand we therefore assign one of the following restorability service class designations:

- R_2 restorability = "platinum" service class: assured restorability to any single or dual span failure.
- R_1 restorability = ordinary assured restorability to any single-span failure (and best-efforts for dual failures).
- R_0 restorability = for generality this is an easily included additional class of services paths. It receives best-efforts in both single and dual failures but has no assured restorability.

In contrast to DFMC and DFMR, MRCP is approached as a joint optimization. To provide R_2 , R_1 , R_0 restorability on a per-path basis requires explicit handling and recognition of working path structures in any case, so it makes sense that by allowing us to choose among options for eligible working routes we can realize the required R_2 paths with greater efficiency than if working routes are on strictly shortest paths.

The formulation finds the minimum total cost of working plus spare capacity, and the routing of each demand, to satisfy the restorability class of service of each demand. For MRCP a *demand group* is now defined as one or more demand units on the same O-D pair and in the same service class. Demand groups must be defined for MRCP to distinguish between demands of different service classes on the same O-D pairs. The sum of all demand groups on an O-D pair equals the prior single service demand bundle quantities, d^j , on each O-D pair. Thus, index r now indexes not just over all O-D pairs but also through each demand group on each O-D pair. To the parameters of DFMC and DFMR above we also need to add:

- D = Set of demand groups, index r .
- Q^r = Set of eligible working routes for demand group (the same as in JCA).
- d^r = the number of individual demand units in the r^{th} demand group.
- $\zeta_j^{r,q} = 1$ if the q^{th} eligible working route for demand group crosses span j , 0 otherwise.
- $\psi_1^r = 1$ if the r^{th} demand group is in the $R1$ or $R2$ service class, 0 otherwise.
- $\psi_2^r = 1$ if the r^{th} demand group is in the $R2$ service class, 0 otherwise.

MRCP

Minimize

Equation 8.30

$$\sum_{k \in S} c_k \cdot (s_k + w_k)$$

subject to:

- All demands are routed:

Equation 8.31

$$\sum_{q \in Q^r} g^{r,q} = d^r \quad \forall r \in D$$

- Working capacity:

Equation 8.32

$$w_j = \sum_{r \in D} \sum_{q \in Q^r} \zeta_j^{r,q} \cdot g^{r,q} \quad \forall j \in S$$

- Single failure restorability for $R1$ or $R2$ services:

Equation 8.33

$$\sum_{p \in P_i} f_i^p = \sum_{r \in D} \sum_{q \in Q'} \zeta_i^{r,q} \cdot \psi_1^r \cdot g^{r,q} \quad \forall i \in S$$

- Dual failure restorability for R_2 services:

Equation 8.34

$$\sum_{p \in P_i} f_{ij}^p = \sum_{r \in D} \sum_{q \in Q'} \zeta_i^{r,q} \cdot \psi_2^r \cdot g^{r,q} \quad \forall (i,j) \in S^2 | i \neq j$$

- Dual failure routing limitations:

Equation 8.35

$$f_{i,j}^p \leq C_\infty \cdot (1 - \delta_{i,j}^p) \quad \forall (i,j) \in S^2 | i \neq j, \forall p \in P_i$$

- Spare capacity for R_1 :

Equation 8.36

$$\sum_{p \in P_i} \delta_{i,k}^p \cdot f_i^p \leq s_k \quad \forall (i,k) \in S^2 | i \neq k$$

- Additional spare capacity to support R_2 requirements:

Equation 8.37

$$\sum_{p \in P_i} f_{i,j}^p \cdot \delta_{i,k}^p + \sum_{p \in P_j} f_{j,i}^p \cdot \delta_{j,k}^p \leq s_k \quad \forall (i,j,k) \in S^3 | i \neq j, i \neq k, j \neq k$$

Constraints 8.31 and 8.32 are the basic pair of constraints seen in any joint model having to do with assigning routes to all demands and generating the corresponding working capacities. Constraints of [Equation 8.33](#) ensure that adequate restoration flow exists for each single-span failure affecting demands that warrant R_1 restorability. [Equations 8.34](#) and [8.35](#) together make the same assurance of adequate restoration flow for each dual failure scenario, but only for R_2 demand groups. The last two constraints generate the spare capacity as required to support the largest combination of restoration flows over each span from either single failure restoration of both R_1 and R_2 paths or from dual failure restorability for the R_2 paths alone.

8.6.5 Experimental Results

AMPL implementations of DFMC, DFMR, MRCP were solved with integer capacity and flow variables with CPLEX either to optimality or a MIP gap of 0.001. The six networks from [Table 8-1](#) that had no degree 2 nodes or cuts were taken as test cases for each design model. All DFMC and DFMR problems solved in less than 30 minutes, most of them in seconds. Good feasible solutions to the MRCP problems were typically found in a few minutes, but complete solution could take several hours. All cases were formulated with the complete set of distinct routes up to a hop limit of five for restoration flows. In MRCP problems the policy for routing of working demands was as follows: The eligible route set for R_1 demand groups was the complete set of all distinct routes found at the lowest hop limit that provided a minimum of three distinct working route choices for each demand. That is, the working path hop limit would be increased until a minimum of three eligible working routes are found, then all routes at that hop limit are represented. For R_2 demands the set of eligible routes was based on the shortest-path hop limit for each R_2 demand group plus one. All distinct routes up to that hop limit, inclusively, were then provided for each demand group. This composite policy typically provided 10 to 15 eligible routes for R_1 demands and at least twice that number of options for R_2 demands. For simplicity in assessment of the results, test cases exclude any R_0 demands for the MRCP cases. R_0 demand groups, when present, would be always explicitly routed over shortest paths because this is optimal when no corresponding investment in spare capacity is made for them.

DFMC Results: The Cost of 100% Dual-Failure Restorability

[Table 8-4](#) shows results obtained with the DFMC formulation for the networks in which 100% dual failure restorability was topologically feasible. For reference, the redundancy of the basic designs for $R_1 = 1$ is restated for comparison. Because both SCA and DFMC designs are based on the identical set of working routes and working capacities the ratio of redundancies gives us the increase factor on the total spare capacity required to achieve $R_2 = 1$ compared to $R_1 = 1$. From these tests it appears that one would have to invest 2.5 to 3 times as much spare capacity as otherwise to obtain full dual failure restorability.

Table 8-4. Test Results with the DFMC Design Model

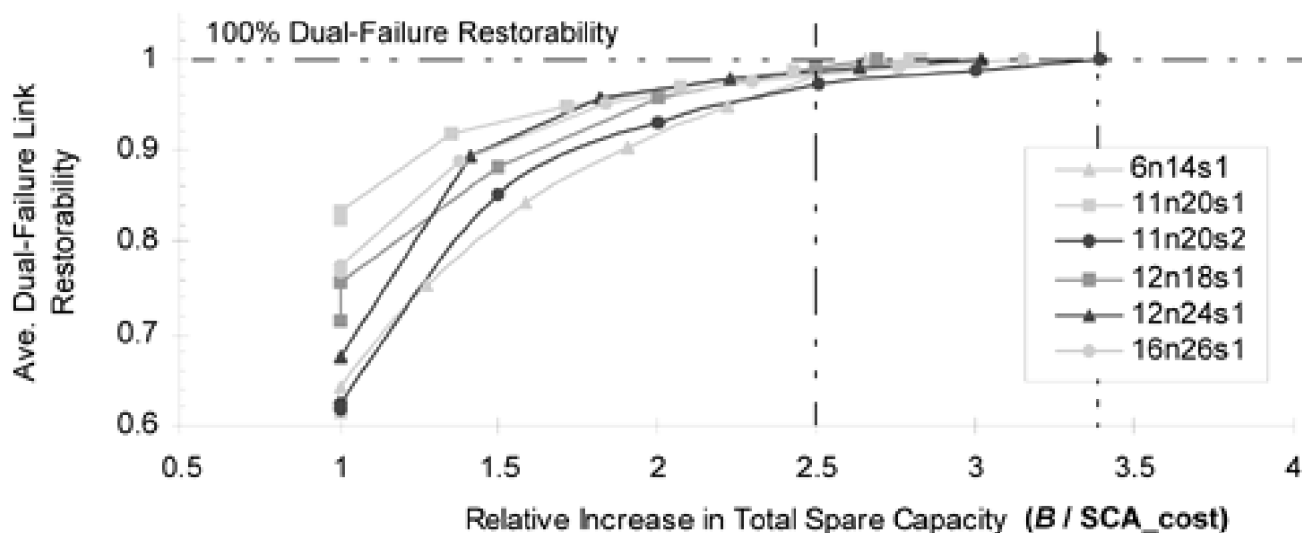
Net	Nodes	Spans	Redundancy at $R_1=100\%$ (SCA)	Redundancy at $R_2=100\%$ (DFMC)	Spare Capacity Increase Factor (times)
6n14s	6	14	44.1	116.8	2.65
11n20s1	11	20	91.6	258.9	2.83
11n20s2	11	20	47.5	161.3	3.4
12n18s1	12	18	99.8	268.6	2.7
12n24s1	12	24	48.4	145.9	3.01
16n26s1	16	26	78.8	248.2	3.14

This indication of such a high price for achieving complete $R_2=1$ restorability is interesting to contrast against the finding of [Section 8.3.3](#) that relatively high R_2 levels arise spontaneously just as a side effect from designing for $R_1=1$. This means that while we can achieve relatively high R_2 levels with modest if any additional capacity, the price strictly to completely assure full $R_2 = 1$ is very high. In other words, it is a phenomenon where achieving the "last few percent" in R_2 is extremely expensive.

DFMR Results: Budget-limited R_2 Enhancement

Figure 8-10 shows sample results for the "budget-oriented" DFMR formulation on the six test networks. Each of these curves is based on a series of DFMR runs as the budget limit B is varied. Each curve shows the improvement in R_2 as the spare capacity budget is increased under DFMR. In each case there is an initially better than linear growth of R_2 as the spare capacity is increased but this slows greatly as R_2 nears unity and merges with the high capacity requirements seen in the limiting case of $R_2 = 1$ in DFMC.

Figure 8-10. Results of DFMR showing how the achievable R_2 rises with budget limit.



Note the special interpretation of "pure redistribution" of spare capacity under the DFMR formulation when B is no more than that required for basic $R_1 = 1$ (i.e., the basic SCA spare capacity cost). This is the $B / SCA_cost = 1$ point on the x-axis of Figure 8-10. With no additional budget for spare capacity beyond that for R_1 design, the DFMR performs a redistribution of the spare capacity for the single failure restorability to enhance the achievable R_2 . In these results this zero cost redistribution benefit to R_2 is relatively small, however, under 5% in all cases.

MRCP Results: Tailoring R_2 to the Priority Service Paths

For tests of the multi-service MRCP formulation the same total demand matrices are used as in the DFMC and DFMR results but we designate a certain fraction of the individual demands on each O-D pair into to the R_2 service categories, with the remainder obtain an R_1 design treatment for the test cases. Although the general formulation allows R_0 , R_1 , R_2 categories, for test purposes it is more meaningful to retain R_1 as a minimum requirement for all demands. This allows better comparative interpretation against the most familiar case where all demands are R_1 type. This is also conservative in terms of the potential R_2 -serving ability and it reduces the dimensionality of the results to be presented. For tests of MRCP the fraction of demands on each O-D pair that were given R_2 status was varied from 0 to 40%. Table 8-5 shows the total working plus spare capacity requirements to support the given mix of " R_1 -assured" and " R_2 -assured" services (R_2 class implies an R_1 assurance too). The bracketed figures are the amount of capacity increase of each case over its pure JCA reference design (where all demands are R_1 service class only).

Table 8-5. Total Capacity at Varying Levels of R_2 Services Guarantees

Net	JCA: all R_1 services	with 20% R_2 services	with 30% R_2 services	with 40% R_2 services
6n14s	203	203 (0%)	206 (~0%)	211 (4%)
11n20s1	944	948 (~0%)	969 (2.6%)	1074 (13.8%)
11n20s2	405	405 (0%)	425 (5%)	449 (10.9%)
12n18s1	894	1030 (11%)	1342 (50%)	1541 (72%)
12n24s1	569	570 (~0%)	587 (3.2%)	611 (7.3%)
16n26s1	1497	1572 (5%)	1970 (31.6%)	2218 (48%)

These results are quite significant. They tell us, for instance, that in network 12n24s1, 20% of the demand flow on every O-D pair could be given assured dual failure restorability without adding any capacity to the network. Given the premium that might be charged for such an ultra-high quality of service guarantee, being able to support even 10% of all demands in this class could be quite significant commercially. Depending on the network, the "breakpoint" of this almost free ability to support R_2 services does not arrive until 30% or even 40% of demands are in the R_2 -assured class. This indicates a tendency of these networks to be capable of supporting select class of "ultra-high availability" $R_2 = 1$ customers without the huge capacity requirement for the network to have $R_2 = 1$ as a whole. Inspection of the designs show that the satisfaction of the R_2 service class arises not only through redistribution of spare capacity to target high R_2 on the key spans, but also from changes in the routing of working paths that bring them through regions of the network that are more easily made R_2 -restorable. These results are probably also only conservative estimates of the operational potential for multi-service design because the present MRCP formulation is also non-modular and we present results with no R_0 service class members. Modularity only increases the inherent R_2 level and R_0 class services require no spare capacity so, to the extent they might take the place of some of the R_1 demands in these results, this can only further increase the fraction of R_2 -servable customers at the same total capacity investment.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

8.7 Dual Failure Considerations Arising From Express Routes

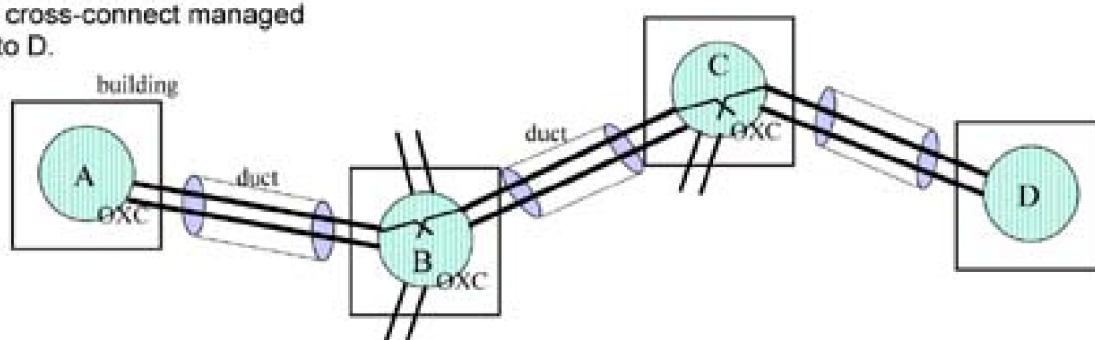
8.7.1 Express Routes and Nodal Bypass

Most studies on span-restorable networks assume that all working signal paths which physically go through a node location are routed through the cross-connect system at the location. Some network operators adhere to this as a policy because it provides the greatest flexibility in access and management of installed capacity and over the long run reduces costs for manual interventions to make reconfigurations or changes. The exceptions, if permitted, are for the creation of dedicated "express route" systems that transit one or several geographical sites without passing through the cross-connect facility at that site.

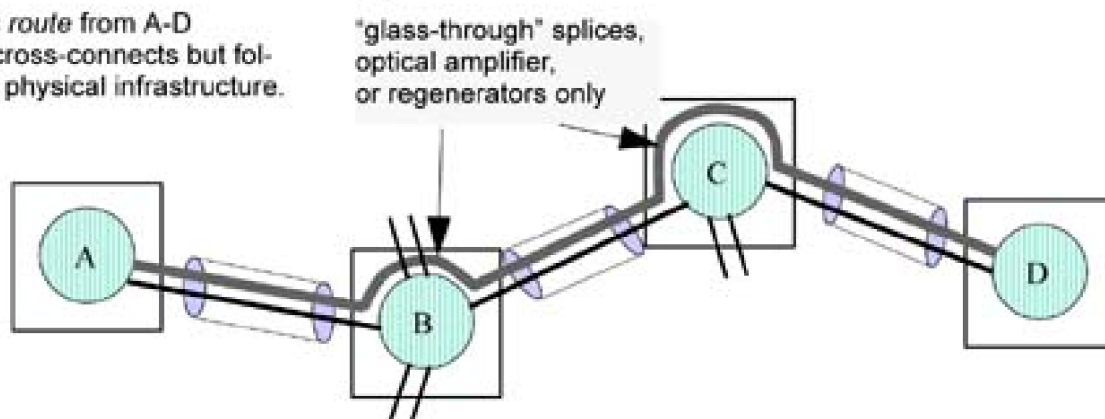
Where policy permits, express routes may be established when a significant enough volume of demand originates and terminates between a single pair of nodes, or when a suitable aggregation of demand shares common intermediate destinations en route, so that a dedicated transmission system can be justified economically for that flow. Typically the action of grooming at a hub site can create these aggregations of co-routed working flows that may warrant an express route. An express route, as the name implies, transports demands directly from one node to a distant node without access at intermediate cross-connecting nodes, although following a route that can be the same as other paths that are cross-connect managed. [Figure 8-11\(a\)](#) illustrates what is called the "fully terminated" capacity model in which all working paths are cross-connect managed on their route through any site equipped with an OXC. In contrast [Figure 8-11\(b\)](#) shows the overlay of an express route through the same nodes on top of a base of cross-connect managed capacity.

Figure 8-11. An express route which bypasses the cross-connect at two nodes en route.

(a) The *fully terminated* capacity model and a cross-connect managed path from A to D.



(b) An *express route* from A-D bypasses the cross-connects but follows the same physical infrastructure.



The economic advantage of an express route is that optical line terminating or optical mux/demux equipment and cross-connect interface ports are not used at the intermediate nodes. The express route through a node can consist of incoming and outgoing fibers that are simply spliced together, or connected through an optical amplifier or regenerator. In DWDM, certain wavelengths or wavebands may be separated by optical filters, passed around the OXC, and remultiplexed with other channels on the outgoing fiber(s). All other wavelength channels interface to, and are cross-connected through, the local OXC. A lightpath or channel is said to be "terminated" on the OXC if it does not bypass the OXC. This does not mean that the path ends at the OXC but only that the lightpath passes through the OXC. Similarly, unused channels are said to be "terminated" on the OXC if they are interfaced to it and accessible and usable by the OXC for cross-connection operations.

The savings that an express route generates in reduced nodal interface equipment is balanced against the cost of providing a dedicated fiber system (or waveband) on the express route and the possibly higher fill levels that the same capacity could support if it was accessible to cross-connect managed routing at every node. A general economic model for express route decisions based on the costs of provisioning the working paths is given in [MaGr97b]. As an example to appreciate the motivation that leads to express routes, assume a DWDM system technology that supports 40 wavelength on a fiber pair, and consider two nodes that are not directly connected in the facilities graph but which exchange a total of, say, 45 lightpaths. Then a dedicated (and completely full) DWDM fiber transmission system could be used to carry 40 of the demands directly between end-nodes without optical mux/demux and interface costs at the OXC nodes en route. The remaining five lightpath demands would not likely warrant another express system and would follow cross-connect managed paths. This example would most likely produce a net savings in equipment cost because the express system is fully utilized but consumes no OXC interface resources en route. A less tangible drawback of creating express routes is, however, a loss of flexibility in network management and the loss of signal monitoring, fault isolation and/or regeneration and other capabilities and functions that the OXC may provide. But another effect—and this is our focus here—is that express routes set up the situation where a single physical cut can escalate to multiple logical span failures.

Traditionally express route decision models have been solely based on considerations about the costs of provisioning the *working* paths alone. One of the reasons why the fully terminated capacity model has often been used for studies of mesh protection or restoration schemes is an assumption that express routes would wind up penalizing the spare capacity. It is an apparently reasonable expectation that express routes would require more spare capacity because:

- Each bypass through a node gives up flexibility in accessing the network's spare capacity for restoration.
- With bypasses present, one cable cut escalates into two (or more) simultaneous logical span cuts.

And it seems reasonable that dual failure situations must require more spare capacity for restoration than the corresponding single failure. For example in Figure 8-11, if the physical duct between nodes B and C is cut then, under the fully terminated model, we face a single logical span restoration problem for span B-C. But in the express route case the same physical failure results in simultaneous logical failures on the physical span B-C and on the logical span A-D between the end nodes of the express route. The total working capacity affected is the same, but the latter problem is multi-commodity in nature whereas the fully terminated environment only produces single-commodity rerouting problems. A central question we consider is whether the two logical span failures in (b) are simultaneously restorable within the spare capacity provided in the baseline SCA design for (a).

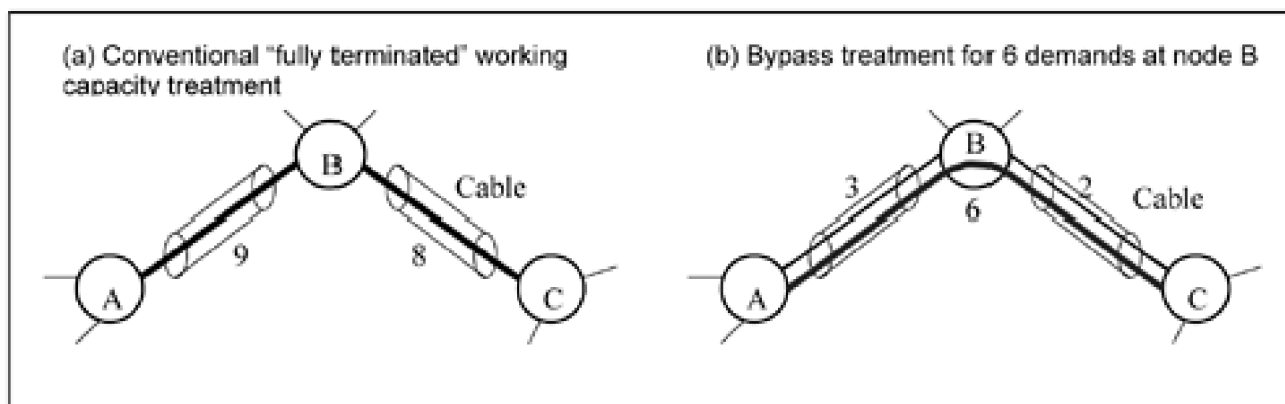
Despite the widely held and seemingly reasonable view that bypassing may thus require more spare capacity, we have found that it never does. Rather unexpectedly, *express route bypasses never require additional spare capacity* and in some cases even permit a reduction in spare capacity. This section is devoted to explaining this and to designing span-restorable mesh networks that are optimized with respect to any of the logical dual span failure situations that arise from the express routes they contain. We start by considering two basic theoretical questions about the effects of bypass. Subsequent sections take advantage of the insights gained to apply them to cost-optimal design of networks containing express routes and nodal bypasses.

8.7.2 Does a Nodal Bypass Require Increased Spare Capacity ?

To address this question let us consider a single physical cut of either span AB or BC in the single bypass situation of Figure 8-12.^[7] In each case, two logical span restoration actions are required simultaneously when the bypass is present. The direct flow between the endpoints of the physical span must be restored at the same time as the express flow between the endpoints of the express route. There are three basic cases to consider depending on how the restoration routes for the two simultaneous logical failures relate to the restoration routes provided in the fully terminated benchmark design.

[7] [Section 8.7.2](#) to [Section 8.8.4](#) include adaptations and extensions of material previously published in [GrLi99](#).

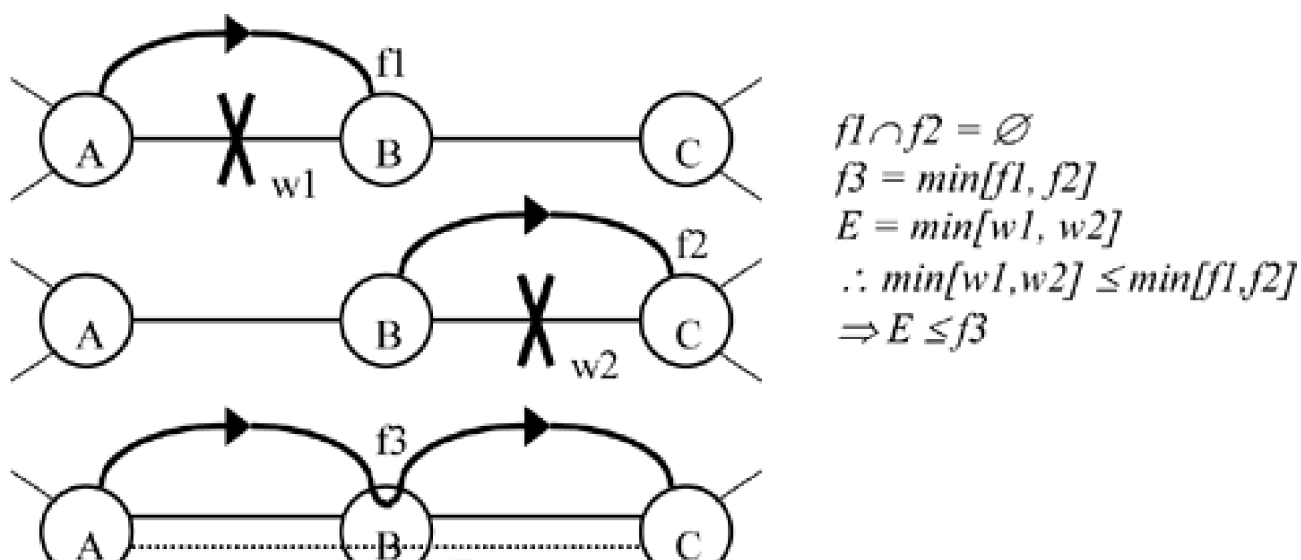
Figure 8-12. Canonical bypass arrangement at node B for case-by-case analysis.



Case 1

In case 1, illustrated abstractly in [Figure 8-13](#), we assume that the restoration flows for the separate span failures AB and BC in the fully terminated benchmark design ([Figure 8-12\(a\)](#)) are span-disjoint. The total flow of these two restoration path sets is represented as f_1 and f_2 , respectively. It follows that if an express route AC bypasses node B within the same spare capacity environment provided to restore [Figure 8-12\(a\)](#), the express demands on logical span AC must always be restorable by routes that are simple concatenations of routes in the benchmark spare capacity design for separate AB and BC restoration routes. For example, when physical span AB is cut with the AC bypass present, a portion of the original restoration routes will protect the lost AB-direct working channels. The AC express demands must also be restorable after this step because at least one option that still exists is for them to simply use the remainder of the AB restoration flow to get from node A to node B. And if the AB and AC restoration flows are span disjoint, then using the spare capacity flow f_1 between AB leaves the spare capacity flow f_2 between BC undiminished. Thus the AC express working flow must be restorable using just the spare capacity of the fully terminated benchmark network. [Figure 8-13](#) includes a formal summary of the argument. f_1, f_2 are the feasible restoration flows in the benchmark SCA design between nodes AB and BC before the bypass, and f_3 is the concatenation of f_1 and f_2 in the bypass design.

Figure 8-13. Case 1: Disjoint AB and BC restoration routes.

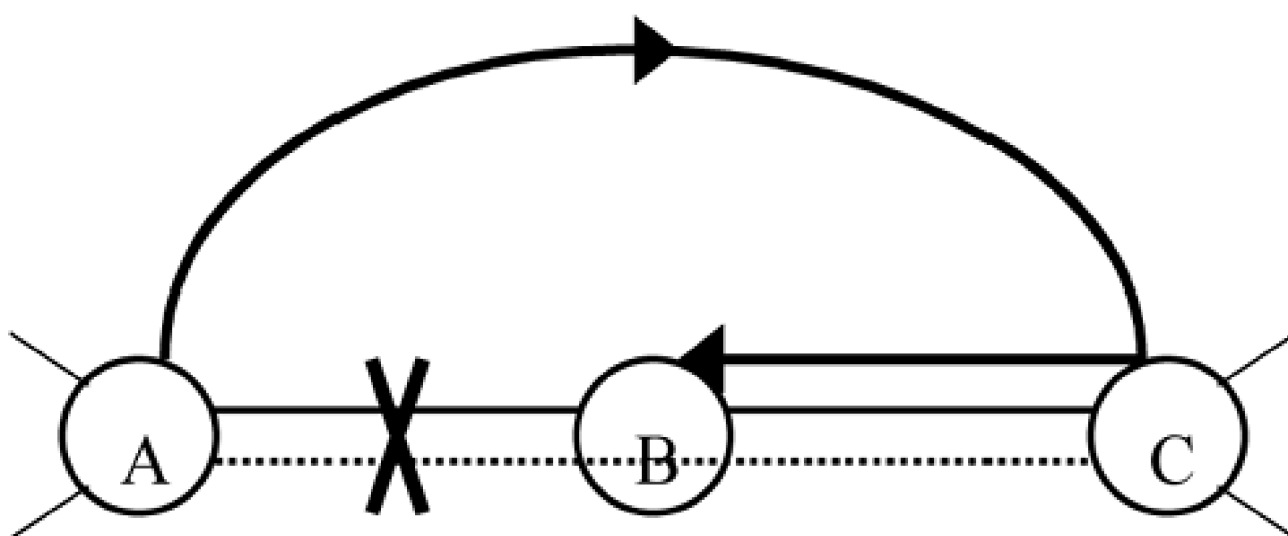




Case 2

The next possibility to consider is that the restoration flows f_1 and f_2 for the non-simultaneous failures of spans AB and BC in [Figure 8-12\(a\)](#) are not disjoint, but that the restoration of span AB in the benchmark design uses routes to node C and, from there, routes back to node B, illustrated in [Figure 8-14](#). In this case it is clear that the failed AC express channels must be restorable simultaneously with the remaining AB direct working flow because the option exists for them to follow the restoration routes that were built into the benchmark design for span AB, but simply to halt at node C en route.

Figure 8-14. Case 2: AB restoration routes travel via node C.

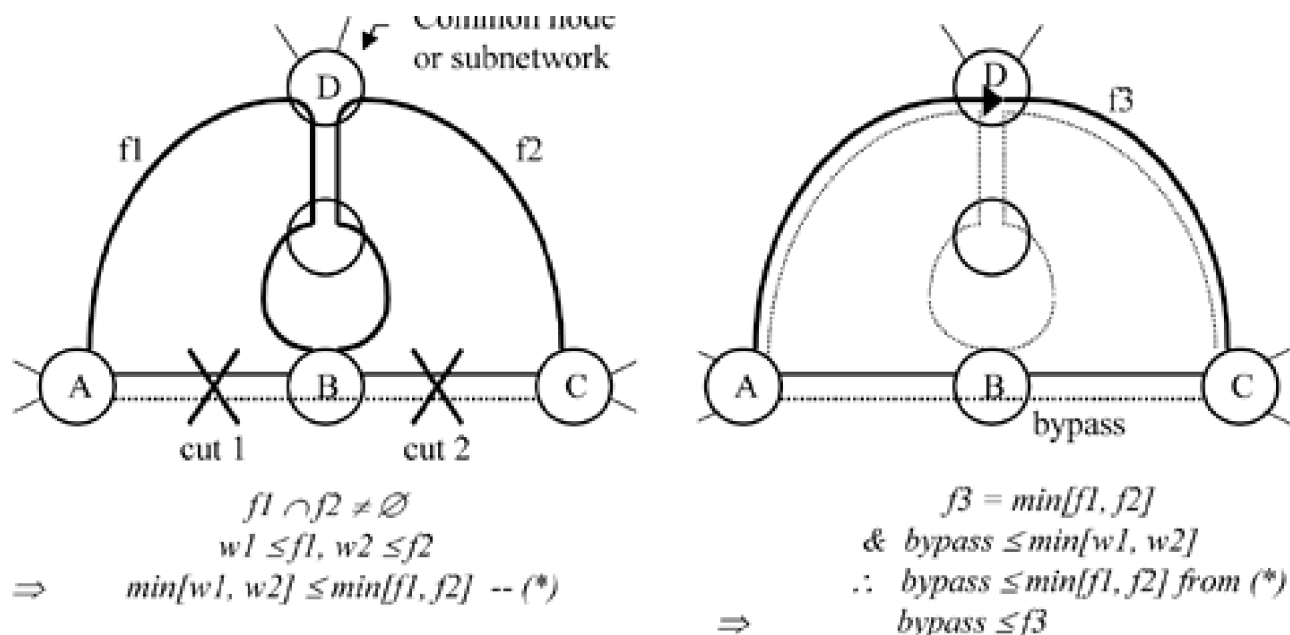


This case also illustrates the first and simplest instance of an opportunity to *remove* spare capacity due to a bypass. This occurs in [Figure 8-14](#) because some of the previously designed-in restoration flow for span AB is now only needed to go as far as node C, not all the way back to node B as is strictly implied for span restoration. In other words, the backhaul or loopback to node B through node C is eliminated in restoration of the AC express demands, so spares may be removable from span BC (i.e., if not needed by wider considerations of restoration flow for other failure scenarios.)

Case 3

Now consider the case where the separate restoration routes for spans AB and BC share some spans in common in the benchmark design, so that their individual restoration path sets are *not* simultaneously feasible within the baseline SCA. This is the most general case and the one where intuition would suggest that total spare capacity might have to increase to deal simultaneously with both of the logical span cuts that arise with the bypass. [Figure 8-15](#) aids us in reasoning about this case. First, if the restoration routes of the non-simultaneous failures are not mutually disjoint, they must meet in at least one common node, which we call D. More generally, the two non-simultaneous restoration flows of the fully-terminated reference case may meet in any subnetwork, D. In [Figure 8-15](#) we represent the general case that over some network regions between nodes A and D and C and D, the flows f_1 and f_2 are disjoint and in another region (between nodes D and B) they are not disjoint.

Figure 8-15. Case 3: Non-disjoint restoration routes.

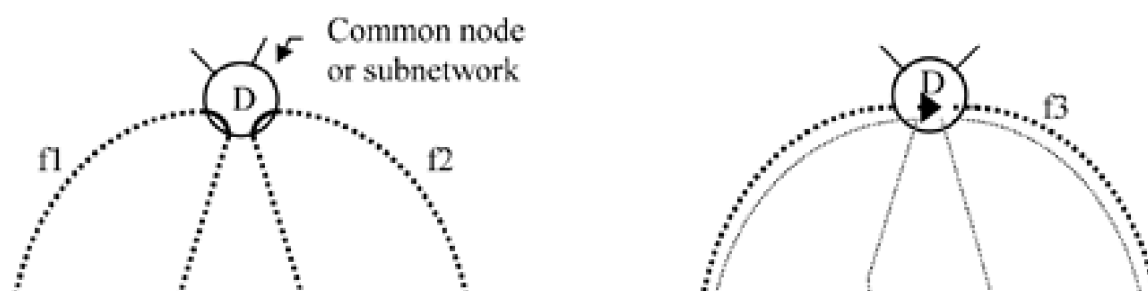


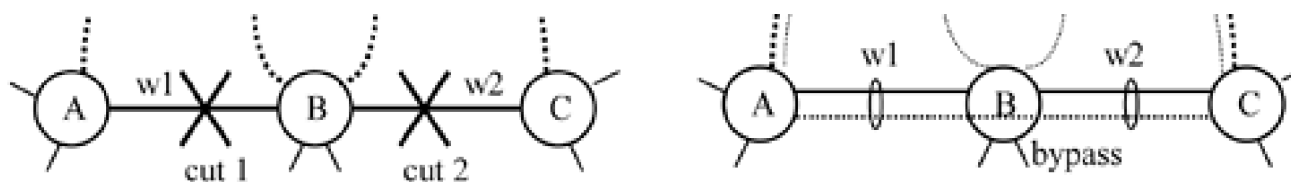
In these circumstances, however, the feasible flow of the restoration routes from node A to node C must still have at least the capacity of the smaller of the two working capacities $w1, w2$ from spans AB and BC from before the bypass. Also, the number of working demands on the AC bypass logical span cannot be larger than this value, because the bypass flow has to be a continuous demand flow through node B. So $w_{AC-bypass}$ must be at most $\min(w1, w2)$. Therefore bypass logical span AC must still be simultaneously restorable with the direct spans of either physical cut, with the reasoning as follows: For cut 1, $f1$ may be fully utilized bringing both the direct and bypass demands to node D. From node D however, only the AB direct demand portion (remaining after the bypass) needs to enter the subgraph between nodes D and B where baseline flows $f1$ and $f2$ are not disjoint and hence not necessarily feasible at the same time. But the bypass amount (for cut 1) must also be less than or equal to $w2$, hence also not greater than $f2$. It follows therefore that the $w_{AC-bypass}$ demands must also be feasible from node D to node C. Because they cannot exceed $f2$, they can use the routes of which $f2$ is comprised in the portion of the baseline spare capacity design that is disjoint from $f1$. They can thereby avoid having to enter the network region between D and B where $f1$ and $f2$ are not disjoint. Furthermore, an opportunity may again exist to remove spares, this time between nodes B and D.

Summary and Generalization

[Figure 8-16](#) is a generalized representation of the complete argument of which the preceding are special cases. Case 1 occurs when node D and node B are the same node. Case 2 is when node D and node C are the same node. And the configuration as drawn is Case 3. These three cases characterize all the ways that a bypass span can relate to the surrounding topology of a fully terminated baseline design. Therefore, these three flow-based arguments constitute a complete explanation of why the single bypass at node B should never require an increase in spare capacity. The last step is to go from consideration of a single nodal bypass at node B to a concatenation of nodal bypasses forming a multi-node express route. The generalization is made by recognizing that all of the above arguments also apply if node B is itself replaced by an arbitrary subnetwork through which the express route demand bundle is continuous, just as it is through the cross-connect matrix at site B. Thus we have a form of proof based on properties that must have been present in the baseline design, that bypasses and express routes *will not require an increase in total spare capacity relative to the fully-terminated SCA design*.

Figure 8-16. The generalized case for reasoning about bypass effects on restoration capacity.



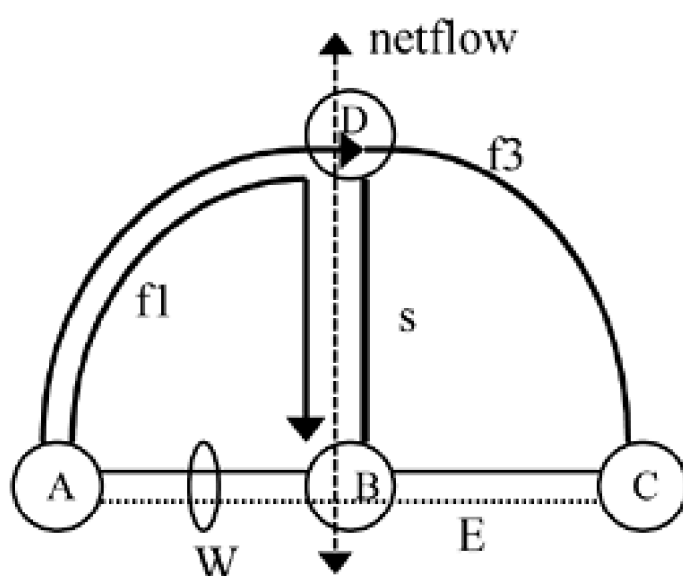


8.7.3 When Does Bypass Yield a Spare Capacity Reduction ?

Previously, we noted in passing that there may be an opportunity to actually *remove* spare capacity in some cases when the bypass is added. One example was in Case 2, where the number of spare channels on direct span BC may, by inspection, be reduced by the size of the AC express route. Also, in Case 3, the spare capacity required to restore direct span AB is reduced by the size of the AC express so the spare capacity requirement on span (or subnetwork) BD may similarly be reduced. However, these considerations are local to the subnetworks drawn in [Figures 8-14](#) and [8-15](#) so a spare capacity reduction is not necessarily assured because the spare capacities of the spans involved may still be dictated by working capacities elsewhere in the network. In other words, a more distant forcing effect may be present.

[Figure 8-17](#) illustrates how the forcer concept ([Section 5.5](#)) can be brought into the considerations. Here we let W be the total working capacity traversing physical span AB. E is the AC express component of the working flow and *netflow* is defined as the maximum amount of restoration flow that span (or subnetwork) BD must carry for its role in restoring all failures in the wider network, outside the ABCD subnetwork. For the following arguments, we consider physical span AB as the failure span in which case W_{AB} is the general W quantity to which we refer in [Figure 8-17](#). Obviously $f1$ must be at least as large as the direct working capacity on span AB which, with a bypass of E units, is $W-E$. At the same time there must be enough spare capacity on span BD to carry the greatest of either *netflow* or $f1$. If *netflow* is greater than W , then the spare capacity on span BD must be at least *netflow*, and there is no spare capacity reduction due to the AC bypass. Similarly, if *netflow* is greater than the remaining AB direct working capacity ($W-E$), the spare capacity on BD must again be at least equal to *netflow*. In these cases, one or more spans outside the ABCD subnetwork are the forcer(s) of span BD, so the spares on BD have to be kept at a higher level than would otherwise be required with the AC bypass. Consequently bypassing at node B to reduce the direct span AB working quantity will not always permit a reduction in sparing at BD. On the other hand if *netflow* is less than W , then span AB must itself be the forcer of the spare capacity on span BD in the baseline design. In this case, as long as span AB is acting as a forcer (i.e., down to $W-E = \text{netflow}$) and the spare capacity on span BD can be *reduced* with every increase in the bypass amount E .

Figure 8-17. Explaining how and when a nodal bypass leads to a net spare capacity reduction.



$$f1 \geq W - E$$

$$s \geq \max[f1, \text{netflow}]$$

Case 1 : if $\text{netflow} \geq W$
then $s \geq \text{netflow}$

Case 2 : if $W > \text{netflow} \geq (W - E)$
then $s \geq \text{netflow}$

Case 3 : if $\text{netflow} \leq (W - E)$
then $s \geq f1 \geq W - E$

Some useful insights arise from these considerations:

- a. A bypass produces a global reduction in spare capacity requirement whenever the effect of the bypass is to reduce the working quantity of a forcer span. This remains true for subsequent reductions until the span is no longer a forcer.

- b. The spare capacity decreases on each other span for which the bypass span takes on flow that was previously on the direct forcer span. As the bypass amount increases, another span will emerge as a new forcer, eliminating further sparing reduction from bypass on the selected span.
- c. It follows that the most effective opportunity for bypassing to reduce sparing is at a node where demand continuity is found between two spans, which are both forcers. In these circumstances each unit of bypass simultaneously reduces two forcer spans.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

8.8 Optimal Capacity Design with Bypasses

Let us now apply the above insights about the role that forcers play in a capacity design strategy that allows for any number of potential bypasses. A first step is to develop the formulation for SCA with a set of bypass candidate locations. This step finds a placement of spare capacity that is minimal in the sense that it sizes the candidate bypasses to achieve a maximal reduction of spare capacity through the principle of forcer reduction by bypass. By itself, however, this does not maximize the port savings obtainable because there are a number of bypass combinations that yield the same minimum spare capacity, but which do not necessarily use the fewest OXC ports. A second stage optimization therefore maximizes the total bypass quantity using the total spare capacity of the first stage design as a constraint.

8.8.1 Minimum Spare Capacity Given a Set of Express Routes

In this problem the basic SCA model for single span cuts in a fully terminated mesh network is our starting point. To this we assume an overlay of express systems whose routes, end-nodes, and working demand flows are already known and given as an input. The problem is to optimize the spare capacity assignment so that all single physical span failures are fully restorable while recognizing that each physical span cut may now create a multiple logical failure scenario. These multi-span scenarios involve a portion of working flow to be restored between the custodial nodes of the physical span and working flow on logical spans that have to be simultaneously restored between the end-nodes of their express routes. Our aim is to find the minimum allocation of spare capacity and the restoration flow assignments so that all demand flow is restorable under each composite logical failure scenario. Express routes are equivalently referred to as *logical* spans. In the solution only physical spans are allocated spare capacity, implying that spare capacity is always accessible at each cross-connect at the ends of the physical span on which it is placed (although the working capacity on an express route still bypasses the OXC nodes). To represent the network state with express routes as input to the problem we define:

- S = the set of physical spans, index i or j for failure span or other span contexts respectively.
- B = the set of express routes, index r .
- w_i = the amount of working flow crossing a physical span i (terminated at the physical span end-nodes).
- w_r = the amount of working flow crossing express route, r (terminated at the end-nodes of the corresponding logical span).
- $R(i)$ = the set of logical spans arising from express routes that are routed over physical span i . Each set $R(i)$ is individually indexed by r .
- P_i = the set of routes eligible for restoration of physical span i , index p .
- P_r = the set of eligible routes for use in restoration of logical span r , index b . These route sets, being defined once only for each logical span, may include routes that in some circumstances cross a physical span that is the cause of their failure. A specific prohibition of express route restoration flow from using the current failure span is therefore required.
- $\delta_{i,j}^p = 1$ if the p^{th} eligible route for restoration of span i crosses span j , zero otherwise.
- $\delta_{r,j}^b = 1$ if the b^{th} eligible route for restoration of logical span r crosses span j , zero otherwise.

The variables are:

- s_j = spare capacity assignment to physical span j , (integer).

r^p

f_i^p = restoration flow assignment to the p^{th} eligible route for restoration of physical span i .

- $f_{i,r}^b$ = restoration flow assignment to the b^{th} eligible route for restoration of logical span r when physical span i failure occurs.

Both types of restoration flow variable are strictly integer but as is common practice may be relaxed as long as spare capacity variables and working capacity inputs are integer. The design model is:

SCA_ER1

Minimize

Equation 8.38

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

- Restorability of physical spans:

Equation 8.39

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

- Restorability of co-failed express routes:

Equation 8.40

$$\sum_{b \in P_r} f_{i,r}^b = w_r \quad \forall r \in R(i) \quad \forall i \in S$$

- Spare capacity for each set of simultaneous physical and logical span failures:

Equation 8.41

$$s_j \geq \sum_{p \in P_i} \delta_{i,j}^p \cdot f_i^p + \sum_{r \in R(i)} \sum_{b \in P_r} \delta_{r,j}^b \cdot f_{i,r}^b \quad \forall (i,j) \in S^2 | i \neq j$$

- No express route restoration flow on the failed span:

Equation 8.42

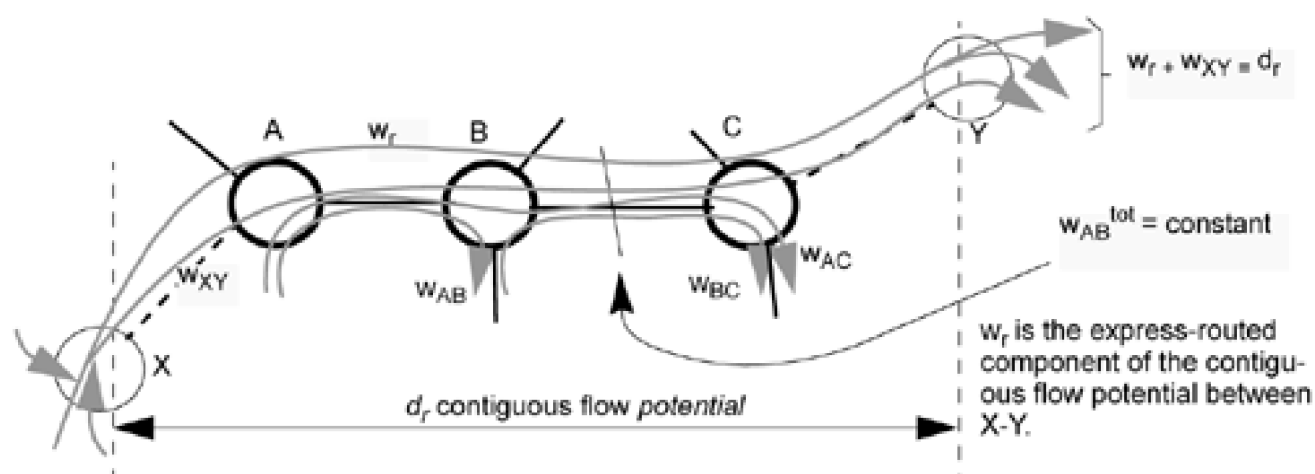
$$\sum_{r \in R(i)} \sum_{b \in P_r} \delta_{r,i}^b \cdot f_{i,r}^b = 0 \quad \forall i \in S$$

This model allows express routes to use different restoration routes depending on where along their path the physical span failure occurred. This is analogous to failure-dependent path restoration between the end-nodes of the express routes and is the reason the last constraint system is needed. Because the eligible route set for restoration of an express route is not excluded from including portions of its own route, such routes must be excluded from use in a failure-specific way by Equation 8.42. For physical spans this is implicit because their eligible route sets always exclude the span itself. A model of lower dimensionality is obtained, and the latter constraint system is unneeded, if the eligible route set for restoration of an express logical span are defined to exclude all spans on the normal working route of the logical span. Then the express route restoration flow variables lose their dependence on i .

8.8.2 Allowing the Model to Dimension the Express Routes

In SCA-ER1 the partitioning of the total working flow crossing a span into a terminated w_i component and a bypassing w_r component that is part of an overlying express route is given as an input. We know in principle, however, from the theoretical considerations about bypass, in particular Section 8.7.3, that the potential for spare capacity reductions due to bypass depends on getting the right balance between local and express flow. When this split is right we are balancing local with wider-scale network forcer effects, allowing minimization of total spare capacity. On the other hand, despite the amount of forcer reduction potential, the amount of working flow referred from a physical span to an overlying logical span cannot exceed the amount of contiguous demand flow that is really present. This is illustrated in Figure 8-18.

Figure 8-18. Working flow referred into a bypass cannot exceed the contiguous flow potential.



Here we show an express route between end nodes X-Y. The route transits nodes A, B, C, through which numerous other demands are also routed. From a purely numerical standpoint we might like to set w_r as high as $\min(w_i^{tot})$ where $w_i + w_r = w_i^{tot}$ and i indexes the spans along the express route. Indeed this would generally show an improvement to SCA-ER1 objective values. However, one cannot actually refer more working flow into an express route than is available as a continuous bundle of demand, unaltered in its composition anywhere along the express route. This is why we define the express flow potential d_r which is the amount of demand that flows contiguously between the end nodes of the express route. Note that d_r is by nature like a demand matrix quantity, but it is not simply the demand from X to Y. It is the aggregation of all network flows (including X-Y demand), that are routed as an intact bundle over the path segment from X to

Y. [Figure 8-18](#) shows how spans AB and BC have a working capacity composition that arises from many overlying routes of flow. Part of what flows over them is d_r units of demand routed intact all the way from node X through to node Y. This is the maximum amount of working flow that could thus be referred into an X-Y express route. As drawn, the amount w_r , which is less than the full potential d_r , is actually referred to the express route. The total X-Y flow is still d_r but it is realized partly as express flow w_r and partly by conventional terminated routing, w_{XY} . Whatever the split between terminated or express routing the total working capacity over each span is unaltered.

Therefore, to solve for the optimal dimensioning of the express routes themselves, we need the added input parameters:

- d_r = the maximum amount of working flow in express router (integer).

and the additional variables:

- w_r = the amount of working flow assigned to express router (integer).
- w_i = the amount of ("fully terminated") working flow on physical span i (integer).

Although we are solving for the w_i values now, this is not a joint formulation. All demand is still routed over its shortest paths, producing what we now call w_i^{tot} quantities, which are input parameters produced before these problems are attempted. It is only the partitioning of w_i^{tot} on each span into express or terminated portions that we are solving for. For clarity, the previously denoted w_i in SCA-ER1 now denote the *terminated* working capacity on the physical span i , and w_r denotes the express flow which is bypassed through the node. If there is only one express route overlying the span $w_i + w_r = w_i^{tot}$. But more generally several express routes, indexed by r , could overlie any one physical span. The design model thus becomes:

SCA_ER2

Minimize

Equation 8.43

$$\sum_{j \in S} c_j \cdot s_j$$

subject to:

- Restorability of physical spans: [Equation 8.39](#)
- Restorability of co-failed express routes: [Equation 8.40](#)
- Spare capacity for simultaneous restoration flows: [Equation 8.41](#)
- No restoration flow on the failed span: [Equation 8.42](#)

plus:

- Partitioning:

Equation 8.44

$$w_i = w_i - \sum_{r \in B} o_{r,i} \cdot w_r \quad \forall i \in \mathcal{N}$$

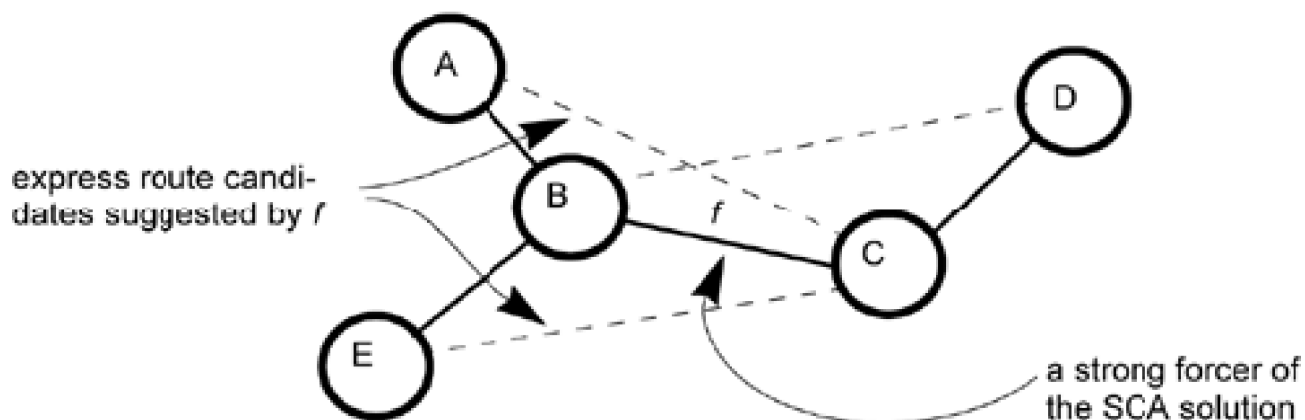
- Express flow limits:

Equation 8.45

$$w_r \leq d_r \quad \forall r \in B$$

Note that because $w_r = 0$ is a valid outcome for any given express route, the model not only dimensions the express routes but also selects among a set of express route candidates that we provide as inputs. In effect all express routes that we represent in the set B for SCA-ER2 are implicitly treated as *candidate* express routes. If their associated $w_r = 0$ they are rejected as express routes. Obviously, however, there are a vast number of potential express routes and it is impractical to represent the complete set in most problems. A practical approach must therefore have some *a priori* way to decide upon good candidates for express routes. In this regard, we know from above that to reduce spare capacity a set of single node bypasses based on the list of network forcers is the most direct approach because it is only through forcer reduction that a bypass can yield a reduction in network sparing. Thus, forcer analysis is a tool to accompany SCA-ER2. We can identify good potential bypass locations based on the forcer strength of spans in the baseline SCA design. An example of how forcers can generate a selective list of single node bypasses (e.g., 2-hop express routes) to represent in B is illustrated in [Figure 8-19](#).

Figure 8-19. Candidate bypasses generated by forcer analysis of the fully-terminated design.



The principle is that each express route represents an option that SCA-ER2 might find advantageous to reduce the forcing effect of span f by referring part of the w_i^{tot} out to a new logical span candidate, such as A-C or B-D in the example. Obviously promising candidates for express routes from this forcer-relieving standpoint also include three-span express routes bypassing a pair of strong forcers sharing a node in common, and so on. Note that for each candidate express route the corresponding d_r limit must be separately worked out from consideration of the contiguous working flow between end nodes of the candidate express route. Thus, while there may be other principles on which to propose express route candidates for consideration in SCA-ER2, routes that comprise many forcers and have high contiguous working flow potential d_r are among the most likely to yield express routes that are highly effective in reducing spare capacity under from SCA-ER2.

8.8.3 Maximum Port Elimination by Bypass at Minimum Spare Capacity

So far in our considerations of express routes in SCA-ER1 and SCA-ER2, we have been focused on the benefit that express routes can yield in terms of spare capacity reduction relative to a fully terminated design. But the historical motivation for express routes was largely

based in interface cost savings from a working capacity standpoint. It is of interest therefore to consider how to further refine the express route designs to consider the total cost of both protection capacity and interface costs in making the assignments of working flow to express route candidates. To do this, we can take the hop count of each candidate express route into account and recognize that for each unit of working flow that is referred into an express route, we save two cross-connect interface ports for each node bypassed (and some pro-rated portion of the OXC core costs) relative to the fully terminated SCA design. Thus, we need to add to our model the parameters:

- H_r = the number of hops on the r^{th} express route candidate
- c_t = the cost associated with terminating one channel at cross-connects *atboth* its ends.

SCA_ER3

Minimize

Equation 8.46

$$\left(\sum_{j \in S} c_j \cdot s_j - \sum_{r \in R} w_r \cdot H_r \cdot c_t \right)$$

subject to:

- Restorability of physical spans: [Equation 8.39](#)
- Restorability of co-failed express routes: [Equation 8.40](#)
- Spare capacity for simultaneous restoration flows: [Equation 8.41](#)
- No restoration flow on the failed span: [Equation 8.42](#)
- Partitioning of total working capacity: [Equation 8.44](#)
- Express flow maximums: [Equation 8.45](#)

This does not actually add any new variables to the problem. Although the objective function has two terms, it is not bicriteria in nature because each term here has the same units of cost. SCA_ER3 differs from ER-2 in that it has an added benefit to putting even more of the potential capacity onto the corresponding express routes. By itself SCA-ER2 only tends to push forcers down to their forcer threshold. Below that there is no further benefit to the spare capacity cost objective of SCA-ER2. But in some cases, even below forcer threshold, further increases in the bypass amounts (up to at most the d limit) can further to reduce port counts, without impact on total sparing. The trick is to maximize the bypass quantities without the bypass spans themselves emerging as new forcers that start increasing the spare capacity again, or if the latter happens, the net savings in port costs still outweighs the added spare capacity. At this point the total cost of ports and spare capacity are minimal.

As expressed, [Equation 8.46](#), could take in negative values, especially if the port cost, c_t , is high relative to the spare channel costs, c_j . This is not a problem, however, because the objective is saying simply "minimize the cost of spare capacity less port cost *savings* due to express routes." This form is more useful for showing the logic of what is involved relative to the prior capacity-only minimization model, and makes it clear why we would want to shift even more working flow into express routes (for port savings) if possible without producing an offsetting increase in spare capacity (due to creating new forcers out of the express routes). The objective could equivalently be written as:

Minimize

Equation 8.47

$$\left(\sum_{j \in S} c_j \cdot s_j + \sum_{r \in B} \sum_{j \in B(r)} \left\{ w_j^{tot} - \sum_{r \in R(j)} w_r \right\} \cdot c_r \right)$$

where $j \in B(r)$ denotes the spans in the r^{th} express route candidate and $R(j)$ (previously defined) is the set of express route candidates that overly span j . This form restricts itself to expressing only absolute cost, not a mixture of cost and savings relative to an implied fully terminated reference model. The second term is a direct expression of the port costs required to terminate the working channels on each span that are not referred into any overlying express route.

In cases where channel costs are more significant than port costs, i.e., $c_j \gg c_t \forall j \in S$, SCA_ER3 can also be approximated in a two-stage approach. The first is an SCA_ER2 problem that disregards port cost savings and makes express route decisions to minimize spare capacity cost alone. The second step is a secondary problem to minimize port costs without using any more spare capacity than produced by SCA_ER2. The rationale is that the second step obtains further port savings by referring increased quantities of working capacity into the express routes, but not to the extent of causing any new express route forcers that would go as far as to increase the spare capacity. We define:

- C^* = the total cost of spare capacity (objective function value) from SCA_ER2.

and a new subproblem we call *minimum port costs at a given spare capacity, MPC(C*)* is:

MPC(C*)

Minimize

Equation 8.48

$$\sum_{r \in B} w_r \cdot H_r \cdot c_r$$

subject to:

- Restorability of physical spans: [Equation 8.39](#)
- Restorability of co-failed express routes: [Equation 8.40](#)
- Spare capacity for simultaneous restoration flows: [Equation 8.41](#)
- No restoration flow on the failed span: [Equation 8.42](#)
- Partitioning of total working capacity: [Equation 8.44](#)
- Express flow maximums: [Equation 8.45](#)

plus:

- No increase in total spare capacity:

Equation 8.49

$$\sum_{j \in S} c_j \cdot s_j \leq C^*$$

The objective function takes on a maximization form because it is expressed in terms of the number of ports *not* needed in the design relative to the fully terminated design. Note that it is only the total spare capacity cost that we import from the prior problem, we do not assert limits on each span's spare capacity individually from the prior problem because we want to permit selection among capacity-equivalent solutions that are compatible with larger express flow assignments that reduce port counts but have no deleterious effect or benefit on spare capacity.

8.8.4 Sample Results

A set of results for SCA-ER2 and the two-stage approximation for SCA-ER3 was obtained for five test cases using four network models. This allows us to demonstrate the two cost saving principles separately in the results, without assuming a specific relative cost of ports to spare channels. In other words, we first obtain results with SCA_ER2 to observe the spare capacity reducing potential of express routes. We then use *MPC()* at the same total spare capacity to see what further *portcount* reductions are technically possible at the stage. (Under SCA_ER3 the number of such port reductions that are actually claimed always depends on the relative cost of ports to channels). Net 1 is the US metropolitan area model in [Figure 5-10](#). This network is complete with a demand matrix provided in [Bell93](#). Networks 2 and 3 were obtained by modifications to Net 1 to provide a sample of four test cases with a variety of characteristics. Net 2 was created by eliminating three direct spans between the end nodes of any pair of adjacent forcer spans in Net 1. This was done to reflect the practical reality that if a direct span exists, it wouldn't make sense to send the demands via a longer indirect route even if the nodes on the latter route are bypassed. This is because a direct span has two terminations and minimal channel cost. The alternate route can only have greater channel cost and still has at least two port costs (at its ends), even if the entire route is an express. Thus, express route candidates are only defined when shorter direct routes are not an option. This creates a test case where the potential benefit from the express route design is greater than in Net 1 and is also representative of several real networks that do not have as high an average nodal degree as Net 1. Whereas Net 1 has average nodal degree of 4.2, Net 2 has a $d=3.6$. Net 3 is another modified version of Net 1 where 3 more nodes and 5 spans were added to provide a fourth test case with average degree of 4. Net 4 is an unrelated planning model that is based on forecasted topology and demands by a large U.S. inter-exchange carrier.

In all networks, the w_j^{tot} values for each span were generated by mapping the demand matrix onto the topology by shortest path routing. For the first four results the bypass candidates were located only at those nodes where adjacent forcer spans met in the benchmark SCA design. This systematically defines a set of bypass nodes to consider and keeps the problem size suitably small to be practical. In a fifth test case, Net 1 and its original demand matrix were again used, but in a context of "full express" routing of the working demands. This means that the demand flow for each demand pair follows the same working routes as before, but all transit nodes en route are bypassed. In other words, demands are only terminated on a cross-connect at their origin and destination nodes. This is the "Net 1—full express" test case. A practical implication of this class of network is virtually unlimited escalation of one physical cut into a number of distinct simultaneous logical cuts. It is included, however, primarily to observe its effect on the spare capacity minimization and to get an indication of the maximum possible port savings.

Results are summarized in [Table 8-6](#). The total spare capacity was reduced by 12% on average over all test cases. At the minimum spare capacity, there was a further reduction on average of 16% possible in the total network port count by *MPC()*. In Net 2, which was the sparse version of Net 1, using the same demand matrix as Net 1, just six bypass arrangements yielded a 27% reduction in spare channels at the same time as a 21% reduction in total port counts was obtained. In the full express test case total spare capacity was reduced only 2.4%, although the greatest of all port count reductions, 38%, was obtained. These results demonstrate that significant savings can be had by optimized design of express route bypass arrangements.

Table 8-6. Results of SCA_ER2 followed by MPC(C*)

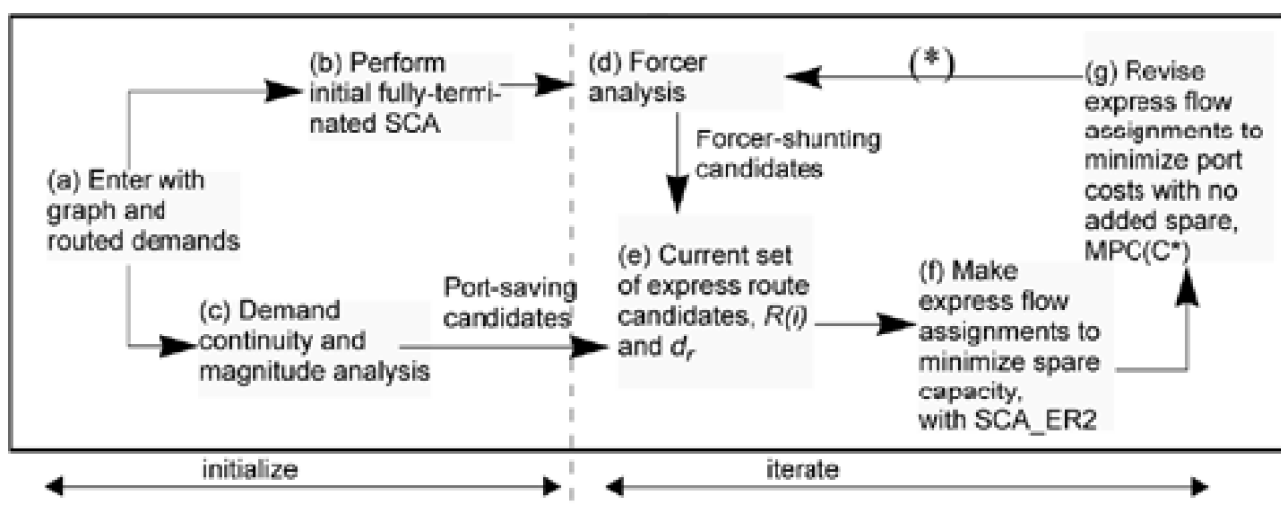
Test Case	Network	Reduction in Spare (%)	Reduction in Port Count (%)	Number of Bypass Candidates
1	Net 1	4	9	5
2	Net 2	27	21	6
3	Net 3	10	19	14
4	Net 4	7	15	19
5	Net 1- Full Express	4.5	38	All possible bypass candidates

In inspecting the results for Nets 1-3, which are sparser versions derived from Net 1, we see that the sparing and port count savings were greatest in Net 2. Net 2 had the lowest average nodal degree and was conditioned to conform to the realistic situation that a meaningful bypass location is not shunted by a direct physical span available for the same demands. This ranking is consistent with expectations from first principles that bypassing should tend to be most effective in low degree networks. This is because strong and more irregular forcers tend to arise on sparse graphs, hence express routes can be more effective in leveling out these forcers. The same demand patterns routed over a higher degree network are likely to yield more leveled flow accumulations on spans and have fewer strong forcers. This not only tends to make more of the spans present into forcer spans in a low degree network, but also more often sets up the nodal conditions of transiting flows that are good candidates for a bypass arrangement. Net 1, which had the highest degree, also had the lowest benefit from bypass design, consistent with this general view. Net 3 had higher degree than Net 2, and exhibited savings that similarly follow the trend based on classification by nodal degree. Net 4, being a completely separate test case, is harder to compare to the family of Net 1-based test cases. The results of the fifth test case, Net 1—full express, can be interpreted as indicating that there is little further relief on sparing relative to that obtained with the five "most promising" express route candidates in Test Case 1 but there is maximum possible relief is obtained on port counts.

8.8.5 Iterative Optimization of Express Routes

A practical limitation of SCA_ER2 on large networks is the number of express route candidates that can be represented. An option in this case is to iterate SCA_ER2 and MPC(C*) in a loop that includes forcer analysis and demand continuity analysis to update and improve the set of express route candidates at each iteration. As it stands, all of the models above require a given set of express route candidates as inputs. Iteration allows the set of express route choices to evolve in conjunction with improving forcer-shunting spare capacity savings and working path port count savings. [Figure 8-20](#) illustrates the overall approach.

Figure 8-20. An iterative strategy for optimizing express routes, spare capacity and port costs.



There are several variations within the basic strategy. One is to solve the MPC stage with some allowance for a slight increase in spare capacity, i.e., solve $MPC(C^* + e)$. As in most meta-heuristics this provides a mechanism to take a step "backward" now and then to escape from a local minima and explore a direction that reduces port costs more than spare capacity costs. The result would be kept as an improvement if the net savings in port costs exceeded e , redefining C^* for the next iteration to become $C^* + e$. Similarly a certain fraction of new express route candidates can be injected at random in each iteration, and variations in the basic working path routing plan can be introduced periodically, for a restart at (a). In all cases, the lowest cost design seen at the point (*) is recorded as the least cost composite strategy involving express routes. The resulting network is fully restorable to any single physical span cut and is cost minimized in terms of both the spare capacity needed and the costs for working channel ports on cross-connect nodes.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

8.9 Effects of Dual Failures Arising from Shared Risk Link Groups

We have seen that express route arrangements lead to a circumstances where a single physical cut can escalate to multiple logical span failures, although no more spare capacity is required than in the fully terminated case. An SRLG presents a somewhat similar set of circumstances where a single duct or other single physical failure leads to simultaneous failure of two or more *physical* spans, and *always* requires additional spare capacity relative to the non-SRLG case. The classic example of SRLG circumstances is a bridge crossing ([Section 1.5.4](#)). As mentioned in [Section 1.5.4](#) the SRLG concept needs a slight refinement to reflect the issue as it pertains to span-restorable networks, as opposed to SBPP. ^[8]

^[8] Collaboration with J. Doucette. While under development for the book, parts of this section were presented at [DoGr02b](#).

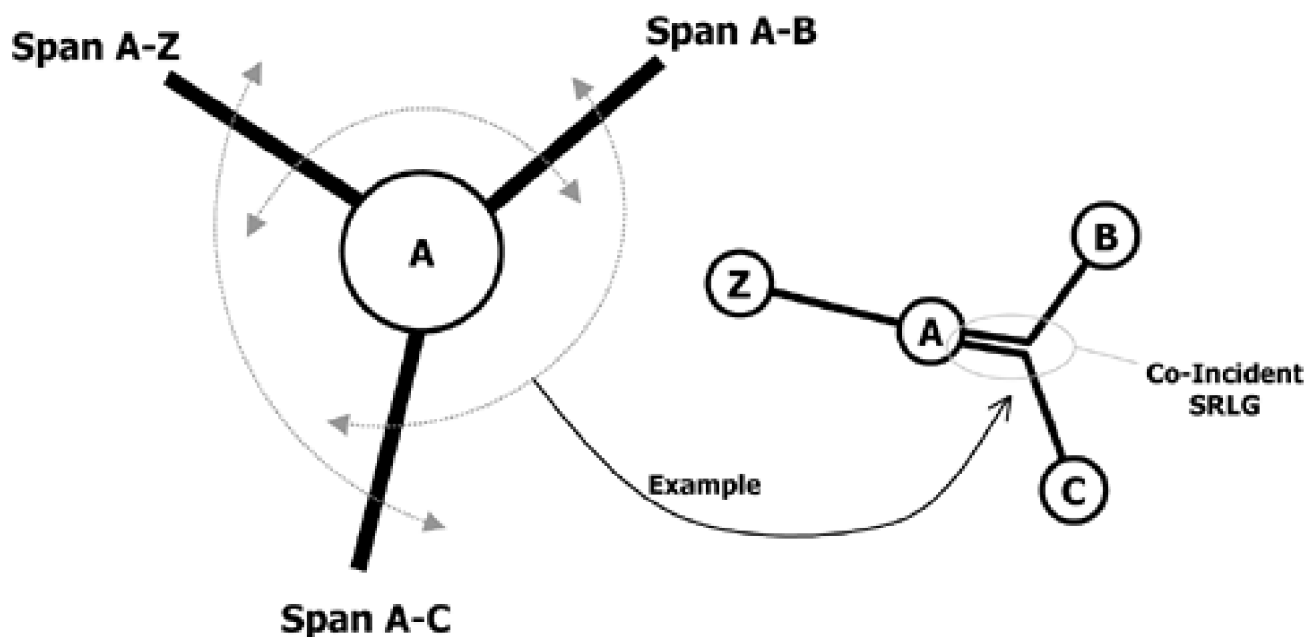
Formally, under the SRLG concept every cable sheath represents an SRLG; one which groups all the lightpaths that traverse it. This is a necessary consideration under SBPP type of provisioning and protection arrangements where paths may share common risks arising from every cable along their routes. In the context of span restoration, however, each cable or duct between neighboring OXCs has always been the relevant failure element and is assumed to fail completely, affecting all channels at once. In the SR context this is treated as a single failure. Such "ordinary" span failures are actually called "default SRLGs" under the more recent SRLG framework. But the treatment of every cable as an SRLG is not really a helpful refinement to span-restorable network design: it just renames what has always been the default failure model. The general notion of fault multiplication due to physical commonality certainly still applies to SR networks, however, but it is of special concern only at the level of *whole spans* that share a common risk. Hence what is more relevant to span-restorable design could be called a shared risk *span* group (SRSG) or a "span SRLG." *Aspan SRLG* is a specific type of SRLG situation where two notionally separate physical spans actually share a common physical risk. By the nature of span restoration, it is only *span* SRLGs that require any special attention. The "default SRLG" failure scenario is the norm in a span-restorable design and is fully captured in the concept of a span and span failure itself.

Our interest now is to consider how various numbers and locations of two-span SRLGs affect the spare capacity requirements of a span-restorable mesh network. An aim is to see if guidelines or insights are possible to understand which SRLGs are the most deleterious to capacity efficiency. Such understanding would help identify SRLGs that may be worth eliminating through physical rerouting efforts. Generally such physical layer rearrangements can be expensive to undertake, so it makes sense to know which few SRLGs are the most strategic to target for elimination. In addition, it is helpful to understand how a network that is designed to cope with an arbitrary number of SRLGs relates to a network that is explicitly designed for complete dual failure restorability. Finally we would also like to get an appreciation of how much coverage against possibly hidden or unknown SRLGs is obtained for any given level of total capacity investment above the basic allocation for single failure restorability.

Coincident Span SRLGs

Our focus is on SRLGs on spans incident to a common node. This particular class of SRLGs is understood to be the most common in practice, and we refer to them as [coincident SRLGs](#). [Figure 8-21](#) illustrates the three coincident (two-span) SRLG combinations at a degree 3 node. An SRLG involving two spans that are not incident on a common node are referred to as a *non-coincident SRLG*. An SRLG of either type, unspecified, is called a *general SRLG*. A *dual failure per se* is more general than an SRLG. A dual failure is realized by any event where two spans of the transport network are in a failed state simultaneously. This includes completely independent failures that happen to have overlapping repair times but do not arise from a common-cause failure. Thus, the set of all dual failures includes all SRLGs, while coincident SRLGs form a yet more specific subset of span SRLGs.

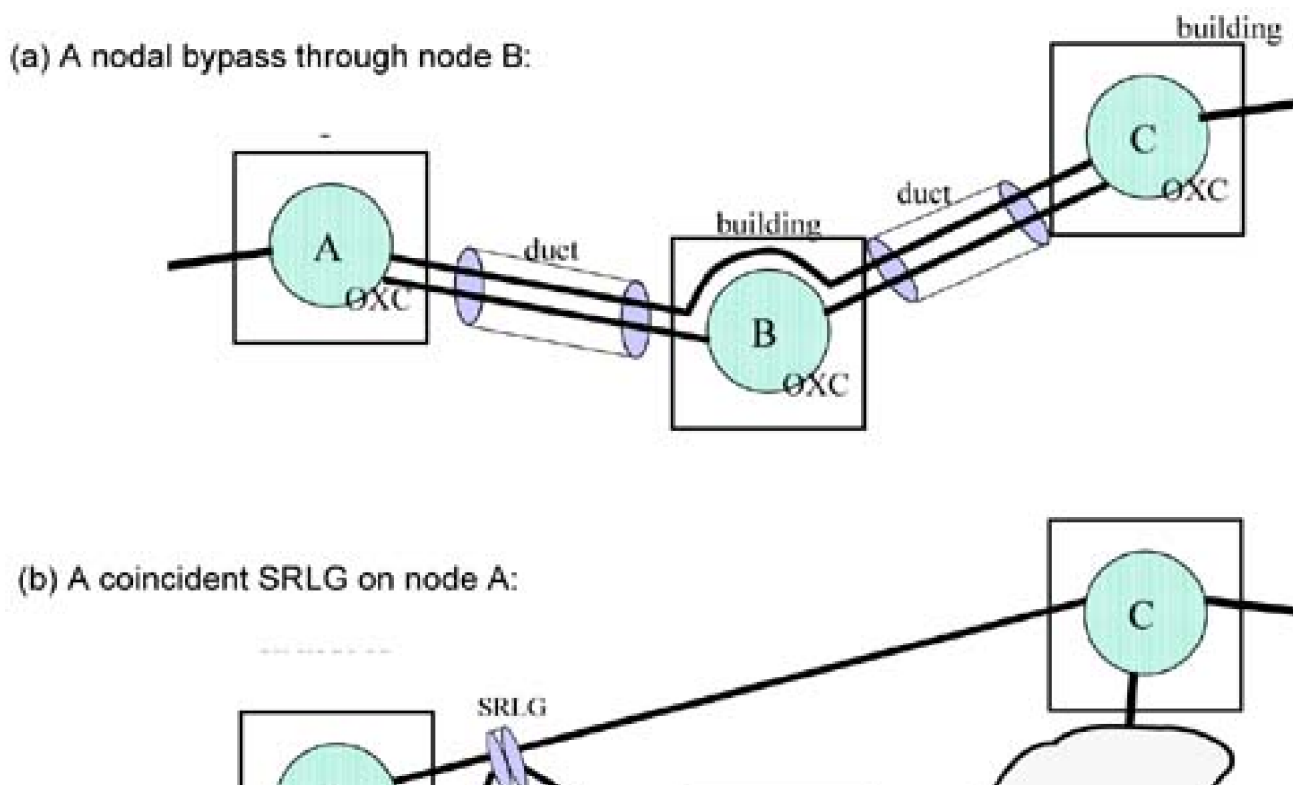
Figure 8-21. The three possibilities for coincident span SRLGs at a degree 3 node. A degree 4 node has six possible span SRLGs.



8.9.1 Comparing Span SRLG and Bypass Dual-Failure Situations

We have seen that the nodal bypass arrangements arising at nodes along an express route, give rise to a type of dual span failure situations. And yet they never require any additional spare capacity to retain restorability. At first glance, the coincident SRLG situation seems similar. And yet SRLGs generally *do* have a deleterious effect on spare capacity requirements, relative to the situation without the SRLG. Let us now pinpoint the subtle but significant differences between a bypass-related dual failure and that of a coincident SRLG that explain their different effects. To help with this, [Figure 8-22](#) details the physical situation for the comparison.

Figure 8-22. Explaining the differences between a nodal bypass and a coincident span SRLG.



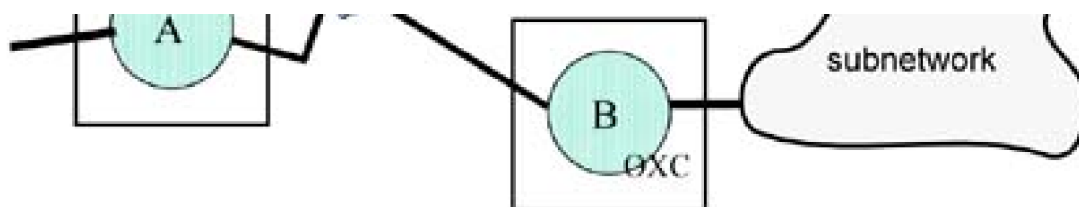
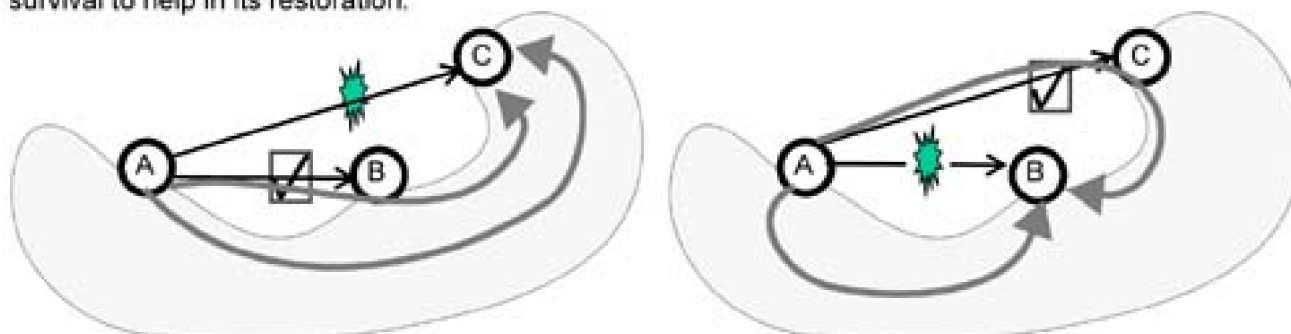


Figure 8-22 (a) shows a nodal bypass arrangement in which one or more wavelength channels bypass the OXC at node B, but follow the same ducts or cables as other channels on spans A-B and B-C. In general several nodes may be bypassed in this way along a single express route. In contrast **Figure 8-22** (b) shows a coincident SRLG situation. If span A-B is cut in (a) or if the SRLG is cut in (b), the resulting two-failure network state may seem at first to be almost identical. Both produce simultaneous logical span failures on nodes (A-B) and (A-C). So how can statements that "bypasses never hurt" but "SRLGs always hurt" both be true? It seems like a paradox.

The first key to unravelling the confusion is to realize that the comparative statements about the relative capacity impacts are not set within the same physical graph. To reason about the effects of an SRLG against the corresponding fully terminated design without SRLGs, we have the situation as abstracted in **Figure 8-23**. The similar abstraction for the bypass case is shown in **Figure 8-24**. In the SRLG case both A-B and A-C are *physical* spans and the question of hurt or help is about having dual failures on these two physical spans, relative to the case of design for either as a single failure. In contrast the question about bypasses is asked in a subgraph context where only spans A-B and B-C physically exist, so the only failure scenarios to be compared are A-B alone (non-bypassed) versus A-B plus A-C (the bypassed case). The scenario of (A-C) as a single failure with (A-B) *surviving* does not exist in the comparative context of reasoning about bypasses, but it does exist in the comparative context of the SRLG question.

Figure 8-23. Why SRLGs always require more spare capacity in a survivable design.

(a) Fully terminated (basic SCA) case with physical spans (A-B), (B-C): Each individual span failure enjoys the other's survival to help in its restoration.



(b) Coincident SRLG: The two separate single-span failure cases in (a) occur together forcing restoration flows to be supported through the remaining network only.

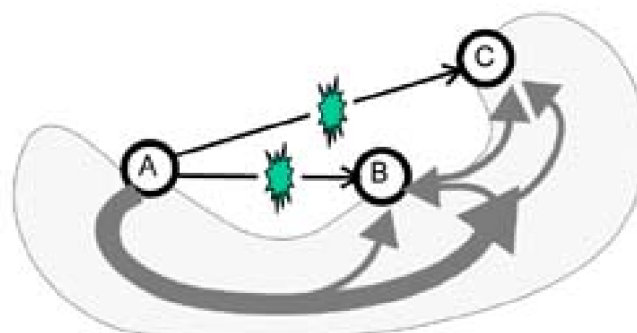
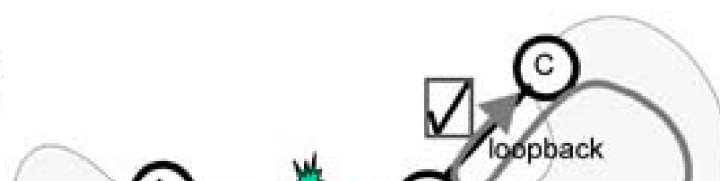
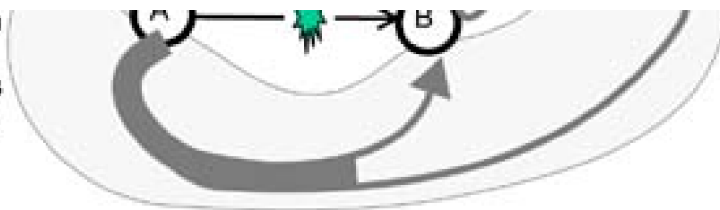


Figure 8-24. Why bypasses take the same or less spare capacity than the fully terminated case.

(a) Fully terminated (basic SCA) case: The logical situation after failure of physical span (A-B) is shown with full termination at node B (no bypass):

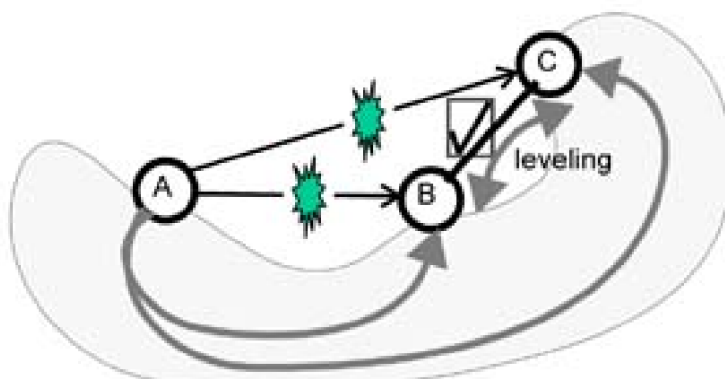


A-B and A-C demand flows are treated as a single total restoration flow requirement between A-B. But the A-C component of this flow may thereby engender loopback spare capacity on span B-C.



(b) **Bypass case:** The *logical* situation after failure of physical span (A-B) is shown with a bypass at node B:

Separate restoration flows for A-B and A-C demands superimpose but each is directly restored to its end-node (without loopback implications).



Surviving span (B-C) enhances efficiency by giving two points of access to surrounding network for either A-B or A-C flows as needed: a flow-leveling effect.

[Figure 8-23](#) shows that the baseline SCA design for the SRLG case will already have assumed the existence of span (A-B) when (A-C) fails, and vice-versa, when considering the capacity design for all single failures. In effect, this is not a luxury that the baseline SCA gets to assume in the bypass case because span (A-B) never gets to be considered as a single failure with a surviving (A-C) span to help out. In effect the baseline SCA design for the SRLG case is more optimistic than it is in the context of the bypass question. And the comparisons are not being made against an identical baseline SCA situation because physical span A-C exists in the SRLG case, but not in the bypass case. Another aspect is that the SRLG case is also slightly more general in that span B-C may or may not exist. If it does not exist, all requirements for the simultaneous restoration flows for physical failures A-B and A-C must be supported through the surrounding network as a whole. The total spare capacity requirement can therefore never be less than when span (B-C) is physically present.

In contrast [Figure 8-24](#) portrays the baseline case of a standard SCA solution for the bypass question. All working flows transit each cross-connect en route, bypassing none, and restoration for all flows occurs between the single pair of end nodes of any span failure. This means that demand that from a working standpoint is part of an A-C flow, is bundled with the A-B working flow and all restored together between the end nodes of the span failure A-B. Any or all of this combined restoration flow may have passed through node C, especially if C is low degree (and with certainty if node B is degree 2). Thus, in general, some of the A-C working flow that previously transited node B is involved in capacitating the spare capacity of span B-C to support loopback.

All other factors being identical, the provision of a bypass through node B in [Figure 8-24\(b\)](#) can, thus, only require the same or less spare capacity in total compared to [Figure 8-24\(a\)](#). With the bypass in place, the one physical cut on A-B manifests itself as two logical span restoration problems, A-B and A-C, each with only part of the total restoration flow. The important advantage is that now the A-C restoration subproblem refers A-C working flow directly to its end-node, avoiding the loopback implications for A-C flow when grouping all restoration on the A-B node pair as in (a). This is the primary advantage of (b) over (a). A secondary advantage of (b), however, is that the surviving physical span B-C directly links nodes B and C. The total restoration flow to be supported from node A to nodes B, C always benefits from this (relative to the more general existing only of indirect network paths between B and C as in (c)). The B-C span in effect provides two points of entry from the network as a whole for restoration flow back to either node B or C. An optimal solver will place spare capacity on span B-C that allows for more efficient leveling of restoration flows to/from nodes B, C back to node A than by using longer routes through the network as a whole.

8.10 Capacity Design for a Known Set of SRLGs

Let us now consider the design of a span-restorable mesh network where a specific subset of known two-span SRLGs are to be accommodated. We go on to use this model as a tool to study how spare capacity is affected by SRLGs in various locations and numbers in a network. We first consider the non-joint design context with working paths taking the shortest routes. Later a jointly optimized model is considered. Joint optimization is of interest where SRLGs are involved because changes in the working routes may be able to mitigate the undesirable SRLG effects on capacity requirements.

The design model for SCA with a specific set of SRLGs can be thought of as an adaptation of the DFMC model ([Section 8.6.1](#)) for all possible dual failures. We therefore rely on the parameters and variables defined for DFMC to which we add and/or restate the following:

- $\Gamma(i) = j$ if span i has an SRLG relationship to span j , $\Gamma(i) = 0$ if span i is not a member of any SRLG. $\Gamma(i) = j$ implies $\Gamma(j) = i$. (Note: $\Gamma()$ should be thought of as a mapping function used for the presentation only (not a set). For AMPL implementation, the same information would be coded as sets of (i,j) pairs common to each SRLG object that exists.)
- f_i^p = the restoration flow assigned to the p^{th} eligible route for span failure i .
- $\delta_{i,k}^p = 1$ if the p^{th} eligible route for span failure i crosses span k , zero otherwise.

For clarity in the present model where spans may be involved in three different contexts, we use: i for a failure span, j for a span that is a co-failure span under an SRLG, and k refers to a span in any other general context, including any surviving span which may be supporting a restoration flow or its related spare capacity variable.

SCA-SRLG

Minimize

Equation 8.50

$$\sum_{k \in S} c_k \cdot s_k$$

subject to:

- Restorability of each span:

Equation 8.51

$$\sum_{p \in P_i} f_i^p = w_i \quad \forall i \in S$$

- Spare capacity (for single failures):

Equation 8.52

$$\sum_{p \in P_i} \delta_{i,k}^p \cdot f_i^p \leq s_k \quad \forall (i, k) \in S^2 | i \neq k, \Gamma(i) = \emptyset$$

- Spare capacity (for SRLG failures):

Equation 8.53

$$\sum_{p \in P_i} \delta_{i,k}^p \cdot f_i^p + \sum_{p \in P_j} \delta_{j,k}^p \cdot f_j^p \leq s_k \quad \forall (i, k) \in S^2 | i \neq k, \Gamma(i) = j, i < j$$

- No restoration flow on SRLG co-spans:

Equation 8.54

$$\sum_{p \in P_i} f_i^p \cdot \delta_{i,j}^p = 0 \quad \forall i, j \in S^2 | i \neq j \cap \Gamma(i) = j$$

The restorability of each span individually (Equation), and the spare capacity requirements to support the set of single-span failures that are uncomplicated by a corresponding SRLG failure (Equation 8.52) are enumerated in the usual SCA-like style. The remaining span failures are SRLG dual failure scenarios, where the spare capacity must support simultaneous restoration flows for span i and its

SRLG-partner $\Gamma(i) = j$. These are enumerated in Equation 8.53. To avoid generating redundant constraints, $(i,j) \leftrightarrow (j,i)$ duplication is avoided with the $i < j$ requirement in the set former of Equation 8.53. i is in effect the primary span of the (i,j) pair to represent that SRLG.

To further reflect the SRLG effects, an instance of Equation 8.54 exists for each span failure i that has an SRLG co-failure span $\Gamma(i) = j$ to assert that no restoration flow can cross span j , for restoration of span i . Note here that (i,j) and (j,i) duplication is explicitly required so that both failed spans are off limits to each other's restoration flow. To assure this we let each span be enumerated as the primary failure span under "i" with corresponding prohibition of flow on $j = \Gamma(i)$.

An important practical difference from DFMC is that this model keeps the same dimensionality of flow variables as in SCA. There is a flow variable for each eligible route under each "primary" span failure, of which there remain only $|S|$. In contrast, for generalized dual failure designs there is a flow variable for each eligible route for each (i,j) dual/failure combination. The total number of spare capacity constraints is correspondingly held at only $|S|$ as well.

A few operational observations are worth making about the capacity design this produces. This model plans for some extreme worst case possibilities (the SRLG failures) that may be far less likely to arise in practice than single failures on spans which are part of those SRLGs. If spans (i,j) are linked under an SRLG their individual failures as single spans may still be far more likely in the real world, because the spans may be a thousand kilometers long and the SRLG only 100 meters. Logically, however, if the SRLG (i,j) failure is to be planned for, it requires enough spare capacity to restore i and j simultaneously in the absence of each other as a surviving span. The corresponding single failure scenarios would therefore be wholly redundant in the model. There is, however, a useful point to note about such designs in practice: normally for most failures, including SRLG spans when they fail outside the SRLG, there will be a *surplus* of

spare capacity for restoration routing. The spare capacity would only be fully needed in the event of the SRLG failure itself (indeed, strictly a forcing SRLG). Thus, SRLG-protected capacity design tends to cost more, but in return for the insurance against SRLG events, one obtains side benefits such as a bigger pool of protected working capacity for dynamic provisioning on other spans, shortened protection paths for single failure events, and undoubtedly yet further enhanced platinum service dual failure restorability against most failures, and so on.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

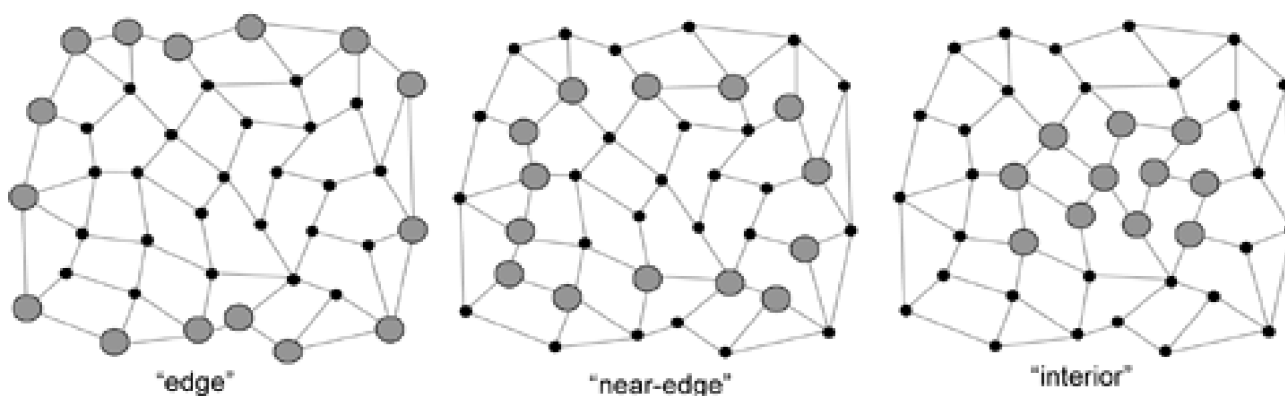
NEXT ▶

8.11 Effects of SRLGs on Spare Capacity

The SCA-SRLG design model was used to conduct a number of experiments in the 40 node, 70 span, test network shown in [Figure 8-25](#) in which each node is classified as shown as an edge, near edge or interior node for later discussion of coincident SRLG effects.^[9] In assessing capacity cost in this network, the length of each span is proportionate to the Euclidean distance on the plane as drawn in [Figure 8-25](#). To avoid infeasibilities when conducting trials of randomly placed coincident SRLGs all nodes of the test case are necessarily of degree 3 or higher. The demand exchanged by each node pair is randomly generated from a uniform random distribution on {1,2,...,10}. There is a total of 780 O-D pairs exchanging 4478 demands for an average of 5.74 per demand pair.

^[9] Work in this section is a collaboration with J. Doucette. During its development a paper based on the work

Figure 8-25. Test network for study of SRLG effects, 40 nodes, 70 spans, $\bar{d} = 3.5$ with identification of edge, near edge, and interior zones for tests of SRLG location.



8.11.1 Randomly Occurring SRLGs

To characterize how the capacity costs rise depending on the number of coincident SRLGs in a network each pair-wise combination of spans incident at each node in the test network ([Figure 8-25](#)) was considered at random to be included as an SRLG. The percentage of such randomly instantiated coincident SRLGs present relative to all possible coincident SRLGs in the network was increased from zero to 100% in increments of 10%. Ten individual random test cases were used at each percentage. Obviously 100% of all possible SRLGs would not actually be present in a network but the interest in conducting the experiments all the way up to 100% is to permit comparison against designs which explicitly support all possible dual failures. In addition, a strategy to cope with concerns that "hidden" SRLGs may be present almost anywhere is conceivably to design for all possible coincident SRLGs.

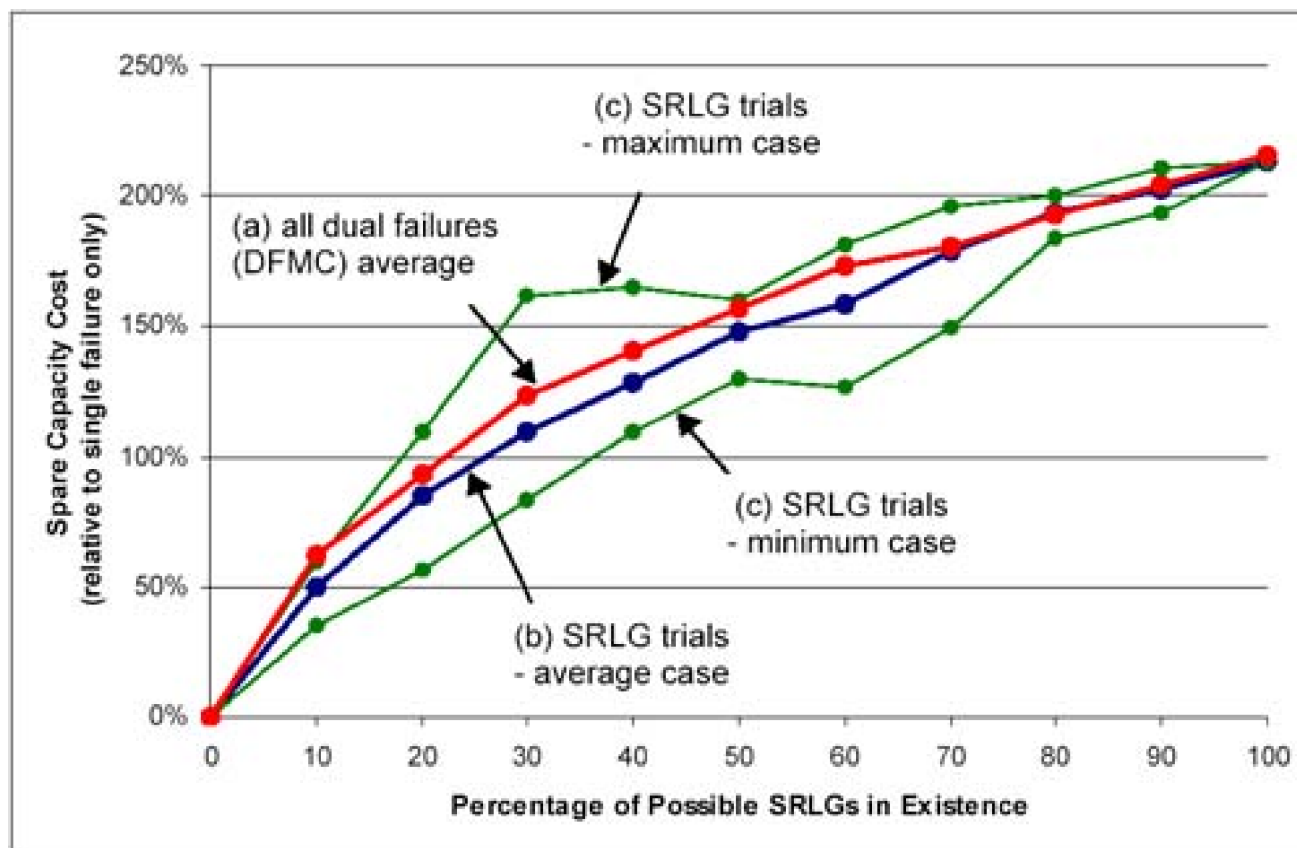
[Figure 8-26](#) summarizes the results in terms of the excess spare capacity cost as the number of coincident SRLGs increase. The curves labeled maximum, minimum, and average coincident SRLG show the average and range of the spare capacity cost over the 10 random trials at each percentage of possible SRLGs present. The test network has 21 nodes of degree-3, 18 nodes of degree 4 and one node of degree 5. There are, therefore,^[10]

$$^{[10]} \text{By } \binom{n}{x} \text{ we mean "n choose x" i.e. } \frac{n!}{x!(n-x)!}.$$

Equation 8.55

$$21 \cdot \binom{3}{2} + 18 \cdot \binom{4}{2} + 1 \cdot \binom{5}{2} = 181$$

Figure 8-26. Excess spare capacity costs arising from span SRLGs in the network



possible coincident SRLGs. Overlaid for comparison in [Figure 8-26](#) is a separate set of results for the corresponding fraction of all general dual failure test cases obtained with DFMC.

If we believe that there are actually relatively few SRLG hazards in real networks, the part of the curve under 10% saturation is of main interest. Here we see that if even 10% of possible SRLGs are present it may cost from ~35% to 60% extra in spare capacity to assure 100% restorability. The average is 50% excess spare capacity if 10% of all possible coincident SRLGs are present. This is, however, ~18 SRLGs on the 40 nodes of the test case, which still may be relatively many compared to real networks. A more meaningful way to consider the impact at the low end is to study single SRLG test cases. This follows shortly.

In comparing the curves for coincident SRLGs to those for dual failures, an interesting finding is that if, in the design, we allow for the specified percentage of all possible dual failure scenarios, the cost of spare capacity is only slightly greater than if we only have *coincident* SRLGs at the same relative frequency. This is somewhat unexpected based on consideration of the number of coincident SRLG combinations compared to the total number of all dual failure combinations. Given the total of 181 possible coincident SRLGs, 50% of all possible coincident SRLGs means approximately 90 SRLGs to consider in the design (in addition to single-span failures). But the number

of *all possible* dual failures is $\binom{70}{2}$ or 2415. Yet covering 50% of the coincident SRLGs costs only about 4% less than covering 50% of all dual failures.

The explanation is that coincident SRLGs are much more important and dominant in the capacity design than are the vast majority of other dual failure scenarios. This is consistent with the notion of span restorability being limited by the incident cutset around the end nodes of each span failure (i.e., the "end-node bottleneck" effect discussed in [Section 5.4.2](#)). It also suggests that in almost any circumstances where X% of the possible coincident SRLGs actually exist, if we provide the extra spare capacity to cover those failure scenarios, we get coverage against nearly X% of all other dual failure combinations as a side effect, despite the greater number of the latter. Indeed for the particular test network studied here, we find that by designing for all 181 coincident SRLGs, 100% of all the 2415 dual failures in general are also restorable under the same spare capacity. Thus, coincident SRLGs are much more important in the capacity design than are the

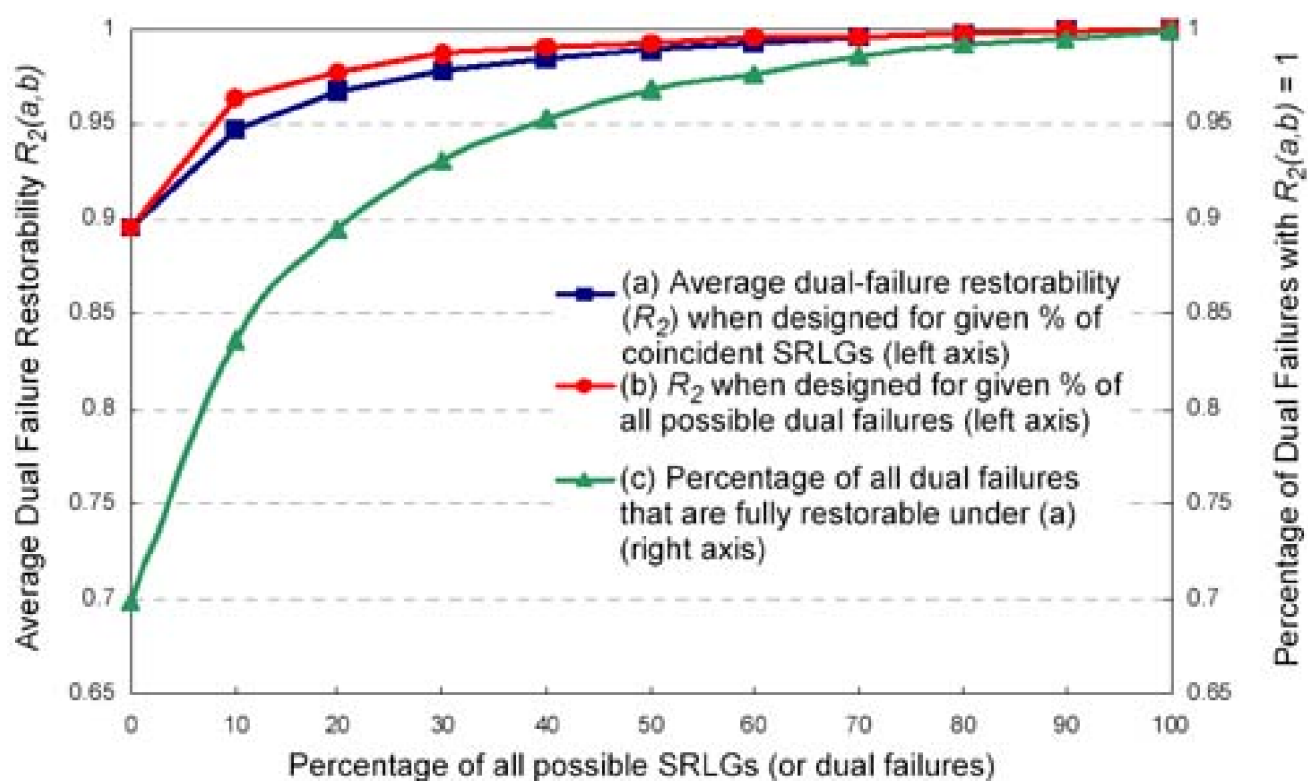
vast majority of other ordinary dual failure scenarios.

8.11.2 Availability Benefit of Coincident SRLG Design Coverage

This section further investigates the effect of dual failure coverage arising from coincident SRLG coverage. Obviously, covering coincident SRLGs increases average path availability compared to not doing so. But because the process adds spare capacity it probably also has a more general availability enhancement effect as well. In [Section 8.4](#) we saw how path availability of a span-restorable mesh network relates primarily to the dual failure restorability, R_2 . We can therefore link the effect of coincident SRLG coverage to availability by determining how R_2 benefits from the particular form of spare capacity addition for coincident SRLGs.

To address this question we used the methods of [Section 8.4](#) to calculate R_2 values for each of the SRLG test case designs in [Section 8.11.1](#). The results appear in [Figure 8-27](#). The abscissa is the percentage of all possible coincident SRLGs or general dual failure scenarios included in the design. On the left we have R_2 of the resulting design. The right ordinate is numerically the same but is a separate assessment of the fraction of all general dual failures that are 100% restorable. The top curve shows how R_2 of a DFMC capacity design for general dual failures rises as a greater proportion of all dual failure scenarios is included. The next lower curve shows how R_2 rises against the corresponding percentage inclusion of *coincident* SRLGs. Keep in mind that there are 2415 dual failure scenarios but only 181 coincident SRLGs. The third curve shows the fraction of individual dual failure scenarios that are fully restorable (i.e., those for which $R_2(i,j) = 1$). We see that, at least in this test network, addressing only the coincident SRLGs is virtually as good as designing for all dual failures. In fact R_2 reaches 100% if every coincident SRLG is covered. This suggests that in practise coincident SRLGs are the most strategic class of dual failures to consider guarding against. It also suggests that an SCA-SRLG solution with all coincident SRLGs may serve as a good starting point or surrogate solution for a more difficult DFMC problem; this is to say in other words, that SCA-SRLG is of interest as a heuristic to approximate a full-blown DFMC design or obtain a starting solution for the latter.

Figure 8-27. Covering coincident SRLGs is effective protection against dual failures in general.



8.11.3 Predicting the Impact of a Specific SRLG

Let us now consider what determines the severity of the impact that a given SRLG has on the spare capacity required cost to retain 100% restorability against single-span failures. If we can diagnose or predict what makes for the "worst" SRLGs, guidelines may be possible to prioritize efforts at SRLG elimination. In this regard, it seems reasonable to test for dependence of the SRLG impact on:

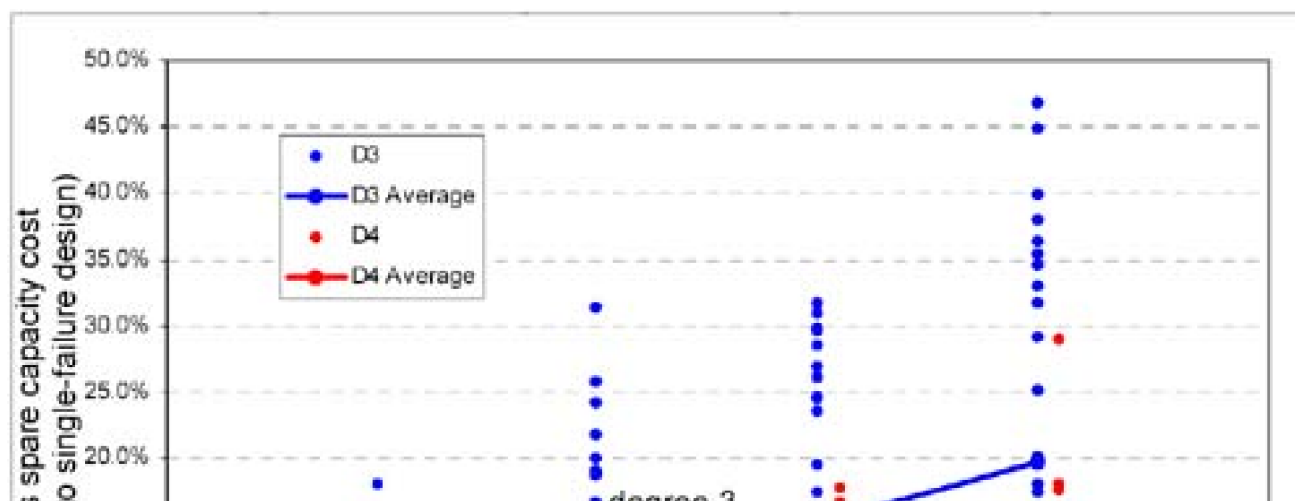
- Topological location in the network.
- Degree of the node where the coincident SRLG exists.
- Working capacity disparity of the node where the coincident SRLG exists.

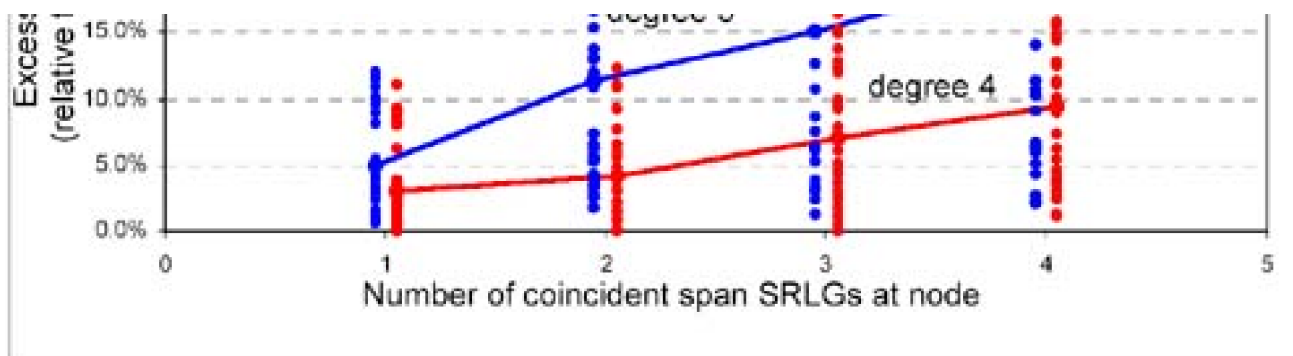
By topological location we mean whether the node is in a core or edge type of region. Nodal degree seems relevant because the simultaneous failure of two spans incident to a degree 4 or higher node should generally be easier to handle than the simultaneous failure of two out of the three spans at a degree 3 node. Further, we know from [Section 5.4.3](#) that the working capacity disparity at a node sets lower limits on the achievable efficiency for single-span failures, so the working capacity of the spans in the SRLG relative to the non-SRLG spans at the same node may also be predictive of SRLG impact. To investigate these effects we generated 240 random test cases. The first batch of 120 involve coincident SRLG placement only at degree 3 nodes. Thirty of the test cases contained only one coincident SRLG at a degree 3 node. Another 30 each contained coincident SRLGs at two different degree 3 nodes. Thirty more each contained coincident SRLGs at three degree 3 nodes, and a final 30 each contained coincident SRLGs at four different degree 3 nodes. A matching set of 120 test cases was similarly divided among coincident SRLGs incident at degree 4 nodes.

8.11.4 Effects of Nodal Degree

[Figure 8-28](#) shows the scatterplot results of each individual test case in terms of the increase in spare capacity cost, relative to SCA with no SRLGs, and the number of SRLGs involved. Test cases involving SRLGs at degree 3 nodes are shown separately from those at degree 4 nodes. On average, SRLGs incident on degree 3 nodes are, as expected, more costly to handle than at degree 4 nodes. This is borne out on the average, but there are many individual cases where the degree 4 impact is greater than the corresponding degree 3 results. Aside from the degree 3 versus degree 4 aspects of [Figure 8-28](#) this data also gives us a look at the range of impact that a single, or just a few SRLGs, can have in a network. The data suggest that on average one SRLG would require a 4 to 5% increase in spare capacity, but depending on the specifics, this may range from essentially no impact to nearly 20% spare capacity increase. Similarly the impact of 3 or 4 randomly occurring SRLGs ranges much more than its own mean. While the average trend continues to increase more or less linearly with the number of SRLGs, even with four SRLGs the overall impact may range from under 2% to nearly 50% increase in spare capacity. Other details of the situation are clearly important, therefore, to predicting the exact impact, but the trend for worse impact at degree 3 nodes than degree 4 is strongly evident. Richly connected nodes are able to distribute their restoration routes (and hence spare capacity) to a greater degree to mitigate the coincident SRLG effects. In and of itself, this suggests that adding a new span could be an option to alleviate an SRLG effect on a degree 3 node (if the SRLG cannot be eliminated at lower cost). The exact impact of a span addition that linked two degree 3 nodes, converting them to degree 4 nodes can be checked with SCA-SRLG.

Figure 8-28. Excess spare capacity incurred by numbers of span SRLGs at single nodes.

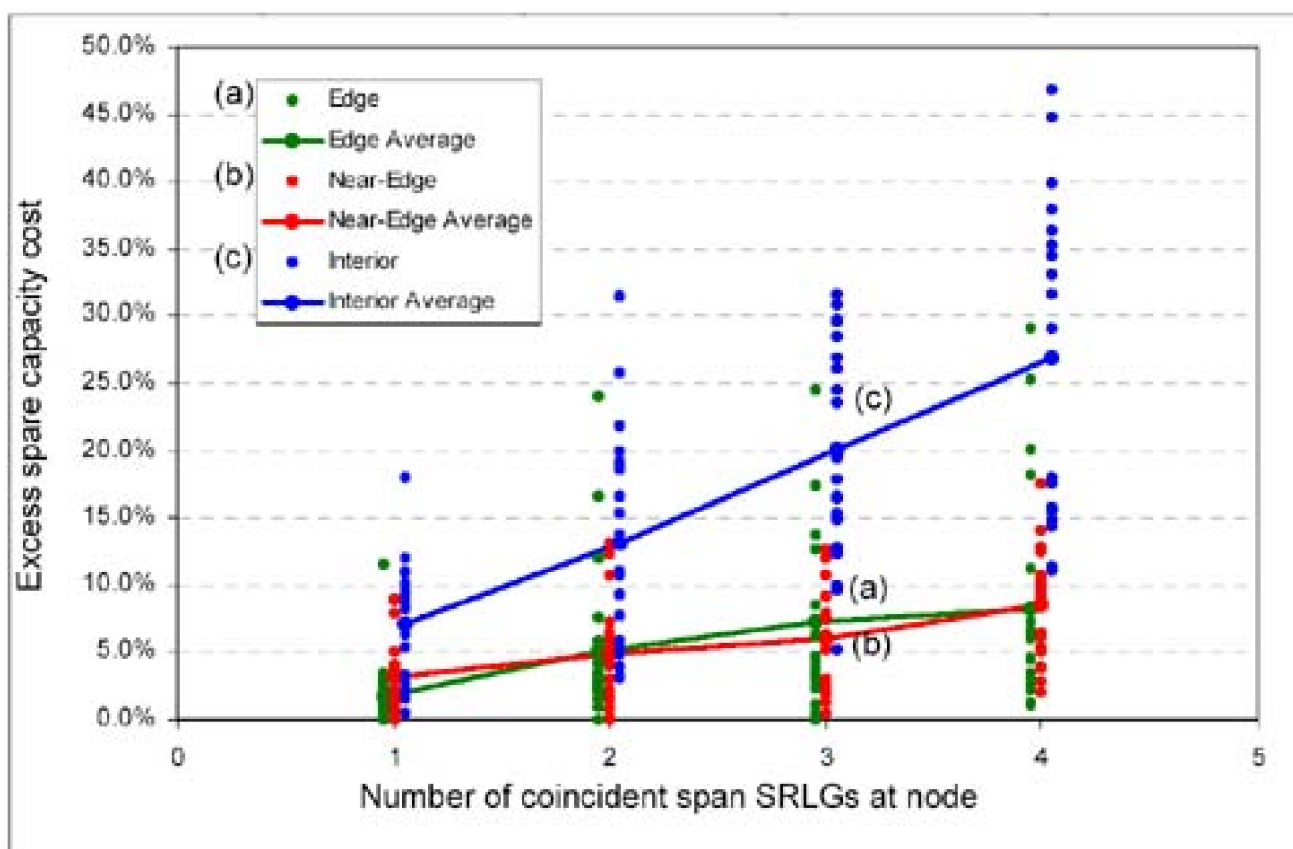




8.11.5 Effects of Topological Location

To test for dependence on location within the network each node was given a classification as either edge, near edge or interior as shown in [Figure 8-25](#). Another round of 240 test cases of SCA-SRLG were then solved, involving one to four SRLG locations placed randomly within the subset of nodes of the same classification. For instance, 20 test cases contain a single SRLG in the set of edge nodes only, 20 have one SRLG at a node "near" the edge, and so on. There are then 20 cases of two SRLGs both at randomly chosen nodes from the edge set, near edge and interior node sets, and so on. The results are in [Figure 8-29](#).

Figure 8-29. Spare capacity costs of SRLGs depending on network location.



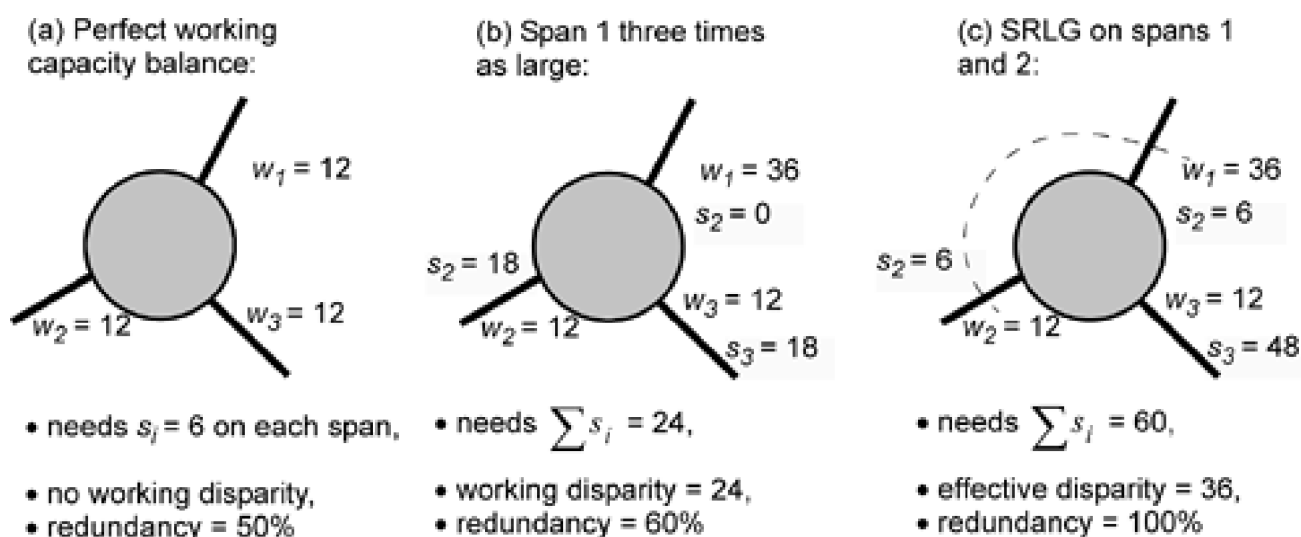
Each data point in [Figure 8-29](#) corresponds to the spare capacity cost increase with the averages of each type of node classification connected by lines. The indication is that SRLGs in the *interior* of the network graph have the most costly impact. The subclassification of edge and near edge is apparently not significant as the edge versus interior core distinction. This seems somewhat unintuitive at first if one considers that routing diversity for restoration is far more limited near the edge than in the interior. Upon investigation, the effect seems, however, to relate more to the greater amount of working flow and capacity on spans at the core. Since demands are shortest path routed, there are many more O-D pair combinations having shortest paths that tend to travel through the interior of the network, than near the

perimeter. Consequently, coincident SRLGs in the interior generally tend to require a higher amount of restoration flow and spare capacity than ones near the edge of the network graph.

8.11.6 Effects of Working Capacity Disparity

Another hypothesis is that when an SRLG maps onto high working capacity disparity at a node, it tends to have a more severe impact on spare capacity requirements. This is a simple extension of the reasoning about end-node limitations to restoration capacity in single failure design of a span-restorable network. The basic observation is that a lower bound on spare capacity is set by the need for egress from the end-nodes of a failure, so that when any span i incident on a node v is cut, the sum of surviving spare capacity at v must meet or exceed the failed working capacity. A consequence of this is that spare capacity efficiency necessarily drops in proportion to the disparity or imbalance between the working capacities at the node. As a measure of the working capacity imbalance we can define the disparity as the difference between the largest and second largest w_i quantities at the node. [Figure 8-30](#) illustrates with a small example of the principle in (a) and (b). [Figure 8-30](#) (c) extends the concept of disparity to show how the minimum spare capacity requirements are further aggravated if the SRLG unifies the larger w_i capacities at the node.

Figure 8-30. Example of how an SRLG aggravates the effective working disparity.



In [Figure 8-30\(a\)](#) the node is shown with perfect working capacity balance, in which case the $d/(d-1)$ topological lower bound on redundancy (for single failures) is realized. For $d = 3$ this is 50% as shown. In (b) we still consider single failures but create working capacity disparity in the example by tripling w_1 . The disparity becomes $36 - 12 = 24$ and the spare capacity minimum rises to 36, and the efficiency falls to 60% redundancy. Now if, with the same nodal conditions as in (b), we add an assumed SRLG on spans 1 and 2, span 3 must bear 48 units of spare capacity. Effectively the disparity rises to $(w_1 + w_2) - w_3 = (36 + 12) - 12 = 36$. Thus for coincident SRLGs we can define the *effective disparity*, which is the difference of the largest amount of working capacity that has to be restored at any one time (i.e., on two spans), relative to the next largest amount of single failure working span capacity.

These observations lead to an expectation that wherever the coincident SRLG at a node tends to aggravate the effective disparity, the spare capacity impact will be higher. The effective disparity is relevant to the spare capacity impact because a large w_i total to be restored in one scenario forces a large amount of spare capacity at the node. If, however, the next largest failure scenario in terms of w_i is much smaller, then the spare capacity from the first scenario is not efficiently sharable over an almost equal amount of different working capacity, and so the achievable efficiency is limited. To test for evidence of this effect we analyzed the individual node working capacity circumstances of the 18 single SRLG test cases in [Table 8-7](#) to see how well SRLG disparity serves as a predictor of the spare capacity impact. The actual measure of disparity implemented for these tests was:

Equation 8.56

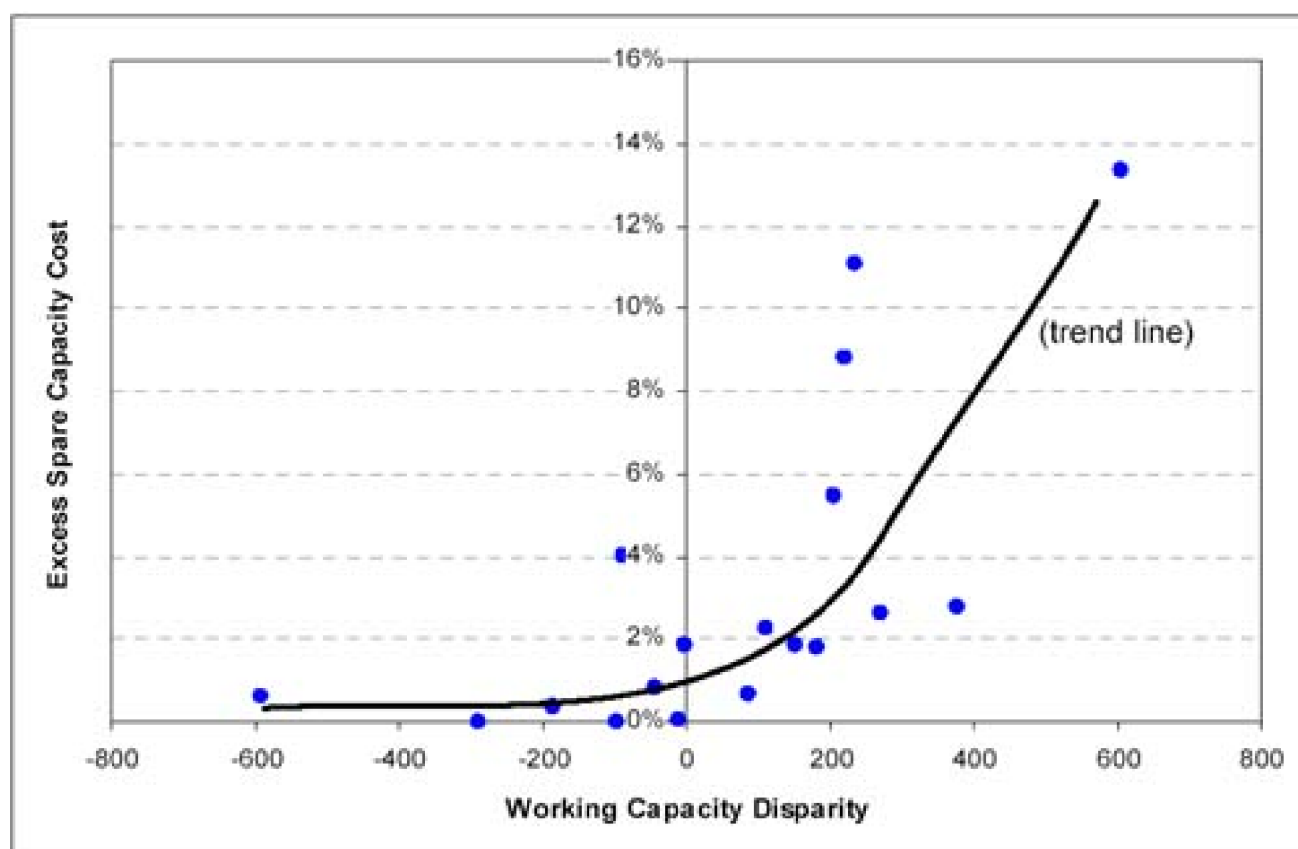
$$\text{SRLG_disparity} = \sum w_i - \sum w_j$$

$$i \in \overline{\{SRLG\}} \quad i \notin \overline{\{SRLG\}}$$

where i indexes the spans incident to the common node of the coincident SRLG and $\overline{\{SRLG\}}$ denotes the two spans linked under the SRLG. Note that as defined the SRLG disparity can be negative or positive. Negative disparity indicates that the SRLG is not impacting the majority of the working capacity at the node. We would expect little spare capacity impact in such cases. Positive disparity is, however, indicative of the deleterious effects illustrated in [Figure 8-30](#) and is expected to have a significant impact on spare capacity.

[Figure 8-31](#) shows the results for the random set of 18 SRLGs. It suggests there is a fairly strong correlation between the spare capacity and positive SRLG disparity, while zero or negative disparity SRLGs are uniformly and markedly less troublesome.

Figure 8-31. Spare capacity cost increase versus SRLG working capacity disparity.



Notably the four highest data points in [Figure 8-31](#) correspond identically to the four most troublesome SRLGs in [Figure 8-31](#) and the discussion of [Section 8.11.7](#), so the effective disparity created by an SRLG at a node seems to be one of the best predictors of which SRLGs would be the most important to mitigate by removal or by other measures to reduce the impact, such as routing changes to reduce the disparity. In this regard a span addition could again be considered as it can simultaneously reduce the disparity effect and increase the nodal degree, both of which counteract the SRLG impact.

8.11.7 Benefit of Removing the Most Troublesome SRLGs

One of our initial motivations in studying the impact of coincident SRLGs was to understand how a network planner might assess the cost-benefit of an actual SRLG removal project. In this section we therefore test whether the *isolated* impact of an SRLG on the SCA-SRLG capacity design is a good predictor for ranking SRLGs as candidates for physical rearrangements that would eliminate the SRLG from the network.

Using the SCA-SRLG design model, we designed a test case network for full restorability with 18 randomly placed coincident SRLGs

(Case 1). (In this network 18 SRLGs is 10% of all possible coincident SRLGs). The benchmark design with no SRLGs was solved and referred to as Case 0. We then solved test cases where each of the 18 SRLGs is present *in isolation* (Cases A through R). The excess spare capacity cost of each SRLG individually was then used to rank them from highest to least impact. Based on this ranking the four worst SRLGs were removed one at a time from Case 1. The results—in [Table 8-7](#)—demonstrate how efforts at SRLG removal can benefit from this kind of analysis. Removal of just the four most individually troublesome SRLGs produces almost all the benefit of a more costly effort to remove all 18 of them. Even if the network operator could only afford to eliminate a single SRLG, choosing by this kind of impact ranking would in this case improve the cost of spare capacity by nearly 22%. Consistent with other findings above, it was also found in this exercise that the five most costly of the 18 randomly placed SRLGs are all coincident to nodes in the core of the network.

Table 8-7. Effects of Identifying and Removing the Most Troublesome SRLGs

Test Case	# SRLGs	Normalized Cost	Rank	SRLGs Removed	Cost Improvement
0	0	0.00%	/	/	/
1	18	45.93%	/	/	/
A	1	0.00%	17	/	/
B	1	0.64%	14	/	/
C	1	0.82%	12	/	/
D	1	8.82%	3	/	/
E	1	0.04%	16	/	/
F	1	2.26%	8	/	/
G	1	5.46%	4	/	/
H	1	1.88%	10	/	/
I	1	1.88%	9	/	/
J	1	1.81%	11	/	/
K	1	2.65%	7	/	/
L	1	13.36%	1	/	/
M	1	0.68%	13	/	/
N	1	0.35%	15	/	/
O	1	2.77%	6	/	/
P	1	4.05%	5	/	/
Q	1	11.09%	2	/	/
R	1	0.00%	18	/	/
2	17	35.87%	/	L	21.9%
3	16	32.45%	/	L, Q	29.3%
5	15	25.59%	/	L, Q, D	44.3%
5	14	20.45%	/	L, Q, D, G	55.5%

8.11.8 SRLG Effects on Other Protection Schemes

A final point to emphasize is that while we have considered span-restorable networks, SRLGs have similar general effects on other schemes as well. Span-restorable networks build SRLG considerations into the restoration routes, whereas SBPP consults an SRLG database and a database of current primary-backup shareability relationships in determining the eligible backup routes for each new primary path. A span SRLG thus forces the adoption of somewhat less efficient backup routes in SBPP, limiting otherwise higher sharing opportunities. There is no survivable networking scheme that will not in general suffer some capacity penalty in the presence of SRLGs. Conversely, there is no scheme for which a hidden SRLG may not also result in some loss of protection or restoration if the spare capacity was planned for a set of failure scenarios that excluded the given SRLG. Therefore, once again, we see the advantage of backstopping a preplanned protection scheme for single failures with an adaptive restoration response in the event of a dual failure or hidden SRLG.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

Chapter 9. Mesh Network Topology Design and Evolution

Almost all studies of mesh-restorable networks have so far been posed in a context where the physical layer topology—the set of actual rights-of-way on which fiber can be placed—is given as an input. Many planning problems fit this context in that the physical topology is essentially "a given" and does not change on the same time scale as logical topology and current capacity planning. On the other hand, longer term planning groups in network operating companies may be looking at acquiring new rights-of-way that would evolve the basic topology. And business planners may be assessing which cities to add as new nodes on the network. One of the reasons physical topology is almost always considered fixed is that topology extension (adding new rights-of-way) is extremely expensive in general and topology change (deletion or swaps of routes) is quite disruptive to existing services and network operations. Many network operators employ a legacy topology that has evolved slowly, in some cases over centuries, so that it becomes natural for planners to view the topology as an inherited constant, for most purposes. In the oldest networks—European networks and those of the original Bell System companies in North America—edges in the physical graph would have been slowly accumulated as communities-of-interest and commercial ties grew between cities, often paralleling new highway or railroad routes. Additional spans would be added on a case by case basis, driven by the economics of working demand conveyance, rather than as part of an overall design which considers the trade-off between demand routing, the requirement of biconnectivity, the sharing of restoration capacity, and the specific restoration or protection processes of a mesh-based survivable network.

In North America, some newer networks have been assembled almost entirely from rights-of-way acquired from utility infrastructures, such as pipelines or railways. Other utility operators, such as electricity and gas providers, want to offer transport services based on their power-line or pipe-line topologies. They have realized that aside from their main business, the rights-of-way they own are a tremendous asset from which added value can be obtained by offering transport networking service.

Before the widespread deployment of fiber optics, long haul backbone networks tended to be tree-like. Trees are the natural topologies that arise from minimum cost design to serve a set of working demands without need or consideration of network level rerouting for survivability. Considerable theory and methods for design of tree-like backbones and private (leased line) network design with limited backbone alternate routing capability have been developed. In private line network design any desired logical edge *can* be provided, it is only a question of the tariff. The actual routing to realize the logical edge is of almost no concern to the private network designers. In contrast, the feasible options to augment or change the underlying facilities graph of a transport network are typically much more limited. And a new edge of the physical graph must truly be what it seems—a new, essentially direct, physical route between node locations. It cannot be simply a logical abstraction created by routing over other existing edges. The mesh-survivable topology planning problem itself is also quite different from the context of virtually all prior theory for topology optimization: never before has the topology been fundamentally restricted to biconnectivity and had to encompass the consideration of restoration routes and shared spare capacity in addition to the traditional requirements of routing and capacity placement to serve working demand.

9.1 Different Contexts and Benefits of Topology Planning

This section identifies practical contexts in which network planners may want to use and/or extend the methods of this chapter to address issues of physical layer topology planning.

Incremental Evolution

Often the practical manifestation of a need for a topology design capability will be to address an incremental or evolutionary planning question: "If I have so much capital for next year's construction budget, which new span establishment project(s) would have the best overall impact?" This question recognizes that in a mesh network a new span must be considered not only in terms of the added efficiencies it provides to routing of working demands, but also on how it interacts with the network as a whole to enhance the sharing (and hence reduction) of total spare capacity for restoration.

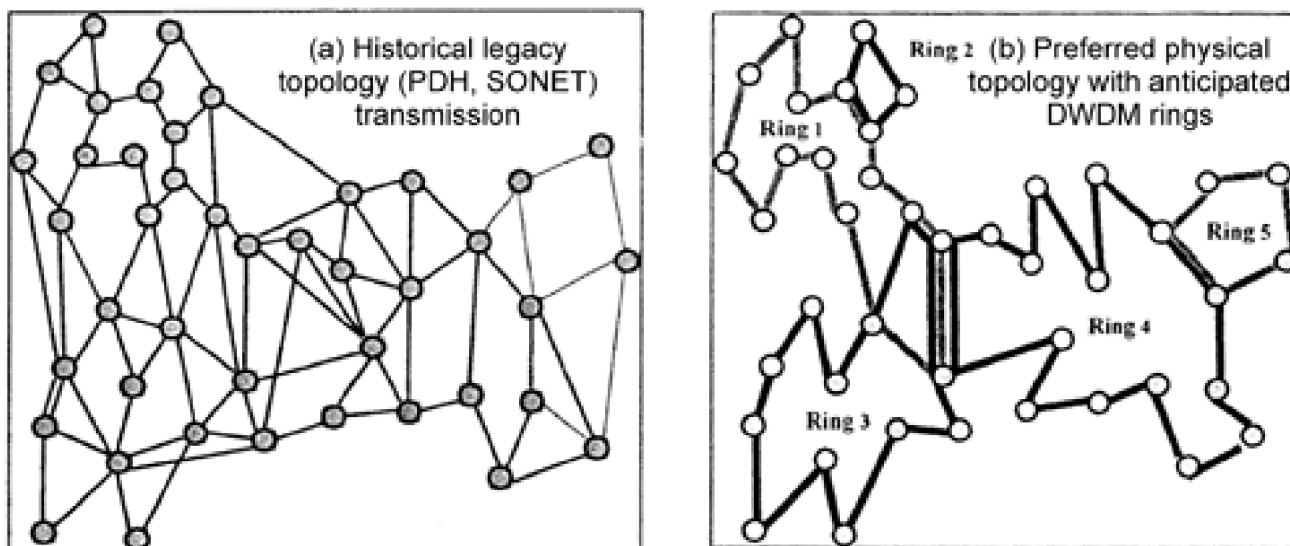
By posing a question of *new span addition*, the network operator is also reflecting that the real, practical, problem is usually one of topology evolution, not a "green fields" or ground-up situation of brand new topology design. Nonetheless insights about the incremental evolution problem arise from considering the green fields problem, as this often indicates target architectures that can suggest the direction of the evolution to follow from the present on the ground situation.

Green Fields—Startups

Actual cases of the "green fields" planning problem do arise, however, with the startup of new facilities-based network operators. An example is Level(3). This company assembled an entire North American topology "from scratch" in the 1990s. Level(3) and others (Flag, 360 Networks, Global Crossing) also built intercontinental networks from a clean slate about the same time. Despite the economic difficulties that later beset these companies, they are cases in point of actual "green fields" network building exercises. The initial set of physical routes in the North American portion of the Level(3) network comprises roughly 16,000 route-miles, requiring billions of dollars of capital for the acquisition and preparation of routes for use. Apparently, about 30,000 separate legal contracts were required to assemble all the required leases, purchases, permits, to create the basic topology. Then every right-of-way had to be prepared with trenching and installation of concrete or PVC-ducting, manholes, equipment housings, signage, power, and so on.^[1] It is a case in point as to just how expensive the acquisition and turn-up of new physical rights-of-way is and, hence, how strategic it can be to carefully optimize the topology. In the Level(3) example, it appears that the rights-of-way were acquired with the layout of eight continental-scale rings in mind, giving presence in about 60 major U.S. cities. Although the topology (see [Figure 1-14\(a\)](#)) is sparse, it was planned with survivability in mind so it seems likely that it may have no span SRLGs whatever. In other words, each logical span between nodes *is* physically disjoint from others, by design. It would be interesting to see how differently this case of a green fields topology design might have been if mesh- rather than ring-based principles had been in mind from the start. But now that this network is built, the methods here can be applied to see what changes or additions of topology might enhance it if it evolves to mesh-based operating and protection principles.

[1] For those of us excited about all aspects of "high tech" telecommunications—EE, CS, mathematics and science disciplines, etc.—it is ironic to note that the dominant cost in establishing a network is actually in civil engineering and law. Some *construction* companies made much more money from telecommunications than did the more obvious high-tech R&D-based companies in the 1990s.

Figure 9-1. Illustration of how technology costs can alter topology requirements [AINa98](#).



Topology Too Sparse?

A notable aspect of some North American carrier networks is a relatively low degree of physical layer connectivity—often $\bar{d} < 2.5$. To an extent this is a reflection of great distances and relatively sparse population in some areas, such as in the U.S. midwest regions and much of Canada. This is combined with the great cost of facility routes and the ever-increasing capacity and economy of scale in transmission systems. Such sparseness tends to be economic if transmission costs are high relative to routing or switching costs, and the available system capacities are large relative to the average working flow. Economy of scale in large capacity modules also encourages such sparseness. But if demand grows to where even a 256 I DWDM fiber system is no longer a "large" amount of capacity, then the balance tips back toward a more richly-connected topology. In either case, the opportunity for network efficiency improvements from the addition of just one or a few new spans is greater when starting with such sparse topologies than in a corresponding European network that may already have $\bar{d} > 35$. Another general concern, aside from strict economic optimization for survivability against any single failure, is pressure from customers for more diversity options within the network for general availability enhancement and robustness to multiple-failure situations.

Topology Too Rich?

Indeed, given the anticipated capacity of DWDM systems, the opportunity for topology planning benefits in European networks may be in the direction of deciding which facility routes to sell off or lease to other operators. Such paring down toward a sparser topology can take advantage of the great changes in cost versus capacity of transmission systems that have occurred since the era in which the current topologies evolved. A dramatic example of how technology —through its effects on capacity versus cost characteristics of transport— can in principle alter the topological requirements is seen in a planning study by Allen et al., [AlNa98]. Under the particular cost assumptions for future transport technology (including both line and node equipment costs), and relatively strong nearest-neighbor demand patterns, the following set of five rings (stacked OC-192 BLSR rings over 16-channel DWDM fiber) were found to be nearly cost-optimal. With the cost data employed in [AlNa98] the mesh alternative was under 5% more expensive, but the topological requirements of the ring solution, as illustrated in [Figure 9-1\(b\)](#), are dramatically different from that of the legacy mesh facilities topology that the planning study assumed to start with in (a).

Although the target architecture in [Figure 9-1\(b\)](#) is rings, which are relatively well suited to a sparse topology, the principle that [Figure 9-1](#) illustrates applies to mesh networking as well. If the cost of capacity (including nodal switching and distance related line costs) drops dramatically through technology evolution, then a sparser mesh topology can become the new optimum. Thus we have a need for planning methods that can also detect when the current topology has possibly become overly rich, creating opportunities to sell off or lease out or discontinue leases on unneeded spans.

Merging Networks

Another context in which the topology-related considerations of this chapter could provide input to corporate decision-making and network planning is in the acquisition of—or merger with—other facilities-based operators. Company A and Company B might each have sparse networks. Their merger could result in either a significant topological enhancement or an equally sparse combined network on more nodes. The network synergies may be greater if Company A acquires Company C instead. Obviously many other business considerations come into such a decision, but the opportunity to run a combined survivable mesh network at, say, 40% redundancy instead of 80% could help influence the overall decision. This may be especially relevant in an era where network operators are struggling economically and may need to consider mergers or may be candidates for purchase at bankruptcy rates. For instance, from a network topology standpoint we could ask if Sprint would be better off buying the right-of-way infrastructure of MCI, 360 Networks, or Global Crossing to complement their existing topology.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.2 Topology Design for Working Flow Only

Although topology optimization for mesh restorable networks is a new problem, topology design to serve a set of working demands without explicit considerations of protection, such as biconnectivity, spare capacity, and restoration/protection mechanisms, is a fairly well-studied area in data communication and leased-line private network planning. In this section we become aware of some of the prior work on topology problems in general, with a particular viewpoint as to how it informs strategies for the mesh-based survivable topology problem.

Most prior work on topology optimization for communication networks has also been undertaken in the context of data communication networks, leased line networks, and circuit-switching trunking networks. With few exceptions, the problems addressed involve one form or another of the fundamental trade-off between demand routing costs and fixed costs for establishment of each edge in the graph with no constraints of biconnectivity and no considerations required for the incorporation of a distributed reserve network of spare capacity for restoration.

Rather coincidentally "mesh networks" are referred to in some of the classic literature—for instance Kershenbaum [Kers93] p.305 and Cahn [Cahn98] p. 205—but in their context the term refers only to the departure from assuming strictly spanning tree-type topologies. Any network that provided more than a single route between nodes was referred to as a mesh backbone, even though biconnectivity was not implied as a requirement and the topologies could still be close to being trees. Classically, "mesh" just refers to any case where there may be more than one possible route between node pairs. This is why we refer to the present problem as the *mesh-restorable* network design problem to keep it distinct from the more general meaning of a mesh network in prior literature.

9.2.1 Branch Exchange

A general class of heuristics for searching on topology is called "branch exchange." These approaches begin with a feasible topology and proceed with local modifications (addition, deletion, or an exchange move) on the graph edges, seeking to maximize some problem-specific figure of merit on each move. For instance in a data communications network one may start with a minimal spanning tree then seek new link additions that maximize the reduction in average delay to the increase in cost for the link. Note that this implies revision of the routings within the network in the presence of the added link to assess the figure of merit. A related possibility is to start with a full graph topology and successively identify links to drop by a figure of merit such as cost per unit flow handled. Rerouting of demands is again implied to evaluate each topology alternation. More generally, as the name suggests, branch exchange algorithms consider simultaneous deletion and addition of edges, equivalent to an exchange. In the data communications context an approach that has worked well is to specify lower and upper bounds on delay and within the allowable ranges, accept any exchange or add/ delete move that reduces cost, even in delay increases, but is within a bound [Robe99]. Kershenbaum points out that while the basic branch exchange approach is quite general, its the main drawback is that the rerouting of demands (to evaluate the benefit at each step) occurs within the inner loop of the process generating the combinations of exchanges to test. "Since routing itself is typically $O(N^3)$ this tends to make even simple branch exchange searches $O(N^5)$, which is prohibitive for moderate to large size networks" [Kers93].

9.2.2 Cut Saturation

One idea for improving the performance of branch exchange algorithms is the cut saturation algorithm [Gavi90] [ChGe74]. The idea is that by detecting flow saturated cuts of the graph, the branch exchange process can be guided to discover effective exchanges in fewer iterations. This is done by generating exchange moves that take a lightly loaded link from one side of the saturation cut and move it to become a link that joins a node on the other side of the cut, thus effectively moving a lightly utilized capacity investment to beef up the cross section of the saturated cut. Furthermore heavily used cuts are efficiently identified with Kuskal's minimum spanning tree algorithm where link utilizations are used as the edge weights.

9.2.3 The MENTOR Algorithm

A widely used algorithm for data network topology design including aspects of concentrator location is the MENTOR algorithm [Cahn98][KKG91]. Although MENTOR is highly specific to the issues of data networking it embodies basic ideas that may be useful in attacking the restorable mesh topology problem. First, as [Kers93] notes, any approach that involves consideration of all $N(N-1)/2$ possible graph edges on N nodes to determine topology and involves a revised solution to the routing problem $O(N^3)$ to evaluate each edge must inherently be $O(N^5)$ or higher. In fact $O(N^5)$ is for greedy one-at-a-time assessment of each possible edge for inclusion in the graph. The globally optimal problem would require consideration of every distinct subset of $(N-1)$ or more edges out of the $N(N-1)/2$ possible edges. In other words, the complexity for strictly optimal solution for the topology involves testing

Equation 9.1

$$\sum_{i=N-1}^{N(N-1)/2} \binom{N(N-1)/2}{i}$$

combinations, which is $O(N^2!)$.

MENTOR, however, is $O(N^2)$ and yet delivers good data network designs. The key is that it replaces the actual rerouting of demands to evaluate link additions or changes with an easily computed *surrogate* criterion for the routing subproblem called "implicit routing." It uses easy to compute metrics that stand as surrogates for the following desirable characteristics:

1. Demands are routed on relatively short paths.
2. Links have moderate utilization levels (not extremely low or high).
3. Demands are relatively well-aggregated so that the design benefits from economy of scale effects in the cost of link capacity.

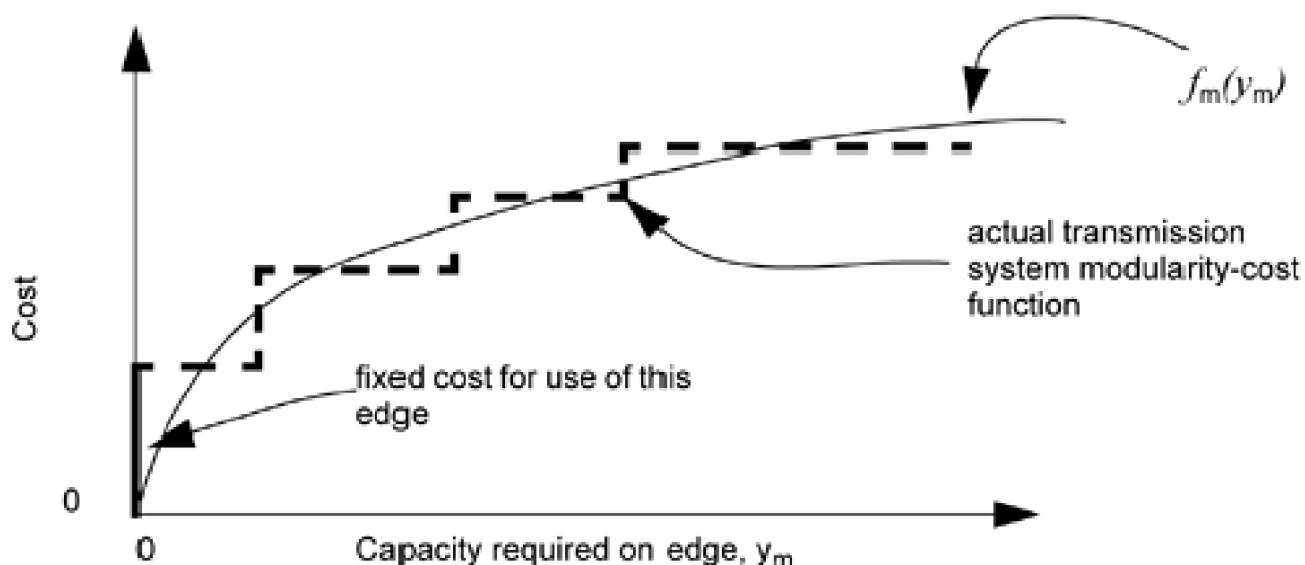
MENTOR starts with an $O(N^2)$ step to find a "center of mass" node that minimizes the total distance-demand product of all other nodes traffic in relation to the centroid node. This defines a main hub or switch location to which all traffic could be brought for redistribution to destinations. Next a process of identifying other backbone nodes is pursued using a simple empirical criterion that blends the relative importance of demand quantity and distance to/from other nodes in controlling the selection of other backbone nodes. The significant point is that detailed routing assessments are avoided, and simple figures of merit based on the designers insights about the problem are used to make these selections. Finally, a tree is formed among the backbone nodes.

9.2.4 Yaged's Fixed Point Convergence Method

A rather elegant approach to determining a minimum cost network topology (and implicitly related routing) in networks where cost depends only on the edges used and the flow on each edge (i.e., no restorability considerations per se) is to let the economy of scale effect attract demand flows to certain routes and edges, to minimize total cost while determining topology at the same time. This is Yaged's fixed point method ([Yage71] and description in [Robe99]). Let there be a cost function $c_m = f_m(y_m)$ that gives the total cost of capacity on edge m if a flow of y_m crosses the edge. The function $f_m(y_m)$ can have any shape as long as it is continuous, positive only in its function values and its first derivative, and has a second derivative that is strictly negative. These conditions just mean any function of the type illustrated in [Figure 9-2](#) where the function bends over or flattens more so as the independent variable rises. Although the function has to be continuous and for convenience describable in an analytic form, functions of this type can approximate the reality of a fixed charge plus incremental routing cost environment. A multi-modular capacity environment with economy of scale can be approximated, as illustrated in [Figure 9-2](#). There can be one such function for each possible edge m , so that both distance and capacity-related cost dependencies for the edge can

be built into the function $f_m()$.

Figure 9-2. Approximating fixed charge plus multi-modular routing costs with a continuous function for application of Yaged's method.



The optimization problem is then:

Minimize

Equation 9.2

$$\sum_{m \in E} f_m(y_m)$$

where E is the set of all possible edges to use in the network graph.

Yaged has shown that under the conditions above for $f_m()$ there is a *fixed-point solution* to the flows and costs on each edge that is also the optimal solution to [Equation 9.2](#). A fixed point solution is a situation where iterative application of a matrix transformation A on some state vector a converges to where repeated application of the transformation no longer changes the state vector. In other words, when a is a fixed-point solution $a = [A] \cdot a$. The significance is that from a wide range of initial conditions, the system converges to the fixed point solution and thereafter is stable in that state. In this case it means that if we start with a set of flows where all demands are shortest-path routed and we begin iteration of the process: routing \rightarrow flows \rightarrow costs \rightarrow routing (updating the costs each time according to the new flow on each edge)—this process will converge to a cost-optimal set of routes, flows, and edge costs.

Mathematically, if we denote our routing process (assume it is least cost routing for each O-D pair, but it can be any routing criterion for the fixed-point result) as A , which operates on a current set of edge costs $\{c_m\}$ and produces updated edge flows $\{y_m\}$, the fixed point theorem says there exists a set of edge costs $\{c_m\}^0$ such that:

Equation 9.3

$$f_m(A\{c_m\}^0) = \{c_m\}^0$$

If one iterates the routing process based on the current costs, updates the flows accordingly, and recomputes the costs, convergence on a fixed point results where all routing decisions (hence edge flows and edge costs) stabilize and represent a minimum cost routing and topology solution. In other words, the edge flows play the role of a and process of updating costs and repeating the least-cost routing is the transformation operator $[P]$. The topology solution is obtained from the result by eliminating those edges with no flow at the converged state. It is because of the "concave"^[2] nature of the cost function that a fixed-point solution exists.

^[2] A function is convex if for any two points p and q that are on the function's surface, all points on the line segment between p and q are contained within its volume. In Robertazzi's comment, the *absolute* cost versus capacity function is actually convex—he is referring to the curve of *normalized* cost per unit demand carried, that results from this characteristic, which is concave.

Robertazzi [Robe99] notes how the driving effects in minimum-cost topology determination under these nonlinear cost versus capacity characteristics, mainly an aggregation of demand flows to exploit economy of scale, is directly opposite to the aim of spreading demand on several routes to minimize queuing delay in a packet network. In that case the "cost" or penalty function is convex; as one aggregates more demand onto on link, the delay rises more than proportionally whereas here, aggregating demand causes cost to rise *less* than proportionally.

9.2.5 The Fixed Charge and Routing (FCR) Problem

FCR is a classic topology problem studied in the O.R. community. By itself it only considers working flow, but we employ it later as a component subproblem in certain approaches to mesh topology design. FCR also serves well to establish the principle of choosing edges at a "fixed cost" per edge and then determining routing costs over those edges, illustrating the key trade-off between "edge" and "capacity" costs.

The FCR problem is to select a subset of all possible edges and route working flows over them at minimum total cost for flows and edges. More specifically where capacity must be provided on each edge to support actual flows between all node pairs, this is known as the *multi-commodity capacitated "fixed charge"* problem. Multi-commodity refers to the aspect of simultaneous routing for all O-D pairs and a "capacitated" problem is one in which the total flow on any edge is a contributor to cost. A noncapacitated context is one where edges must exist to make routing to all destinations feasible, but there is no specific additional cost that is a direct function of flow over the edges. (Much data network design involves fixed charges to establish an edge, but any edge is assumed to support routing of all data flows at no extra cost.) Fixed charge refers to the discrete "get started" or establishment cost for each edge selected. With the following definitions, the basic fixed charge plus routing problem can be stated as:

- \mathbf{N} is the set of network nodes. N is the number of nodes.
- \mathbf{A} is the set of $(N(N-1)/2)$ possible bidirectional (or *undirected*) edges in the graph on the set of nodes \mathbf{N} . More generally \mathbf{A} is the set of candidate edges and if all possible edges are under consideration, \mathbf{A} is referred to as the *universal* edge set. It is indexed by (i,j) node-name pairs, but only enumerated over $i < j$ in keeping with (i,j) and (j,i) being representations of the same bidirectional edge.
- \mathbf{D} is the set of all non-zero demand quantities exchanged between nodes, indexed by r .
- d^r is the amount of demand associated with the r^{th} demand pair in \mathbf{D} . In the model below, demands are treated as being unidirectional but the unidirectional solution information implies the bidirectional capacity design corresponding to a real transport network.
- $O[r]$ is the node that is the origin for the r^{th} demand pair in \mathbf{D} . $T[r]$ is the corresponding target or destination node.
- $c_{ij} (= c_{ji})$ is the incremental cost of adding one unit of capacity to edge (i,j) .
- F_{ij} is the fixed cost for establishment of an edge in the graph (bidirectionally) between node i and node j .
- w_{ij}^r is the amount of working flow routed over the edge between nodes (i,j) in the direction from i to j for relation r .

- w_{ij} is the working capacity assigned to the edge between nodes (i,j) to support all working flows routed over that edge in the (i,j) and (j,i) directions.
- $d_{ij} (= d_{ji})$ is the 1/0 decision variable indicating whether an edge in the graph is to exist between nodes (i, j) in the design. It equals 1 if edge is selected, zero otherwise.
- W_{∞} is an arbitrary but large positive constant, larger than any expected accumulation of working capacity on any one edge in the solution.

FCR

Minimize

Equation 9.4

$$\sum_{ij \in A} \{c_{ij} \cdot w_{ij} + F_{ij} \cdot \delta_{ij}\}$$

Equation 9.5

$$\sum_{nj \in A} w_{nj}^r = d^r \quad \forall r \in D; \quad n = O[r]$$

Equation 9.6

$$\sum_{jn \in A} w_{jn}^r = d^r \quad \forall r \in D; \quad n = T[r]$$

Equation 9.7

$$\sum_{in \in A} w_{in}^r - \sum_{nj \in A} w_{nj}^r = 0 \quad \forall r \in D; \quad \forall n \notin \{O[r], T[r]\}$$

Equation 9.8

$$w_{ij} = \sum_{r \in D} w_{ij}^r \quad \forall ij \in A | i < j$$

Equation 9.9

$$w_{ij} \leq W_{\infty} \cdot \delta_{ij}; \quad \delta_{ij} \in \{0, 1\}; \quad w_{ij} \text{ integer} \quad \forall ij \in A | i < j$$

The first thing to note is that this is a node-arc or "pure flow" oriented model. Edge decision variables "gate" on or off the ability to send flow over any candidate edge. On the set of currently enabled edges, however, we recognize a pure multi-commodity transportation-like flow structure with source, sink, and transshipment. As a flow-oriented (as opposed to path-oriented) model, we need to again manage a unidirectional orientation to individual links, and recognize that indexing is on node name pairs. An edge ($i \rightarrow j$) is selected into the topology if δ_{ij} is one, in which case the "fixed charge" for the associated edge f_{ij} is contributed to the objective function. Equations 9.5 to 9.7 are the familiar flow-balance constraints of the node-arc transportation problem. They assert, for each demand pair, that total source flow equals the demand, that the total sink flow also equals the demand, and that no net sourcing or sinking of flow for the given O-D pair occurs at any other node (i.e., "transshipment").

The node-arc approach to FCR is essential (and is similarly required in the mesh topology problem) because it is essentially impossible to represent eligible route sets when every edge is itself a variable. (A route enumeration exercise would be required in advance for each possible topology, but there are $2^{|A|}$ possible edge combinations.) As presented, constraints (9.8) are really only the definition of required edge capacity in terms of the simultaneous flows over the edge. As an alternative the cost for these capacities can be referred into the objective function with an additional summation over all demands. The approach above, however, lets us assert integrality on the edge capacities and provides the capacities as explicit output variables. Other versions of the problem may involve a family of capacity units without there being a dominant "get started" edge cost and smaller subsequent capacity unit step.^[3] FCR may be easily generalized to include preexisting edges or already installed capacity on some edges.

^[3] For instance this would be the more common paradigm for private leased-line network design. Each leased-line STS3, 12 or, 48 acquired would have a lump establishment cost with no further cost for its use up to its capacity. There is thus an aspect of fixed charge for every capacity acquisition, rather than fixed charge for edge selection per se with no limit on subsequent capacity but with a cost for each capacity addition on the edge.

An additional constraint that can help solve the problem by eliminating a significant portion of the topology space is:

Equation 9.10

$$\sum_{ij \in A | i < j} \delta_{ij} \geq N - 1.$$

This expresses our *a priori* knowledge that the topology must at least consist of a spanning tree. Note in practise that when we later consider the spare plus working fixed charge problem that separate *a priori* studies may give us opportunity to apply constraints of this type based on knowledge that a survivable result must at least be based on a biconnected graph ($|S| \geq N$). In addition we may also have strong experience or data from prior studies expressing a strong belief about the maximum network average nodal degree that is likely to correspond to an optimum topology. For instance if $d < 5$ is almost a certainty in practice, it allows us to add a further constraint effecting $\leq \text{ceil}[5N/2]$.

Gendron et al. [GeCr99] provide a survey of various formulations and solution approaches for capacitated multi-commodity FCR problems. Gendron points out that it is the mutual capacity constraints that make the capacitated versions of FCR "NP-hard and very difficult to solve in practice." Gendron also explains that one of the difficulties in applying branch and bound to solve FCR problems is that the "strong relaxation" (dropping all integrality constraints, including on the edge variables) gives quite weak lower bounds. This is because the mutual capacity constraints are so crucial to determining an optimal FCR solution. In the unrelaxed FCR problem, the choice of routes for each working flow is strongly coordinated with that of other flows, to use as few edges and capacity units as is optimal. We later see that this is abundantly true of the topology problem involving spare capacity considerations as well. Under any relaxation for 1/0 edge variables to real values, each flow is more or less independently routed since there is no shared efficiency effect from the fixed charge component. In

other words, the solution space to an FCR problem is strongly and discretely structured by the topology variables. If edge variables are relaxed, we have an amorphous sea of uncoupled flows with total costs that are almost completely unrelated to the real problem on a discrete graph. Flows are uncoupled in the sense that each can in effect just pay for the "fraction" of the edges it needs to directly route itself. This is why relaxation of the 1/0 edge decision variables gives an almost meaningless and extremely loose lower bound.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.3 Interacting Effects in Mesh-Survivable Topology

As a first exercise in approaching the problem of designing the physical topology for a mesh-restorable transport network, let us make a qualitative appraisal of all the factors that topology influences. Having stated and discussed these various push-pull effects from a qualitative standpoint, the rest of the chapter endeavors to quantify the various effects and put together models and heuristic solution methods trying to optimize the whole system of interactions for minimum total cost. The factors we identify as interacting with topology are:

- Spare capacity for restorability
- Edge establishment costs
- Working path routing costs
- Modularity and economy of scale effects
- Two-connectedness (or preferably a biconnectivity) requirement
- Access to demand (revenue) sources

(a) Spare Capacity Sharing

One of the most easily and widely appreciated general notions about mesh-restorable networks is that the sharing of spare capacity for restoration becomes more efficient as the network becomes more connected. The $1/(\bar{d}-1)$ lower bound on redundancy ([Section 5.4.2](#)) confirms this intuition. But generally, it is easy to reason that there will be a reduction of spare capacity as average nodal degree, \bar{d} , increases through span additions because greater connectivity allows greater diversity of restoration routes and sharing of spare capacity for restoration. Thus, by itself, the reduction of spare capacity for restorability is an economic push toward *high* connectivity. A related observation is that in general one wants to place spare capacity where the working capacity is *not*. To protect working flows on one span, we want a topology that supports spare capacity rerouting over diverse surviving spans.

(b) Edge Establishment Cost

On the other hand every new span added to the topology has some possibly quite significant one-time establishment cost, or a lease cost associated with rights to use it. Notionally the contribution this makes to total costs would be proportional to the number of spans and their distances. It is under this heading that we find the classical interest in minimum weight spanning trees as the basis for (non survivable) topologies. A minimum cost spanning tree represents the least investment in span establishment costs that supports communication between any two nodes. A tree on N nodes has $(N-1)$ spans, the average degree of nodes approaches 2 as N gets large. Trees, however, are not restorable in our sense. Physical repair is the only restoration mechanism. The corresponding entity for a mesh survivable network (in the sense of minimal edge costs, all nodes connected, and restorable) is a minimum cost biconnected subgraph. Obviously, the establishment of new spans is a cost effect that is directly opposed to the idea of developing a richly connected topology.

(c) Working Path Routing

The next factor to consider is again in favor of more spans, not fewer. Every span we add permits a shortened routing for some number of working paths. A demand flow traversing a 3-hop route A-B-C-D may be converted to a 1-hop routing with the addition of a new span (A-D). This frees transmission capacity on the spans A-B, B-C, C-D and reduces OXC core sizes and interface costs. Traditionally in data, voice trunking, or leased line network design these beneficial routing effects (in the forms of queuing delay, blocking, or throughput), in counterpoise with the route establishment costs, alone determine an optimal topology. This effect should continue to be a significant principle in determining an optimum mesh topology because as the topology becomes more connected the amount of spare capacity diminishes. Total working capacity also goes down as working routes shorten but generally the network as a whole also becomes less redundant as \bar{d} increases. This means that as connectivity increases the spare capacity savings from further increases in \bar{d} are of less economic importance than further savings possible in the working demand flows. In other words, by achieving what we aim for in mesh restoration (to make the spare capacity perhaps only 40% - 60% of the working capacity), at the same time we make further percentage savings on the spare capacity not worth as much as a corresponding improvement in working path routing. In other words, efficiency in spare capacity is important but the leverage on improvements to working route efficiency is always greater because the amount of working capacity usually exceeds the total spare capacity. These considerations lead us to the view that a good FCR solution for the working flows and edges only is still an important part of a good overall topology.

(d) Modularity

Modularity, coupled with economy of scale effects, is another factor counteracting high connectivity. Large capacity modularity, especially with a strong economy of scale, pushes toward sparseness in the facilities graph. Strong economy of scale tends to say "never mind routing efficiency" (in a geographical distance sense): it can be less expensive to group many flows together and take a longer route than to have the extra edges that make shorter routes possible. For this effect to be significant, however, the modular capacities involved must be significantly large relative to the typical working flow between nodes and their aggregation on common routes. If the demand is so high that many large modules are required, the detouring of routes to aggregate into cost effective large modules is not such a pronounced effect and shortest paths over a richer topology can again be preferred. Therefore only "significant modularity" contributes to sparseness. The significance depends on the volume of demand relative to the available capacity modules.

(e) Two-connectedness

Even if the factors pushing for sparseness were infinitely weighted, there is a firm "bottom line" on how sparse a survivable mesh network can be. Any topology we can even consider must be two-connected or preferably biconnected. A two-connected graph provides an edge disjoint pair of paths between every node pair, but may contain articulation nodes (nodes that are single points of failure). A biconnected graph has no such cut-nodes, and by implication has a minimum cut between any O-D node pair that contains two or more edges. $\bar{d} \geq 2$ is thus also a minimum requirement for each node individually. Qualified graphs are easily recognized visually; they are topologically "closed" with no degree one stub nodes and no nodes that are evident pinch points. This class of graph is the conceptual parallel to the family of spanning trees that cover all nodes of a network that does not have to be restorable in our sense.

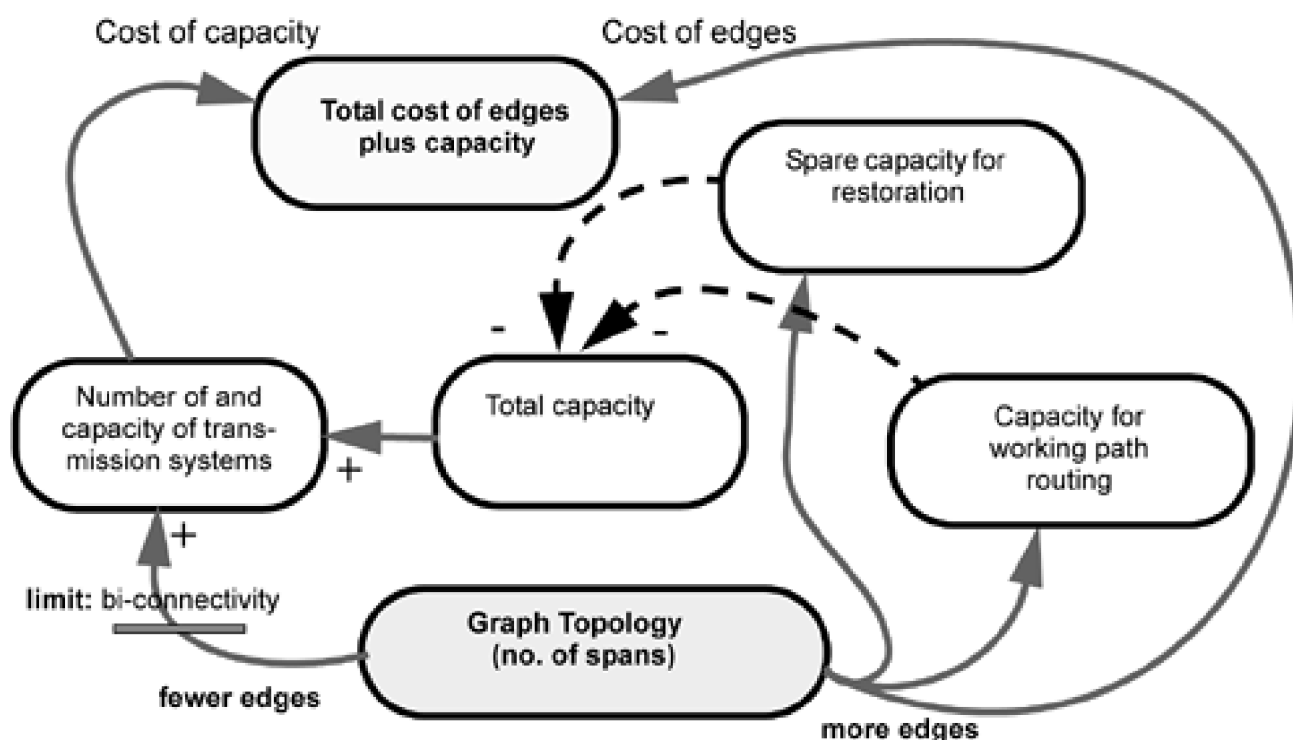
(f) New Nodes and Revenue

In a wider view of the topology optimization problem, especially in context of competitive carriers, one can consider the basic question of what cities should even be on the network. Or, if a given new facility route is added, can new sources of demand be picked up in conjunction with the added route? It is after all up to each operator in a deregulated environment to decide where to offer services and

where not to. This is a major expansion of the overall problem involving topology that brings new revenue considerations into the picture, as well as counteracting cost considerations. This is unlike the other factors listed in that it influences both the number of nodes *and* the number of spans. If we assume that business efficiencies arise in serving more nodes, it will be a tendency toward having more nodes in the graph, but the most efficient way to add connections to many new sites would be by addition of pairs of spans that pick up the new location as an additional degree 2 node. To that extent it would have the net effect of lowering the overall average nodal degree. In the present treatment of the mesh survivable topology design problem, however, we assume that all desired nodes are stipulated and constant among all topology alternatives to be considered. The topology extension problem to include limited node additions is considered briefly.

[Figure 9-3](#) summarizes the main causal relationships we have touched upon. A plus sign indicates higher connectivity, higher capacity or higher cost. The sketch shows ultimately how the choice of physical topology drives both the "edge costs" for establishing the topology itself and, through economy of scale and modularity effects in transmission systems, how it also drives the total cost of needed capacity in a counteracting direction. The best topology leads to a minimum cost trade-off between edge costs and capacity costs.

Figure 9-3. The opposing ways through which topology influences total network cost.



9.4 Experimental Studies of Mesh Capacity versus Graph Connectivity

As a next step let us now look at results from experimental studies focused only on span-restorable networks with varying average nodal degree. Recall from consideration of isolated node conditions in a span-restorable network we deduce that redundancy should fall as $1/(\bar{d} - 1)$. In this section we systematically vary \bar{d} over a range of test graphs to observe the actual dependency of working and spare capacity requirements on nodal degree. The results let us gain several insights that help further prepare us for a formal treatment of the whole design problem including topology as a variable.

The following procedure is a relatively simple systematic way to inspect the dependency of a mesh-restorable network on its topology. In particular, let us assume that we have an existing transport network and want a broad indication of whether the existing graph of facility structures is well suited, too sparse, or too richly connected for the best economic operation of a restorable mesh. The results give an early indication of which basic orientation network evolution planning should take: toward more spans or fewer spans?

Procedure pare-down(initial graph, demand matrix, route and equipment costs);

(Optional preliminary step) To permit a sweep of model network designs that will place the existing network somewhere near the middle of the range of networks generated in the following process, first add one span to each existing node, increasing its \bar{d} from $2S/N$ to approximately $(2S + N/2)/N$. Most of these additions can be generated by making pair-wise selections of existing nodes that are not already adjacent and connecting them with a new edge. If this leaves some nodes without an extra incident edge, such edges are added as needed accepting that other nodes may thereby receive two extra incident edges overall. (This is why the new \bar{d} guideline is approximate.)

Route the demand matrix over the graph in a minimum cost or shortest path manner. This generates the w_i quantities.

Solve the span-restorable mesh SCA problem for the given w_i and topology.

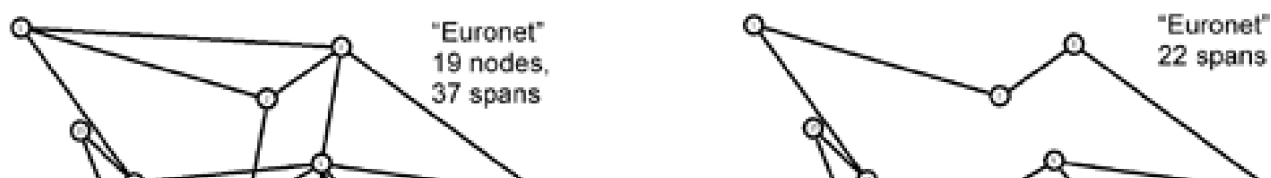
Record the total design cost including fixed costs for edges used, working and spare capacity placements, and record the topology (the set of spans bearing non-zero $(w_i + s_i)$).

Find the span in this design that has $\max(\text{fixed costs}/w_i)$. Other criteria can be considered here as well; simply $\min(w_i)$ is another. Remove this span from the graph. If the graph is no longer biconnected, select the next ranking candidate for elimination instead.

Iterate steps 2- 5 until no further spans can be eliminated without loss of biconnectivity.

This process was carried out on a selection of test networks to obtain general insights about how working and spare capacity requirements of a mesh network separately respond to the graph topology, from sparse to richly connected. [Figure 9-4](#) shows two of these networks in their initial and fully pared-down states.

Figure 9-4. Two test networks subjected to procedure **Pare-down() showing their initial (left) and final biconnected limiting topologies (right).**



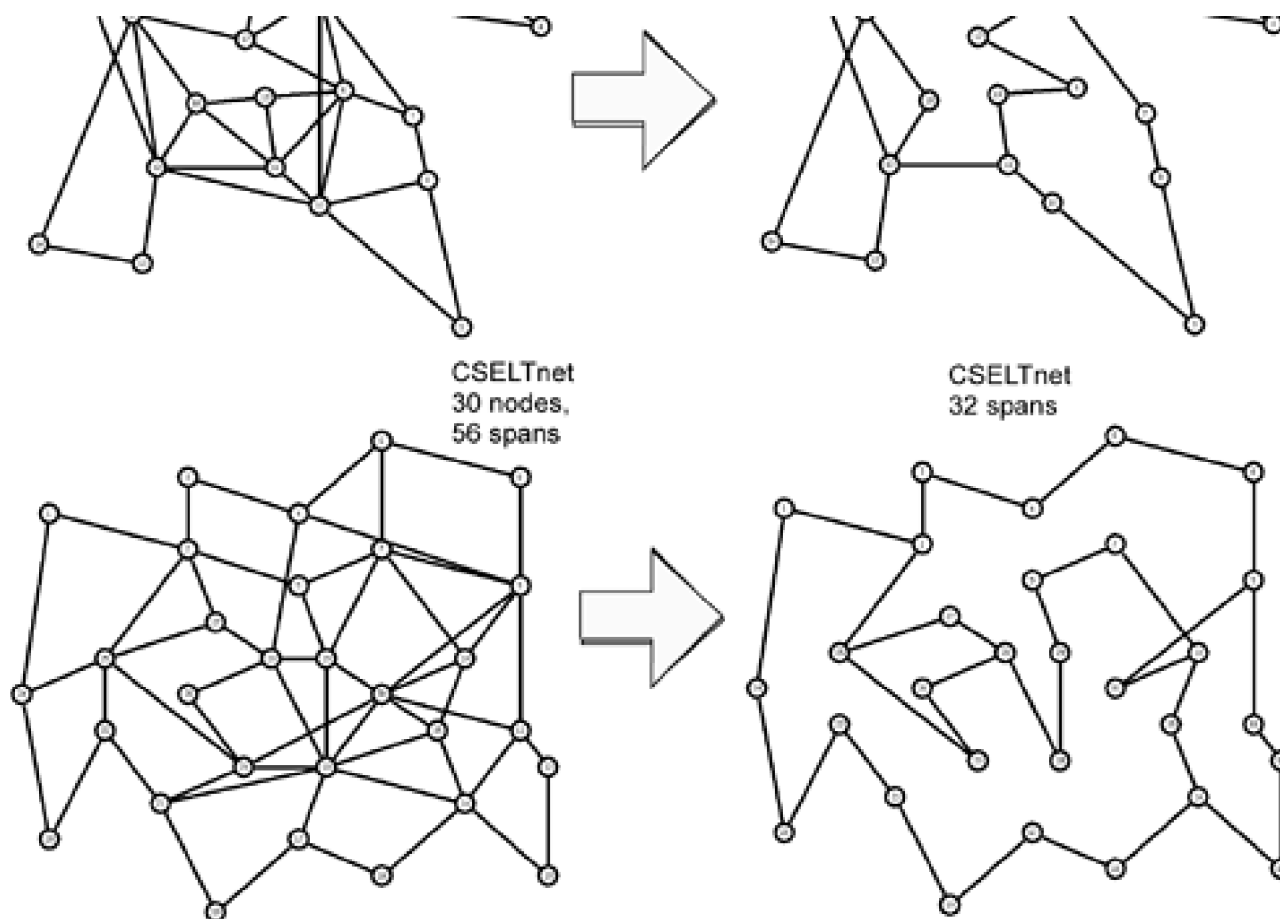
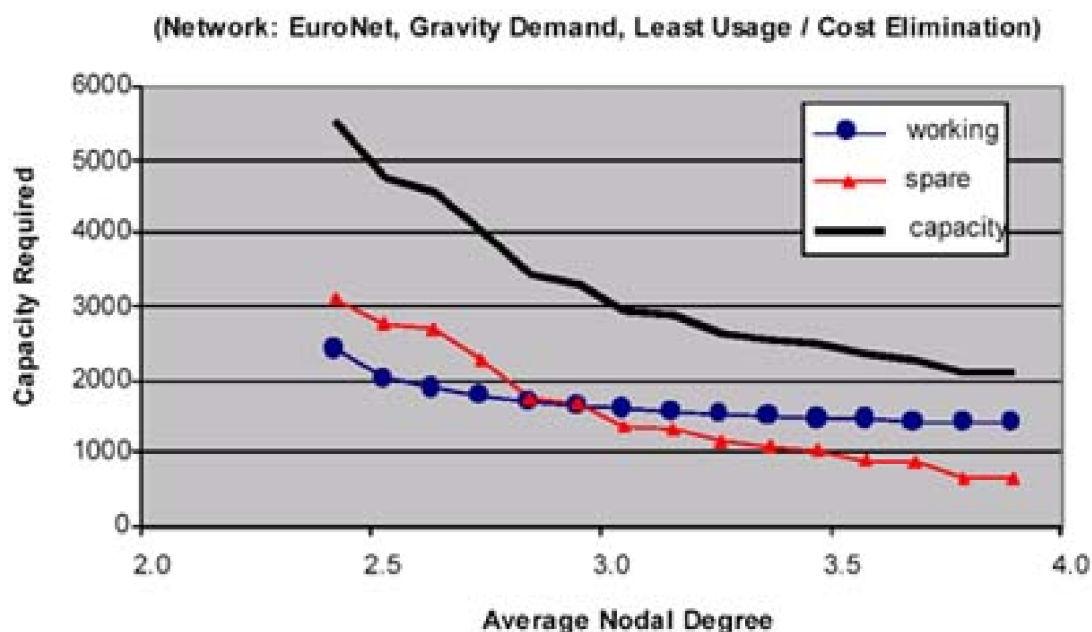
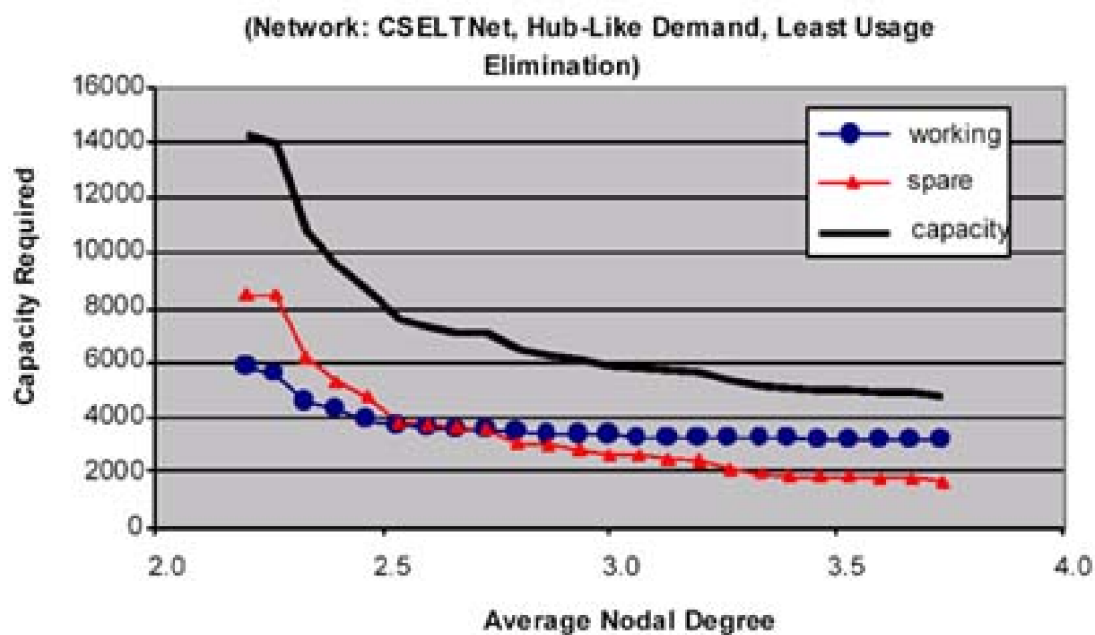


Figure 9-5 shows the working, spare and total capacities for these two networks at the network average nodal degree of each intermediate network state as span eliminations are effected at each iteration. Note that although we present the curves with nodal degree increasing on the abscissa, the curves were actually generated by `pare-down()` starting at the high nodal degree (right-hand side) and moving leftward to produce these results. At each stage of network reduction, the demand matrix was rerouted over shortest paths with a split in total demand quantities for each O-D pair for which there are two or more approximately equal shortest path routes. Each revision of the working path routes was followed by a solution for SCA based on Herzberg's model (Section 5.3.4) with a hop limit of six. The span selected for elimination at each step in the results was the span with lowest w_i . All edge having an assumed equal cost, this is the same as the first (and second) criterion at step 5 above. Ties were broken arbitrarily.

Figure 9-5. Working, spare, and total capacity response to systematically varying topology.





The resulting curves varied in detail but were remarkably consistent in a number of attributes, regardless of the demand pattern or exact sequence of removals to reduce the topology. These were:

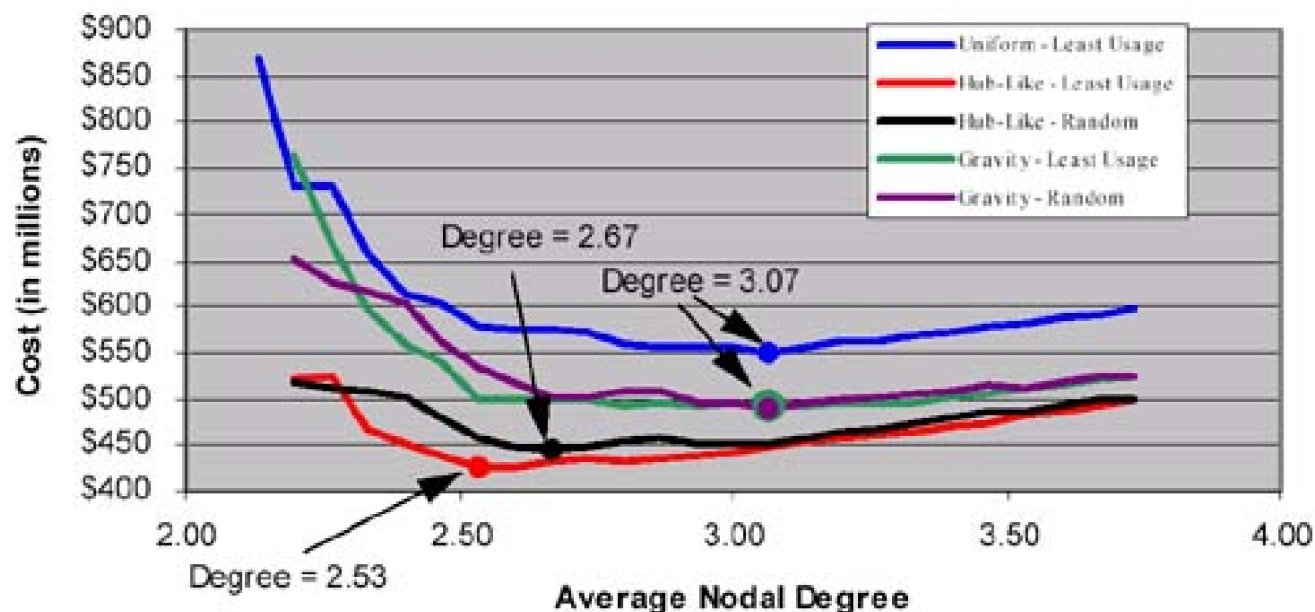
1. Both working capacity and spare capacity decrease monotonically with increasing network degree.
2. The spare capacity requirement always drops more quickly than the working capacity, as nodal degree increases.
3. With remarkably little variation, the cross-over point (where spare capacity drops below working capacity) occurred in these test networks at $\bar{d}^* = 2.6$ to 2.8 .
4. Below this \bar{d}^* the total spare capacity required for survivability exceeds the working capacity being protected.
5. The total working capacity requirement was often nearly constant or decreasing quite slowly shortly after the crossover point.
6. The total spare capacity often went on dropping significantly to well past $\bar{d} = 3.5$.
7. A numerical fit to the ratio of spare to working capacity is consistent with the $(\bar{d} - 1)^{-1}$ functional form.

Some insights, explanations and observations are as follows: Spare capacity drops more rapidly than working because it benefits in two ways from higher connectivity. First, in the presence of a fixed hop limit the diversity of eligible routes over which spare capacity sharing occurs increases nonlinearly and, second, the average number of working paths crossing each span to be protected decreases as working paths shorten. Unless the fixed charge for spans was high relative to capacity routing costs, a mesh network should generally have a degree of 2.8 or higher because it is above this crossover point that the efficiency in spare capacity drops well below 100% redundancy. Finally, the fact that the capacity curves are remarkably similar under least usage (or random elimination—see below) suggests that the process of routing and spare capacity placement is quite accommodating of minor detailed differences in the topology. The total cost of capacity is much more dependent on the total number of edges present (in a biconnected configuration) than on precise details of exactly which spans are present. This is an early indication that heuristics that greedily consider one span addition or move at a time may perform well in topology searches. (If the curves of [Figure 9-5](#) had varied more greatly depending on the exact sequence of span removals, we would have a converse indication that global optimization of the exact span combination was crucial.)

In the results so far presented we are studying only the response of the working and spare capacity requirement to the network degree. So how do curves such as in [Figure 9-5](#) lead to an optimum topology choice? The location of an optimum topology depends on the fixed edge costs that are added to the capacity costs. While capacity-dependent cost is monotonically decreasing as degree increases, each edge adds a fixed charge. Here, for discussion, all spans are assumed to have equal-edge costs, so to the curves of [Figure 9-5](#) is added a ramp of total-edge costs. The result is the creation of a cost minimum from the superposition of a linear rising term and a $1/d$ -like decreasing term. [Figure 9-6](#) illustrates using the CSELT network and assuming the edge cost is equivalent to 360 channels on a span. This ratio, called Ω , means that the cost of establishing the facility route was equivalent to the cost of putting 360 channels into service on the edge. In [Figure 9-6](#) the fixed charge is assumed to be \$24,000 per km and all spans are 300 km in length (implying a per-channel cost of \$66.7 per km for $\Omega = 360$). Obviously edge costs can vary enormously in the real world—from nearly zero for a pipeline utility with existing

rights-of-way, poles and ducts, to the fixed charge being equivalent to thousands of subsequent capacity units where city streets are ripped up to pour new concrete ducts but channels once installed may not even need amplifiers (because metro inter-office distances are short). In [Figure 9-6](#) the curve for "hub-like demand— least usage elimination" corresponds to the topology family and total capacity curve from [Figure 9-5](#) for CSELT, with edge costs added as detailed ($\Omega=360$). The other curves in [Figure 9-6](#) show how the location of the optimum also varies significantly with the demand pattern. The types of demand pattern (hub, uniform, gravity) are those described in [Section 1.5.5](#). Least usage or random refers to how the pare-down procedure operated to generate the family of lower-degree test topologies that each curve is based upon.

Figure 9-6. Total cost curves for both capacity and edge costs for span establishment with demand pattern as a parameter (for CSELT Network model).



If span establishment costs are large relative to capacity costs on established spans, then we expect a well-defined optimum, occurring at relatively low average nodal degree. If span establishment costs are small, the optimum region becomes broad and shallow and spread over a range of higher degree solutions. In intermediate ranges both fixed and incremental capacity costs are significant factors, such as in [Figure 9-6](#). Although it is tempting to note that the optimum region still looks shallow and broad, the absolute scale of the costs involved still motivate detailed design optimization efforts. For instance in [Figure 9-6](#) the difference between the optimum topology for uniform demand and a hub-like demand pattern, both found under a least usage paring-down experiment, is a matter of about $N/2$ fewer spans required in the hubbed model (e.g., a difference of about 0.5 in d) and a cost difference of over \$100 million. Note that whether edge or capacity costs dominate the total cost, and hence push for a sparser or richer graph, also depends on the total volume of demand and the pattern of demand. In [Figure 9-6](#) the uniform demand pattern gives every O-D pair 100 units of demand and is best served with a degree 3 connected graph. When nodes have demand only to two predefined hub locations, (in the "hub-like" demand pattern) the optimal connectivity drops to $d \sim 2.5$. The total cost is lower because less demand is served in total but the optimum region is more sharply defined. A final note is that despite absolute cost differences, three of the six test cases are all best served by a single topology that is encountered at $d \sim 3.07$. This is our first indication of how there tends to be certain discrete topologies that are optimal for a whole range of local conditions in terms of changes in Ω demand, or demand pattern. The optimal graph requires a more significant change to these factors to cause a jump to another discrete configuration of higher or lower connectivity (or detailed construction).

9.5 How Economy of Scale Changes the Optimal Topology

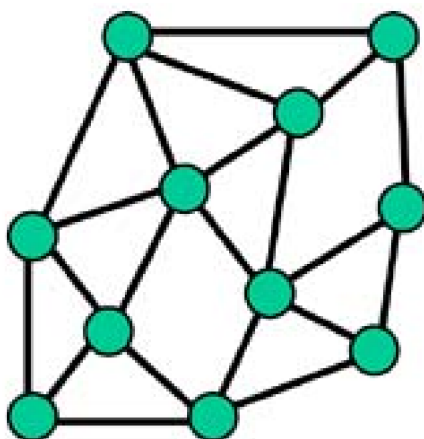
Experience with transmission and multiplexing technologies is that one usually gets n times the capacity for less than n times the price, relative to the preceding technology. Typically SONET systems on the OC-24 to OC-192 range exhibit economy of scale in the range of 3 times capacity for 2 times price ("3x2x"). It was shown in [DoGr00] that such economy of scale and modularity effects are worth taking into account directly in a span-restorable mesh network design because of the lower cost of a truly modular design formulation, compared to the simple integer-capacity design approach with after the fact modularization. The greatest improvements in design efficiency and costs were found with the modular JCP (MJCP)^[4] design formulation and there were clear indications that MJCP's effectiveness was attributable to both working paths and restoration routes being carefully aggregated and coordinated to fit together in well-filled, cost-effectively chosen modules of the largest sizes that can be economically exploited. In the MJCP formulation, one of the main effects observed is the tendency for working paths and restoration routes to go "out of their way," and be coordinated to take advantage of design opportunities for larger and more cost efficient modules.

^[4] Acronyms JCP and MJCP ("P" for placement) are used here for consistency with the original source [DoGr00], but are synonymous with JCA and MJCA ("A" for assignment) used in Chapter 5 and elsewhere in the book.

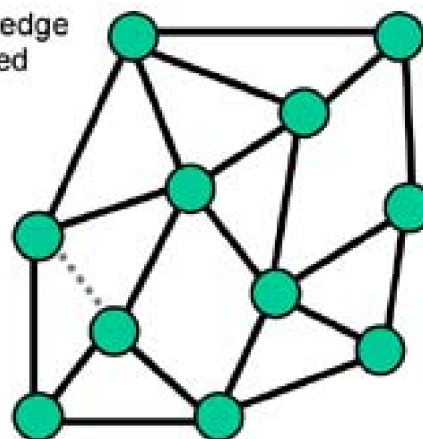
This effect can go to the extent of totally disusing one or more spans. This occurred even under the weakest economy of scale model (3x2x). The explanation is that with the greater economy of scale and modular capacity available, the more incentive there is for a working path or restoration route to detour to help fill a larger module. The cost-optimal solution is characterized by use of the fewest, largest modules, with the routings being subservient to this internal dominance effect. In contrast a capacity minimal solution is dominated by shortest path routing considerations. Only when capacity cost is linear (and non-modular) is the minimum-capacity solution also the cost-optimal solution. The minimum *capacity* solution always benefits from greater network average nodal degree but the modular minimum *cost* solution can counteract and overtake this effect under sufficient economy of scale effects. An example is shown in figure 9-7 of how the subgraph associated with the cost optimal solution evolves (for network 11n21s2) as the economy of scale effect in capacity increases.

Figure 9-7. Letting economy of scale effects identify a core topology under MJCP [DoGr01].

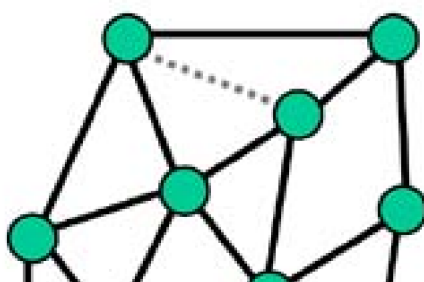
(a) Set of all edges



(b) 1 edge omitted



(c) 2 edges omitted



(d) 6 edges omitted

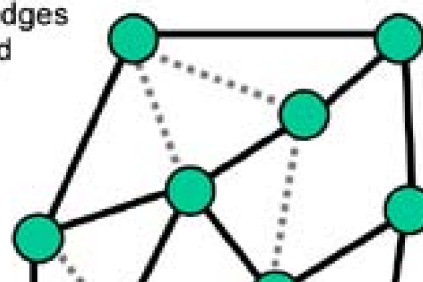
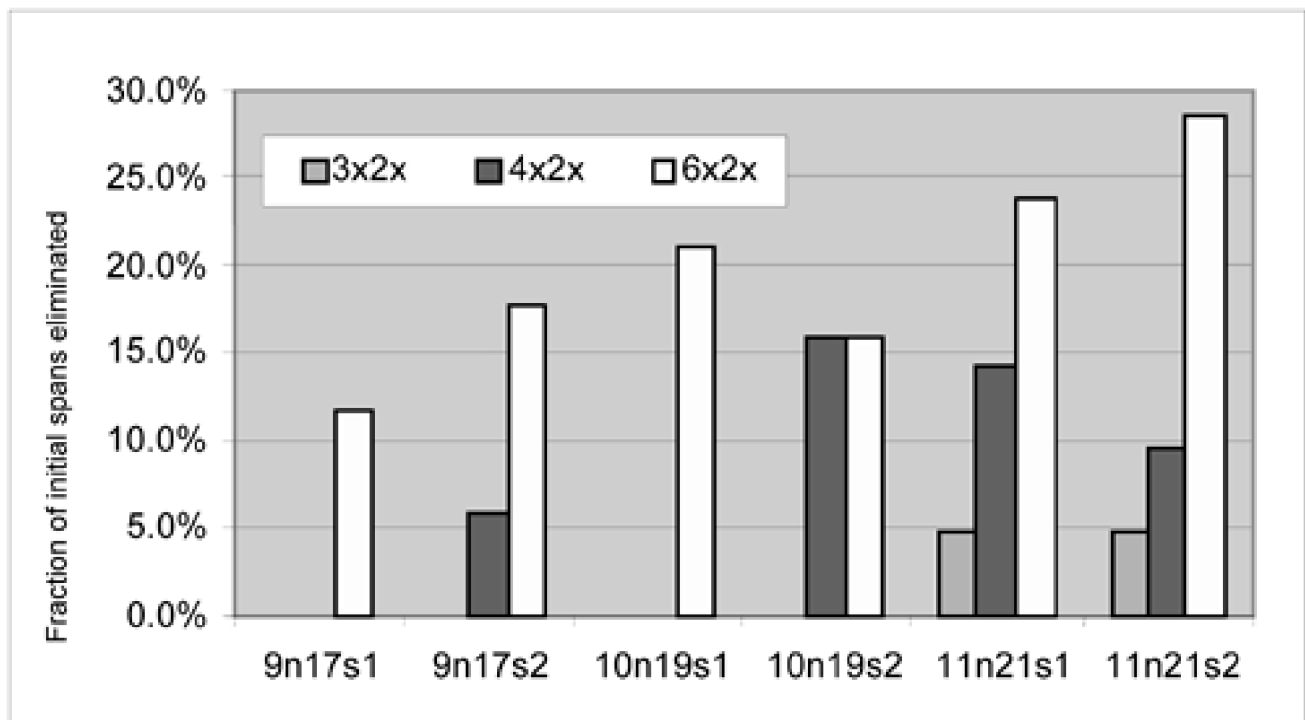




Figure 9-8 summarizes the extent to which economy of scale effects selected a more reduced subgraph as its operating topology in the other test cases.

Figure 9-8. Extent of the topology selection effect as economy of scale increases under MJCP design on several test networks.



"Core Topology" Identification through Economy of Scale

Note that even for small test networks, with the 3x2x economy of scale model which is thought to be fairly realistic, some spans are eliminated in the MJCP model. The greatest impact is with the extreme "what if" case of 6x2x where an average of 19.8% of the spans are eliminated, and one network (11n21s2) saw a 28.6% reduction in spans. Other inspections of the results suggest that the effect is greater in larger networks and when larger hop limits are used, both of which are also consistent with the explanation above. The observation of designs realized on a topology that is a subgraph of all the possible spans present suggests one strategy of implicit identification of a core topology. This topology-sparsening effect increases as the economy of scale strengthens.

A multi-step growth planning principle, involving topology augmentation [SrSo02] has also been based on these effects. The planning process starts with a view of a future demand and topology and works backward to identify a core topology through economy of scale reduction effects such as we see in Figure 9-7. The idea is that the initial network to build is the core component identified in this way, and as growth materializes we augment it out toward (but not necessary all the way to) the full topology initially assumed. In this framework the initial full topology would represent the set of all possible or available edges that the network could use. For instance in Figure 9-7, the graph (d) is interpreted as being comprised of the most fundamentally important or core set of edges for efficient routing and protection of the given demand pattern, drawn from the set of possible edges in Figure 9-7(a). This foundation network would be built initially. Then at a certain point as demand grows the four edges in (c) that were not present in (d) would be commissioned. Subsequently the two edges in (c) would be added, followed by those indicated in (b). In practice, this "backward" planning approach to adding edges as growth materials could be frequently rerun based on actual demand and transport cost evolutions with time, repeatedly checking how well suited the current

topology is (i.e., does the extreme economy of scale model tend to isolate a core set of edges that are in the present network?) and producing a view of what specific next-span additions to consider further.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.6 The Single-Span Addition Problem

One approach to topology evolution is to plan single incremental span additions to an existing network. If there are N nodes and S existing spans, then

Equation 9.11

$$Y = \frac{N \cdot (N - 1)}{2} - S$$

is the number of single-span addition projects that could be considered. A degree 2.5 network of 100 nodes (which would have 125 spans) would thereby strictly pose 4,825 single-span addition possibilities to test. Testing each case is itself fairly straightforward. If a non-joint mesh design was the goal, one would just add the new span, re-provision demands on the shortest routes, update the eligible restoration route sets, then solve an SCA problem for the spare capacity. For joint optimization, one would update both eligible working and restoration route sets with the candidate span in place and run the JCA model. The merit of each new span candidate is the difference between its own construction or acquisition cost and the reduction in total network capacity cost that it provides.

Of course many of the 4,825 possible single-span additions could be ruled out quickly by inspection because they reach across the whole graph, or can be disqualified based on many other practical factors. Nonetheless, the number remaining for detailed cost-benefit assessment could clearly still be daunting. There is a role, therefore, for some heuristic principles to identify a "short list" of the more promising prospects, reducing the list of cases to evaluate in detail.

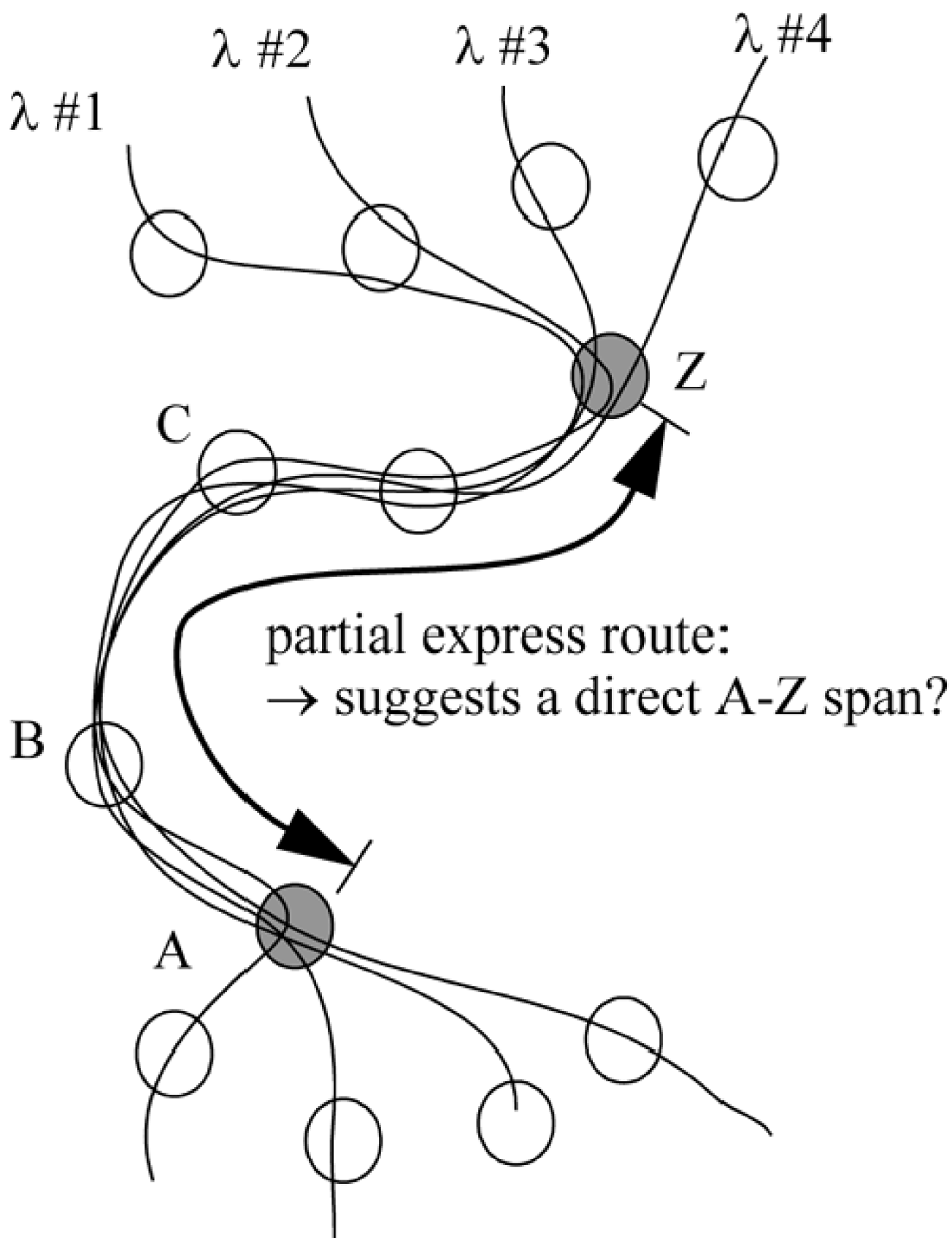
In this section we describe two heuristic search procedures that serve to identify promising new span additions based on considerations of routing and topology in the existing network and how they relate to a prospective new span. One procedure takes a working-oriented viewpoint, suggesting new graph edges that might be especially efficient in conveying working flow in a more direct way. This is the search for "partial express" routing opportunities. The second procedure assesses the extent to which a new span, if added, would support greater sharing of spare capacity for restoration. The following metrics and the overall approach was originally devised for use in an unpublished planning study where the set of possible single-span additions was over 100. These methods were used to reduce the number of cases to study in detail to under ten.

9.6.1 How "Partial Express" Flows Can Suggest New Spans

In "partial express" routing, the idea is to identify particular subsets of demands that maintain their integrity as a group of co-travelers over some contiguous sequence of spans en route [MaGr97b]. No two such demands necessarily have the same origin or destination nodes and no hub is used to effect such aggregation. Rather, demands follow their shortest (or otherwise assigned) route and the resulting composite pattern of flows is inspected for naturally arising groups of demands that share a suitably long segment of common travel.

Figure 9-9 is an example where four lightpaths have different sources and destinations, but are coherent as a group over four spans of travel en route before the group dissolves at nodes A and Z. The demands each have different origin-destination nodes, so they do not individually warrant a dedicated express system. This is why we call them "partial" express flows. They are just natural groupings of demand flow arising over several hops of a route segment that happens to be common to the shortest path routing of these demands over the graph.

Figure 9-9. Partial express routing opportunities can suggest an efficient new span addition.



In work done on partial express routing in [MaGr97b], the intent was to find groupings of such co-routed demands that would be more economically served by dedicated transmission systems that would follow the same physical route but not pass through the OXC nodes en route. The cost of the express fiber, waveband, or lightpath treatment (at possibly lower utilization) could be compared against the cost of routing the same demand segments through OXC-managed paths to discover cost reduction opportunities. For instance, if a 48-l fiber system was loaded with (say) 40 or more in-service Is, over the coherent segment of 4 hops in Figure 9-9, requiring line terminating equipment only at nodes A and Z (and line amplifiers as dictated by the distances), this would eliminate port and core usage costs at three intermediate OXC nodes. On the other hand, by routing through the OXC at every node, lightwave capacity on spans is constantly

groomed and usually can operate at higher utilization. That trade-off was the main prior interest in partial express routing opportunities.

The present interest in partial express phenomena is that we can interpret them as indications of where a *new* span addition seems to be wanted by the natural flow of demand on the existing network. Imagine, for example, in [Figure 9-9](#) if the amount of co-routed flow was found to be 100 lightpaths and they were co-routed over a long indirect route between A to Z. A new A-Z span may be a good candidate to prove-in economically. Our aim at this stage is not to assess the economics in detail but only to produce a ranked list of new span opportunities, assessed in terms of the partial express magnitude times distance product associated with them. A procedure to find such highly ranked opportunities is as follows:

1. Start with the existing network with all demands routed via shortest paths, or by whatever routing policy is the norm for the given network.
2. Create a master list of the route vectors taken by each demand over the network.
3. Pick a node x to serve in the role of node A in [Figure 9-9](#). Mark the node as having been visited.
4. Delete or ignore all routes in the master list not containing node x .
5. Find the statistical frequency of each other node name in the network among the remaining route vectors.
6. For each other node y with non-zero statistical frequencies form the partial express demand-distance measure:

Equation 9.12

$$\psi(x, y) = d(x, y) \cdot I(x, y)$$

where $d(x,y)$ is the shortest path distance on the graph from node x to node y and $I(x,y)$ is the statistical frequency of node y in routes that travel through (or originate at) node x .

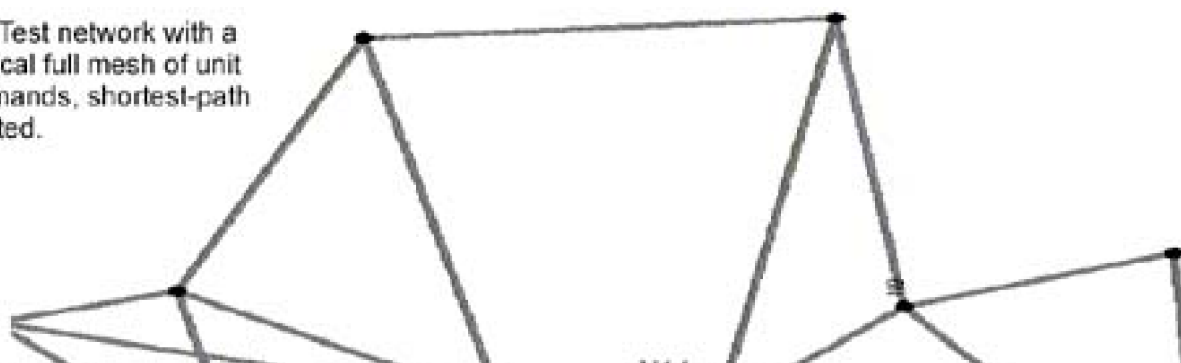
7. Repeat steps 3-6 for each node in the role of the node A.

At the end of this analysis one has an $N \times N$ matrix of $\psi(x,y)$ values. For each node x , the $\psi(x,y)$ values list the distance-magnitude product of demands flowing over the network that traverse node x and also go at least as far as node y . In the context of single-span addition planning, the proposition is that the top-ranked $\psi(x,y)$ values suggest the detailed testing of a new direct span addition between nodes x and y .

[Figure 9-10 \(a\)](#) shows a test case of an example network in which one unit of demand is assumed between all nodes (a unit-logical full mesh). Demands are shortest-path mapped onto the initial graph, then the above analysis is performed. [Figure 9-10 \(b\)](#) shows the actual result for $I(x,y)$ where node x is the circled node (N2). The histogram of other nodes on the routes of demand flowing through that node clearly suggests three new span possibilities. Each bar is related to the specific new span to which it would correspond. All three of these new span candidates could be placed and tested in detail or, if further winnowing is desired, the $I(x,y)$ data can be weighted by the shortest-route distances $d(x,y)$ and compared the length of the new span itself. An example of this further decision process is shown in [Table 9-1](#).

Figure 9-10. Sample results of a partial express flow analysis to suggest new spans.

(a) Test network with a logical full mesh of unit demands, shortest-path routed.



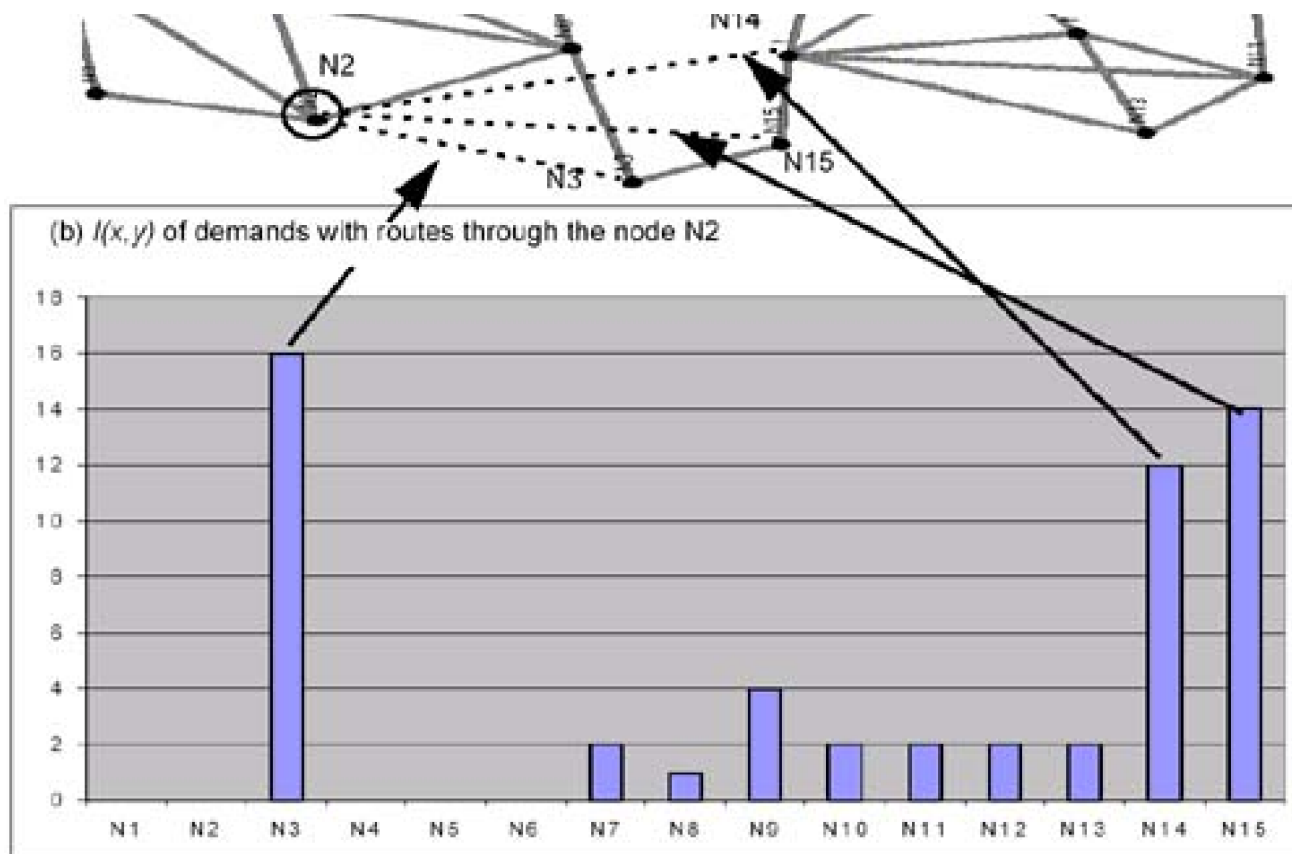


Table 9-1. Assessing working capacity-distance savings of new span candidates

New Span Candidate	Other Node	Existing route $d(x,y)$	$l(x,y)$	Length of new span $d^*(x,y)$	Net capacity-distance reduction $l(x,y)[d(x,y)-d^*(x,y)]$
1	N3	6	16	3.5	40
2	N15	8	14	5	42
3	N14	10	12	5.5	54

9.6.2 Frequency and Remoteness Metrics for Prospective Span Additions

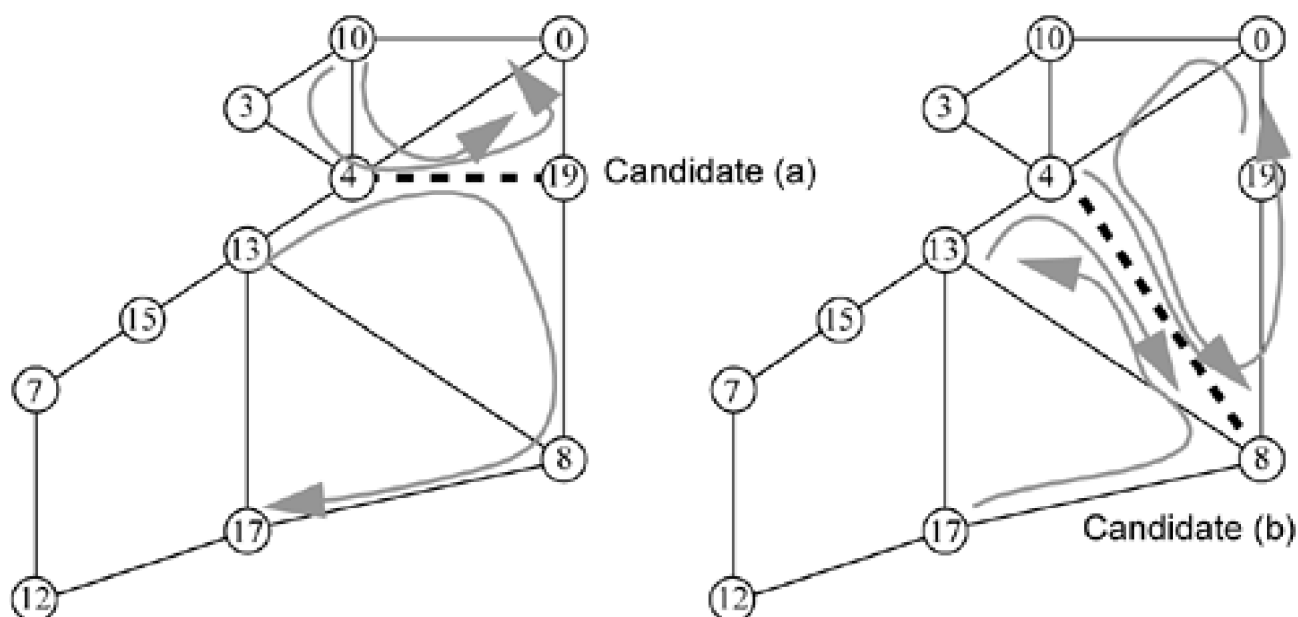
Finding strong partial express flows as indicators of promising new span additions only considers working-oriented factors. But for maximal efficiency in adding a span to a mesh-restorable network, the new span should ideally also offer new efficiencies in the sharing of spare capacity for restoration. This section describes a way of assessing new span additions from the latter standpoint. There are two basic elements to the assessment:

- a. *Frequency*: How often does the span appear in the lists of eligible restoration routes for other existing span failures?
- b. *Remoteness*: How far away is the span on the restoration routes of which it would be part from the corresponding failure spans?

The idea behind these metrics is to discover new edge candidates that, if placed, would have good sharing of spare capacity allocated to them, without requiring long restoration paths to facilitate such sharing. The general notions is illustrated in [Figure 9-11](#).

Figure 9-11. The idea of frequency and remoteness for assessing potential of a new span from a

spare capacity standpoint.



- Which new span addition will shorten the most restoration routes?

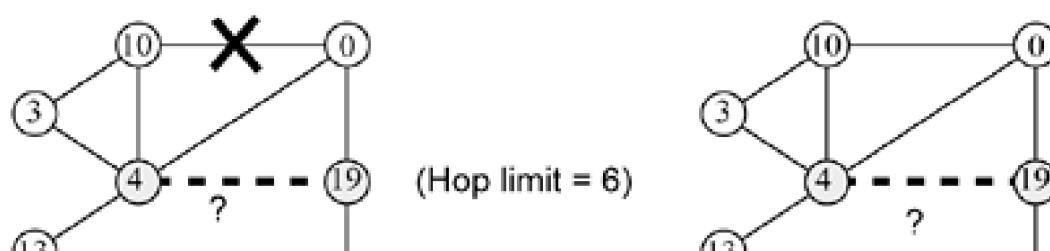
Of course the real answer as to the effect of a new span on spare capacity can be answered exactly by inclusion of the span in an updated JCA or SCA solution. But our aim is to find a surrogate indicator based only on quickly computed considerations for use in networks where there may be thousands of new span possibilities. These heuristics for identifying "promising" candidate edges will also later play a role in populating candidate edge sets for MIP-based solutions of the topology selection problem.

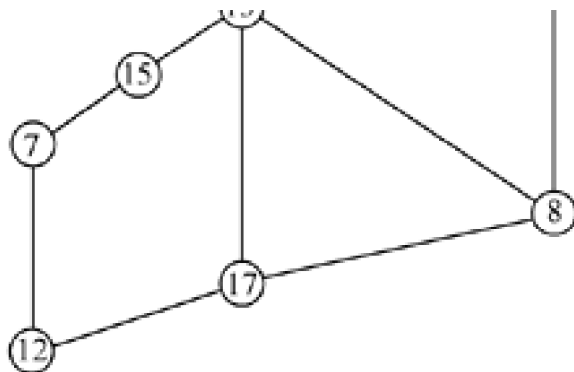
In [Figure 9-11](#) two edge candidates are shown along with their relationships to the routes eligible for restoration of two spans chosen for the example (13-17) and (10-0). The example assumes a hop limit of six for restoration. In [Figure 9-12](#) we see that out of four distinct eligible routes for failure of span (10-0), the (4-19) node-name-pair appears twice. This implies that if an edge (4-19) were created it would enable at least two new route entries in the eligible route set for that span failure. Moreover, we can also see the extent of restoration route shortening that would occur from the routes in the list where the node-name pair appears. With respect to span failure (10-0) two routes would be shortened by three hops each. And with respect to failure span (13-17) we see that new span (4-19) would shorten one route from 6 to 4 hops. This is the aspect of frequency in assessing how frequently a prospective new edge would have a shortening effect on eligible restoration routes. Formally we define:

Equation 9.13

$$Freq(x, y) = \sum_{j \in S} \sum_{i \in P_j} \delta_i(x) \cdot \delta_i(y)$$

Figure 9-12. Assessing the frequency with which a new span (4-19) would appear in the restoration routes for span failures (10-0) and (13-17).

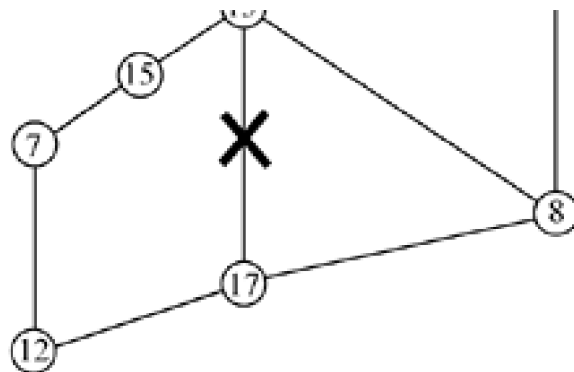




Eligible restoration routes for 10-0

- 10 - 4 - 0
- 10 - 3 - 4 - 0
- 10 - 4 - 13 - 8 - 19 - 0
- 10 - 3 - 4 - 13 - 8 - 19 - 0

Frequency = 2 out of 4
hop-savings = 2 out of 5, 2 out of 6



Eligible restoration routes for 13-17

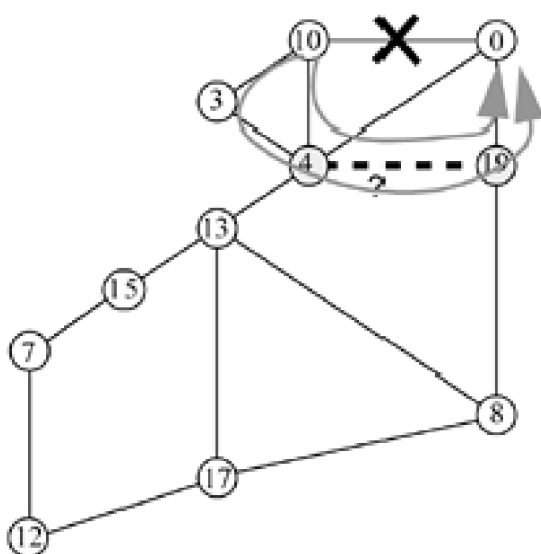
- 13 - 8 - 17
- 13 - 15 - 7 - 12 - 17
- 13 - 4 - 10 - 0 - 19 - 8 - 17

Frequency = 1 out of 3
hop-savings = 2 out of 6

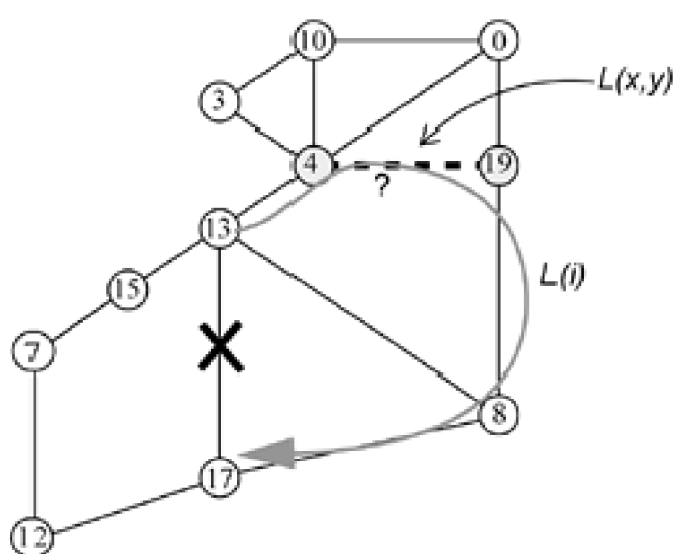
where x and y are the end nodes of a prospective new span, S is the set of all existing spans (more generally the set of all failure scenarios of interest in the existing network), P_j is the set of eligible restoration routes for the j^{th} failure scenario, and $d_i(x)$ is 1 if node x is present in the i^{th} route-vector of P_j and zero otherwise.

The remoteness is also relevant, however. Remoteness characterizes how much spare capacity use is implied to get from the failure span to where the new span would have its potential benefits of shortening restoration routes. For the same examples [Figure 9-13](#) shows how the remoteness measure would be assessed based on the new restoration routes a (4-19) edge would provide. In a hop-oriented measure, remoteness is defined as the sum of the hop counts on the route leading up to and away from the prospective new span. This can be directly converted to a geographical distance context as well. The idea of assessing remoteness is that since these are all considerations on the eligible route sets for solution of a detailed spare capacity assignment, we want some way to derate the merit of a new span candidate where its effect appears only in the longer resultant eligible routes. These routes are less likely to have an important effect in spare capacity sharing and total spare capacity requirements. A new span that has a remoteness of one hop from one of the failures under consideration (i.e., it is part of a new 2-hop eligible route) has greater significance than a span with the same route-shortening effect but higher remoteness.

Figure 9-13. Assessing the remoteness at which new span (4-19) would have its effect on restoration routes for span failures (10-0) and (13-17).



Eligible restoration routes for 10-0



Eligible restoration routes for 13-17

$$\begin{aligned} \text{(a) } & \bar{10} - 4 - 19 - 0 \\ \text{Remoteness} & = 1 + 1 = 2 \\ \text{(b) } & 10 - 3 - 4 - 19 - 0 \\ \text{Remoteness} & = 2 + 1 = 3 \end{aligned}$$

$$\begin{aligned} \text{(a) } & \bar{13} - 4 - 19 - 8 - 17 \\ \text{Remoteness} & = 1 + 2 = 3 \end{aligned}$$

In a formal measure of remoteness we need to consider that an (x,y) pair with high frequency and low remoteness on many routes should not accumulate this remoteness at its expense relative to an (x,y) pair that appears in perhaps only one other route. For this reason the remoteness measure of a prospective new span is not considered in isolation of its frequency. We can also note that the remoteness value of an edge candidate with respect to any one restoration route is always just the hop or distance length of the complete route, less than that of the prospective new span itself. Accordingly we define the *average remoteness* of a prospective new span between x and y as:

Equation 9.14

$$Rem(x, y) = \frac{\sum_{j \in S} \sum_{i \in P_j} [L(i) - L(x, y)]}{Freq(x, y)}$$

where $L(i)$ is the length or hop count of the i^{th} route-vector of P_j and $L(x,y)$ is one if the metric is hop count-based or if length-based $L(x,y)$ is the length of the prospective span (x,y) . Because the indexing ranges of both double sums in [Equation 9.13](#) and [Equation 9.14](#) are the same, it makes sense to compute both $Freq(x,y)$ and $Rem(x,y)$ in one pass over the set of eligible routes for each failure scenario with respect to one prospective new span candidate.

Note that remoteness is not a measure of route shortening effects per se, but only of the total distance away from the failure span at which the new span would be employed—enabling new (shorter) restoration routes. A route *shortening* measure—although notionally tempting—is not really advantageous or required because all longer routes (up to the hop limit) are still present in the list of distinct eligible routes, even if the new span is added. The important point is that new shorter routes are added, not that existing routes are shortened. All existing routes continue to exist as options, at their initial lengths. So what is important is the length of these new routes themselves, not how much they might have shortened other routes. In this regard it can be seen that the length of these new routes are just $Rem(x,y) + L(x,y)$ so in fact it is *remoteness* (not "route-shortening") that is important to characterize the benefit of the new span candidate.

Note also that these surrogate metrics for spare capacity efficacy are not computed on the eligible restoration route sets as they actually would be with the prospective new edge in place. All assessments are based on properties of routes found in the existing network without any edge changes. The advantage is that a complete set of eligible restoration routes is only generated once. If we actually add each new edge candidate and recompute the eligible route set in its presence, we would discover additional new routes made eligible by virtue of the new edge bringing the routes under the initial hop limit. We do not discover such new routes with the approach above, only shortening effects on routes that were initially under the hop limit in the baseline network. The problem with adding each edge and recomputing all restoration route sets is the obvious one of computational effort. With this in mind, however, it suggests that we should generate the single set of eligible routes that is used from the initial network with as high a hop (or distance) limit as can be managed (higher than the limit anticipated for actual network operation). We then compute all metrics on that one route set, without regenerating the route set with every possible nonadjacent node pair present. By representing initial routes of higher than normal hop limits, however, new restoration routes that come into existence by shortening from the added edge tests can be discovered.

9.6.3 Overall Study Technique for Single-Span Additions

Recall that the overall aim in this section is to "short-list" relatively few edge addition candidates for detailed assessment, in cases where there may be thousands of single-span addition possibilities to otherwise consider in detail. Thus, measures of partial express flow, frequency, and remoteness are *a priori* measures because they only claim to detect potential merit in either a working or spare capacity efficiency sense. The measure of ultimate significance is how the total network costs change when the candidate span is included in a revised JCA (or SCA) design problem, and how this cost reduction compares with the span establishment cost itself.

There are several ways the individual $y(x)$, $Freq(x,y)$ and $Rem(x,y)$ metrics can be used at this point. Ideally we want to further consider

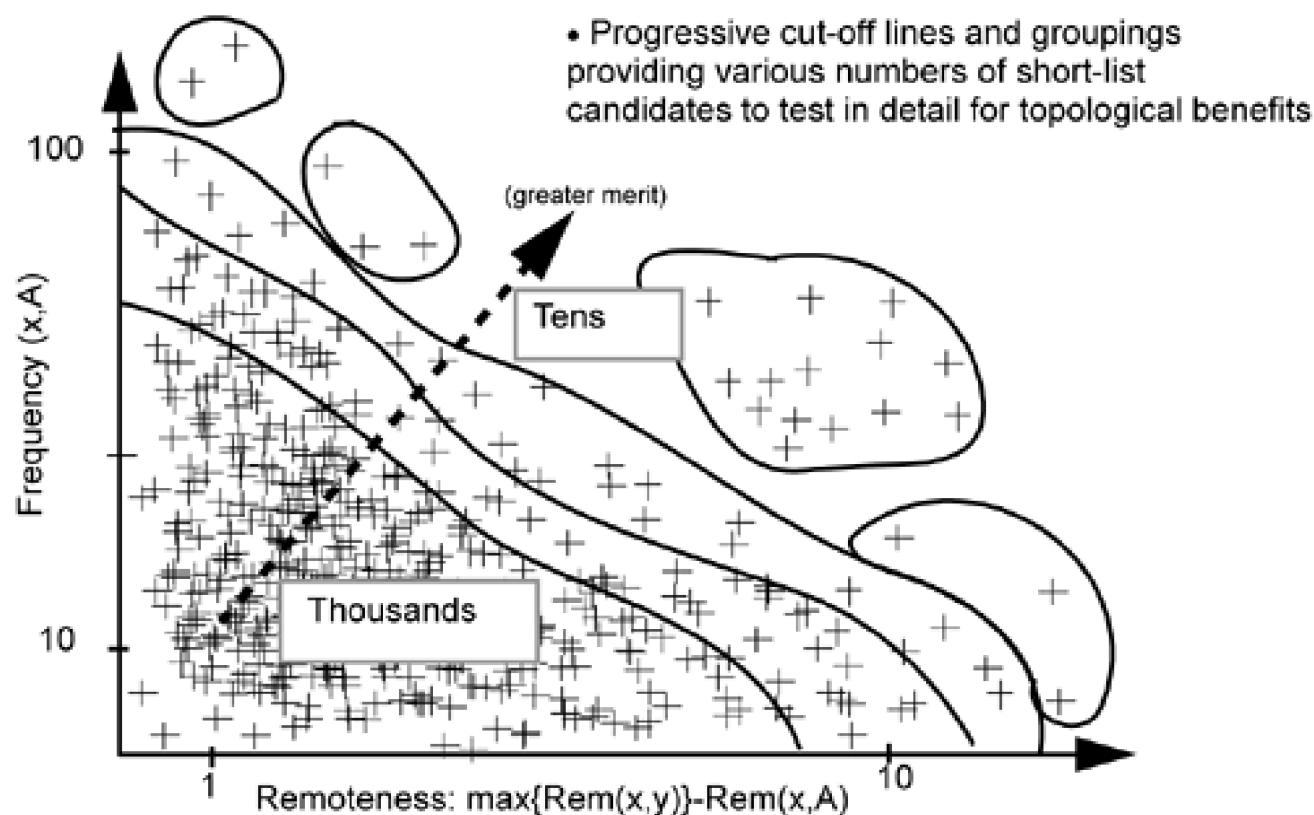
only those (x,y) edge candidates with high $y(x,y)$, high $Freq(x,y)$, and low $Rem(x,y)$. It is possible to combine all three individual metrics into vector-distance notion of composite merit, by "reversing" the remoteness axis and taking the vector distance to each point in the triple-space, i.e.:

Equation 9.15

$$M(x, y) = \sqrt{\psi(x, y)^2 + Freq(x, y)^2 + [\max\{Rem(x, y)\} - Rem(x, y)]^2}$$

where $M(x,y)$ is the combined measure of the (x,y) node pair combination as a new span addition and $\max\{Rem(x,y)\}$ is the maximum remoteness value in the data, used to permit a reversal of the sense of merit for remoteness. The advantage of a single composite measure is that the whole process can be mechanized to the point of relying on $M(x,y)$ analysis to report the desired number of top-ranked candidates by the combined criterion. On the other hand, the scales of the three dimensions above are not the same and vector distance measures of any type tend to be dominated by the dimension with the largest scale. There may be no real need, however, to boil it all down to a single overall measure, as this could lose sight of some individually interesting candidates. The alternative use of the raw metric data can be direct in a scatterplot. Let us say for example only, that a given network had a total of 5,000 new edge candidates. One might then retain the top 500 (x,y) pairs ranked by $y(x,y)$ partial express analysis of demand flows. These 500 (x,y) pairs may then be plotted in a scattergram in the $Freq(x,y)$ versus $Rem(x,y)$ plane. Visual inspection can then easily identify any clusters of "high merit" candidates or candidates that rank highly in one dimension only and are worth including on that basis alone. An arbitrarily shaped dividing boundary can then be drawn as illustrated in [Figure 9-14](#) selecting individual cases of high merit in either individual sense or a combined sense. This approach could also employ formal cluster analysis of the data to select a short list of candidates.

Figure 9-14. Using Frequency - Remoteness data to short-list the edge candidates.



Once the short-list candidates are identified, the procedure to evaluate each in detail is to repeat the optimal design problem on the topology with the new edge. Alternately, with later methods, the short list may be viewed directly as a space within which to solve directly for a multiple-edge topology augmentation. In practice, when all of the short list candidates have been evaluated individually in this way, planners may then also wish to test a few select combined placement redesigns. For instance, the net benefit of placing two spans, each individually of moderate cost-benefit, may be more or less beneficial as a combined plan than the sum of their individual benefits. In practice, however, once our aim for topology planning goes beyond considering a single new edge addition at a time, we are just as well

off to raise our sights and consider the problem of solving directly for an entirely optimal topology in the first place, or (if we can do the latter) solve directly for multi-edge augmentations of an existing topology.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.7 The Complete Mesh Topology, Routing, and Spare Capacity Problem

Let us now look at the complete optimization problem that involves topology in a span-restorable mesh network. Here we go beyond considering single-edge additions to solving directly for the optimal set of edge decisions in conjunction with the usual mesh routing and spare capacity design. Not surprisingly it turns out to be difficult to solve the resulting model directly to optimality above just a few nodes. However, this is a problem for which, if we can solve it completely in the "green fields" sense at small sizes with a universal edge set, then it gives us an ability to apply it to incremental topology planning problems with a reduced set of "high merit" candidate edges to consider.

In the basic model to follow, all $N(N-1)/2$ bidirectional edge candidates are considered. (Equivalently in a directional orientation, if either direction edge is chosen between two nodes, its reverse direction is also asserted.) There are no length or hop limits on the routing of working or restoration paths due to the arc-flow nature of the model. The model is specifically based on span restoration and the set of failure scenarios consists of all single-span failures. As with FCR, we have to abandon the arc-path method altogether in favor of a pure edge flow type of formulation to cope with the topology becoming part of the solution variables. We use the sets, variables, and parameters so defined for FCR (in [Section 9.2.5](#)), to which we add:

- s_{ij}^{kl} is the amount of restoration flow routed over the edge between nodes (k,l) in the direction from k to l for restoration of failed edge (i,j) .
- s_{ij} is the spare capacity assigned to the edge between nodes (i,j) to support the largest combination of simultaneously imposed restoration flow requirements over that edge in the (i,j) direction.

The complete formulation, for mesh topology, routing, and spare capacity (MTRS):

MTRS

Equation 9.16

$$\min \sum_{ij \in A} \{c_{ij} \cdot (w_{ij} + s_{ij}) + F_{ij} \cdot \delta_{ij}\}$$

1.

Source-sink for all demands at their end-nodes and trans-shipment of each demand flow at all other nodes:

Equation 9.17

$$\sum_{nj \in A} w_{nj}^r = d^r \quad \forall r \in D; \quad n = O[r]$$

Equation 9.18

$$\sum w_{jn}^r = d^r \quad \forall r \in D; \quad n = T[r]$$

$$jn \in A$$

Equation 9.19

$$\sum_{in \in A} w_{in}^r - \sum_{nj \in A} w_{nj}^r = 0 \quad \forall r \in D; \quad \forall n \notin \{O[r], T[r]\}$$

2. Working capacity sufficient to support all demand flows:

Equation 9.20

$$w_{ij} = \sum_{r \in D} w_{ij}^r \quad \forall ij \in A$$

3. Source-sink for all restoration flows at their span-failure end-nodes and trans-shipment of restoration flow at all other nodes—all using only spare capacity:

Equation 9.21

$$\sum_{ik \in A; j \neq k} s_{ij}^{ik} = w_{ij} \quad \forall ij \in A$$

Equation 9.22

$$\sum_{kj \in A; i \neq k} s_{ij}^{kj} = w_{ij} \quad \forall ij \in A$$

Equation 9.23

$$\sum_{nk \in A; k \notin \{i, j\}} s_{ij}^{nk} - \sum_{kn \in A; k \notin \{i, j\}} s_{ij}^{kn} = 0 \quad \forall ij \in A; \quad \forall n \notin \{i, j\}$$

4. Spare capacity is sufficient to support largest simultaneous restoration flow imposed on each edge, over all non-simultaneous failure scenarios:

Equation 9.24

$$s_{kl} \geq s_{ij}^{kl}; \quad s_{kl} \geq s_{ji}^{kl} \quad \forall (ij), (kl) \in A^2; \quad (ij) \neq (kl)$$

- 5.

Working or spare capacity can only be had on an edge if that edge is chosen:

Equation 9.25

$$w_{ij} + s_{ij} \leq K \cdot \delta_{ij}; \quad \delta_{ij} \in \{0, 1\}; \quad w_{ij}, s_{ij} \text{ integer}; \quad \forall ij \in A$$

to which we add the following "added valid knowledge" constraints to help in the solution:

- The solution topology must at least be a tree:

Equation 9.26

$$\sum_{ij \in A | i < j} \delta_{ij} \geq N$$

- Every node must individually have degree of two or more:

Equation 9.27

$$\sum_{k \in N; i \neq k} \delta_{ik} \geq 2; \quad \forall i \in N$$

and, optionally:

- Do not consider any graph with unreasonably high overall average degree:

Equation 9.28

$$\sum_{ij \in A | i < j} \delta_{ij} \leq d_{max} \cdot N/2$$

where d_{max} is a "belief constraint" based on empirical recognition of a practical upper limit on the maximum average nodal degree of any plausible solution.

We may also add any already existing edges:

Equation 9.29

$$\delta_{ij} = 1 \quad \forall j \in \hat{E}$$

and bounds on the maximum and minimum number of edges in total:

Equation 9.30

$$S_{max} \geq \sum_{ij \in A} \delta_{ij} \geq S_{min}$$

where S_{min} and S_{max} are inputs based on knowledge about the probable whereabouts of an optimal topology in terms of the minimum biconnected subgraph constraint or tighter knowledge about the minimum degree to consider from pare-down studies and \hat{E} is a set of spans that either already exist or for other reasons are already accepted as part of the solution.

As mentioned, MTRS is cast in an pure flow framework that is a significant departure from the arc-path approach used extensively before

this. When the topology has been defined ahead of time, an arc-path approach is often preferred because it allows explicit control and direct observability of the working and restoration routes employed in the solution. If needed, it also allows a trade-off between solution quality and run times through strategies that control or ration the total number of eligible routes represented for working and restoration flow assignment in such problems. But when the graph topology is itself admitted as a solution variable, the setting up of data files for an arc-path formulation becomes untenable: a master set of eligible routes would have to be developed for representation (in the AMPL DAT file) that is structured in some way so that, for each combination of edges selected, it is evident which routes, among all possible on the full-mesh graph, are "enabled" under the specific set of non-zero edge variables. It is as though every plausible topology would have to be identified ahead of time and a set of eligible working and restoration routes determined and stored for each topology instance. But we will see that simply enumerating the plausible edge-vectors of a solution is itself a big part of the optimization problem. Hence we are forced to switch to a transportation-like representation of working path routing and restoration flow because of its self-contained nature.

There are two places in MTRS where the transportation-like structure is evident. In [Equation 9.17](#) - [Equation 9.19](#) there is a simultaneous multi-commodity transportation-like structure dealing with the normal routing of working flows. For each O-D pair there is a "source node" and corresponding "sink node" constraint followed by assertion of transshipment constraints at nodes that are neither source nor sink for a particular demand. The need to express the concept of transshipment at other nodes (total incoming flow = total outgoing flow for a given commodity) is ultimately why the whole formulation (capacities, flows, and edge selection variables) is forced into a unidirectional framework (which is easily mapped back to the corresponding bidirectional capacity allocations for a fiber optic transport network). [Equation 9.20](#) generates the (directional) working capacity assignments on each edge to simultaneously support the required working flow variables on each edge, for each demand pair.

The second transportation-like structure appears in [Equation 9.21](#) - [Equation 9.23](#). This is a set of *non-simultaneous* single-commodity flow subproblems, each describing the corresponding source, sink and transshipment constraints pertaining to the restoration flows for one particular failure. [Equation 9.24](#) is the corresponding spare capacity generating constraint. As in standalone SCA, it is an inequality because the requirement is to force the spare capacity on each edge to satisfy the largest of the non-simultaneous restoration flows imposed on the given edge. Finally, [Equation 9.25](#) deals with the edge selection variables that define the topology on which the above routing and restoration solutions are jointly coordinated to minimize total cost.

9.7.1 Added Valid Knowledge Constraints

The additional constraints [Equation 9.26](#) through [Equation 9.28](#) are not logically required but can speed up the solution times by expressing topological properties that have to exist in any connected network that satisfies the restorability constraints in [Equation 9.21](#) - [Equation 9.23](#). First, [Equation 9.26](#) is a single global constraint that the topology must contain at least as many edges as there are nodes for the network (otherwise the network is not two-connected). The qualifier " $i < j$ " in the range variable for the summation is just to avoid enumerating identical edges twice. In this model we are forced technically—by the need to express trans-shipment—to employ directed edges, but in our *intent* all edges are actually bidirectional. In other words (i,j) stands for (j,i) too, so we only enumerate over $(i,j) | i < j$ to avoid double-counting edges. Alternately [Equation 9.26](#) would just have $2N$ on the right hand side.

The solution corresponding to equality in constraint [Equation 9.26](#) is a Hamiltonian ring—which, interestingly, *does* emerge in test cases when a Hamiltonian cycles exists and edge costs are much higher than the incremental capacity costs. Secondly, [Equation 9.27](#) says that in addition each node must *individually* be of at least degree two. Corresponding additional constraints can be applied to FCR as well. In correspondence to [Equation 9.26](#), FCR would have:

Equation 9.31

$$\sum_{ij \in A | i < j} \delta_{ij} \geq N - 1.$$

The corresponding individual node constraint in FCR is that every node has at least one edge incident on it for FCR, i.e.,

Equation 9.32

$$\sum \delta_{it} \geq 1; \quad \forall i \in N.$$



Whereas [Equation 9.26](#) and [Equation 9.27](#) may or may not be applied, they are certainly true mathematical properties of any feasible solution. On the other hand, [Equation 9.28](#) is "belief-based" constraint representing *a priori* knowledge that (for instance) no known transport network has an average nodal degree higher than five. In other words, if we put credence in the merit of real transport graphs for their intended purposes, we can produce a guideline on the maximum number of edges an optimal design could plausibly contain. To the best of our knowledge in practice, all real networks lie in the range $2 < d < d_{max}$ with $d_{max} < 5$. Thus, with current and foreseeable technologies and cost structures, it is practically reasonable to adopt some d_{max} for the graph as a whole, or for individual nodes as well. Of course, in a purely general instance of MTRS as a mathematical problem only, it would not be known *a priori* what d_{max} value brackets the optimum and this would not be advisable. But in problems where the costs of edges and capacities are derived from real circumstances, it is quite reasonable and useful and to apply something like $d_{max} < 6$ (or certainly $d_{max} < 8$) to restrict the solution space without affecting optimality. The philosophy is that $d_{max} < 6$ is still quite a loose and conservative upper bound, unlikely to ever impinge on our solution quality with any typical edge and capacity costs, but it is still quite worth stating because a vast number of solutions having $d_{max} < d < N-1$ are ruled out by this one constraint.

9.7.2 Relaxations

Consistent with Gendron's experience with FCR, the 1/0 edge selection variables in MTRS are fundamental to structuring the mutual capacity and edge-cost sharing issues in any feasible, realizable design, so we do not relax them (except in later lower bounding trials). We also keep the working and spare *capacity* variables integral (but non-modular) and relax the underlying working and restoration *flow* variables. A useful property inherited from SCA is that if integrality is asserted on the s_j and w_j capacities, the restoration flow variables f_i^p may be relaxed without affecting solution quality or feasibility. In this case each restoration flow subproblem for an individual failure scenario is a single-commodity integer-capacitated network flow problem for which flows remain integral if demands and capacity are integral. This was pointed out by [\[Wang98\]](#) with reference to the basic properties of minimum cost network flows.

On the other hand, the relaxation of working flows is justified as an acceptable practical measure when attempting direct solution of the full MTRS problem. Fractional working flows may arise in the solutions but our experience, and work by Kennington [\[KeLe98\]](#), indicates that a simple "repair procedure" can reintegrate fractional working flows at minimal or no impact on the objective function cost. Picavet and Demeester [\[PiDe00\]](#) also comment on the gap due to working flow relaxations being only ~1% in their experience with the same issue. In addition, in the subproblems we later define, the relaxation of working flows is acceptable because those subproblems are solved only for the purpose of *nominating* edge candidates for a final design step which itself can then be solved without working flow relaxation if desired.

9.7.3 Complexity of MTRS

To assess the number of variables and constraints in a direct solution of MTRS (assuming the universal set of edge candidates), let us define $Y = N(N-1)/2$ to represent the number of all possible (bidirectional) edges in an instance of MTRS. Then we have Y edge selection variables, Y working capacity variables, Y spare capacity variables, $Y \cdot (Y-1)$ restoration flow variables, and (assuming all O-D pairs may exchange demands) another Y working flow variables. The total is $2(Y + Y^2)$ or $2 \cdot (N^4 - 2 \cdot N^3 + 2 \cdot N^2 - N)$ variables. Y of these are 1/0 variables and the rest are strictly integer. Allowing that all nodes may exchange demands, [Equation 9.17 - Equation 9.19](#) generate $2Y + Y(N-2)$ constraints. [Equation 9.20](#) adds Y . [Equation 9.21-Equation 9.23](#) add $2Y + Y(N-2)$. [Equation 9.24](#) adds $Y(Y-1)$ and [Equation 9.25](#) adds Y constraints. The total number of constraints in an N node MTRS problem is therefore $(N^4 - N)$. A 50-node problem therefore has over 3 million variables in ~ 6.25 million constraints.

From another standpoint we can view the complexity of the edge-decision space alone. Consider that a ten node problem involves $Y/2 = 10(9)/2 = 45$ (bidirectional) 1/0 edge decision variables implying a $2^{45} = 3.5 \times 10^{13}$ distinct combinations of edge decision variables. Obviously not all, in fact very few, of these edge vectors might describe feasible graphs for an MTRS solution, but assuming that a solver

could evaluate 1000 edge vectors per second, either rejecting it for lack of connectivity or for having stub nodes, or—if it is a qualified topology—going on to evaluate its capacity cost for routing and protection, then it would still take *over 1,100 years to consider all solutions to a 10-node problem!*

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.8 Sample Results: Studies with MTRS

MTRS can, nonetheless, be implemented and solved to optimality for suitably small problems. Although too complex for direct general use in planning tools, MTRS provides us with a research tool which serves as kind of microscope with which to view the structure and properties of strictly perfect topology solutions. This leads to some interesting results and understanding about how topology responds to the relative costs of capacity, edges and the demand matrix. It also leads to highly effective heuristic solution techniques that are usable in practical network planning tools.

9.8.1 Effect of Edge-to-Capacity Cost Ratio, Ω

[Figure 9-15](#) shows how the total cost of optimal 8-node MTRS solutions, with its universal set of 28 possible edges (and hence 2^{28} possible topologies) responds as the cost of edge establishment increases relative to the cost of a unit capacity on the same edge, Ω . In these results, edge and capacity costs are proportional to the Euclidean distance between the 8 nodes as positioned in the plane for the test case, and shown in [Figure 9-16](#). For each edge, the edge establishment cost is Ω times the cost of a unit capacity, per unit length. The relative positions of the 8 nodes and the demand between them are based on the COST-239 network and data shown ahead in [Figure 10-13](#). The test case here is essentially COST-239 without Paris, London, and Copenhagen and where each 2.5 Gb/s of traffic is converted into one demand unit, e.g., a lightpath requirement. None of the spans shown in [Figure 10-13](#) are assumed to exist here, however, and we consider the universal set of candidates on 8 nodes.

Figure 9-15. Total cost of optimal 8-node MTRS solutions as Ω is increased.

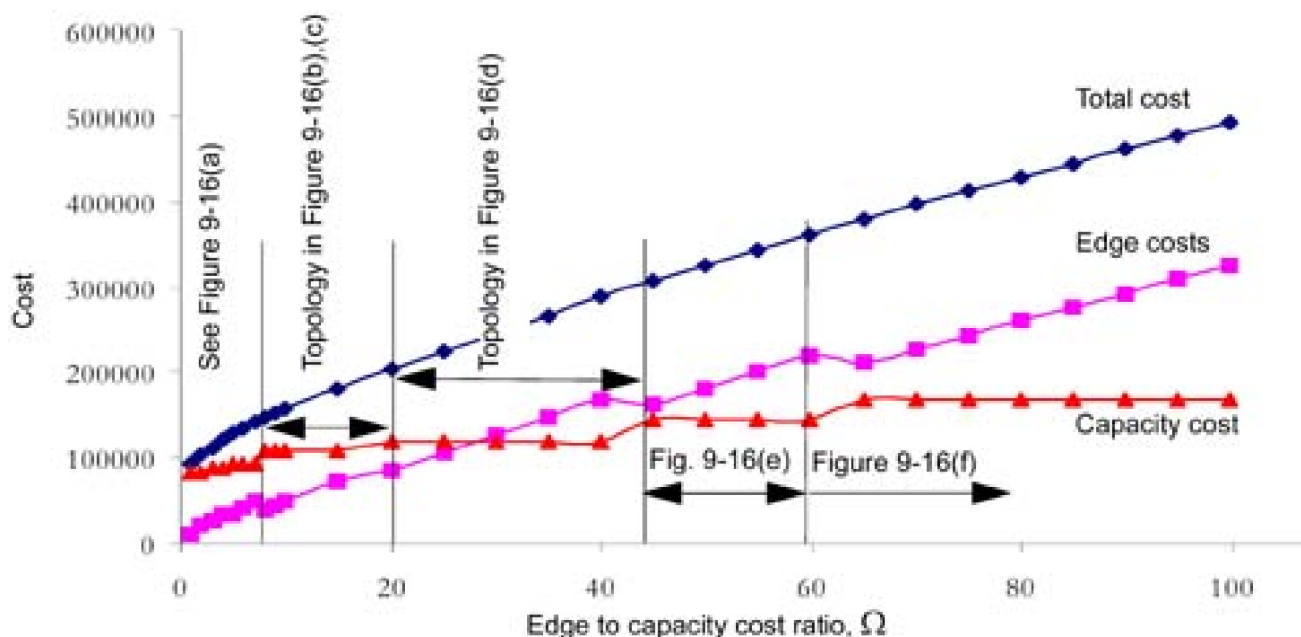


Figure 9-16. Evolution of the Optimal Topology from MTRS as Ω is increased [Ezem03].

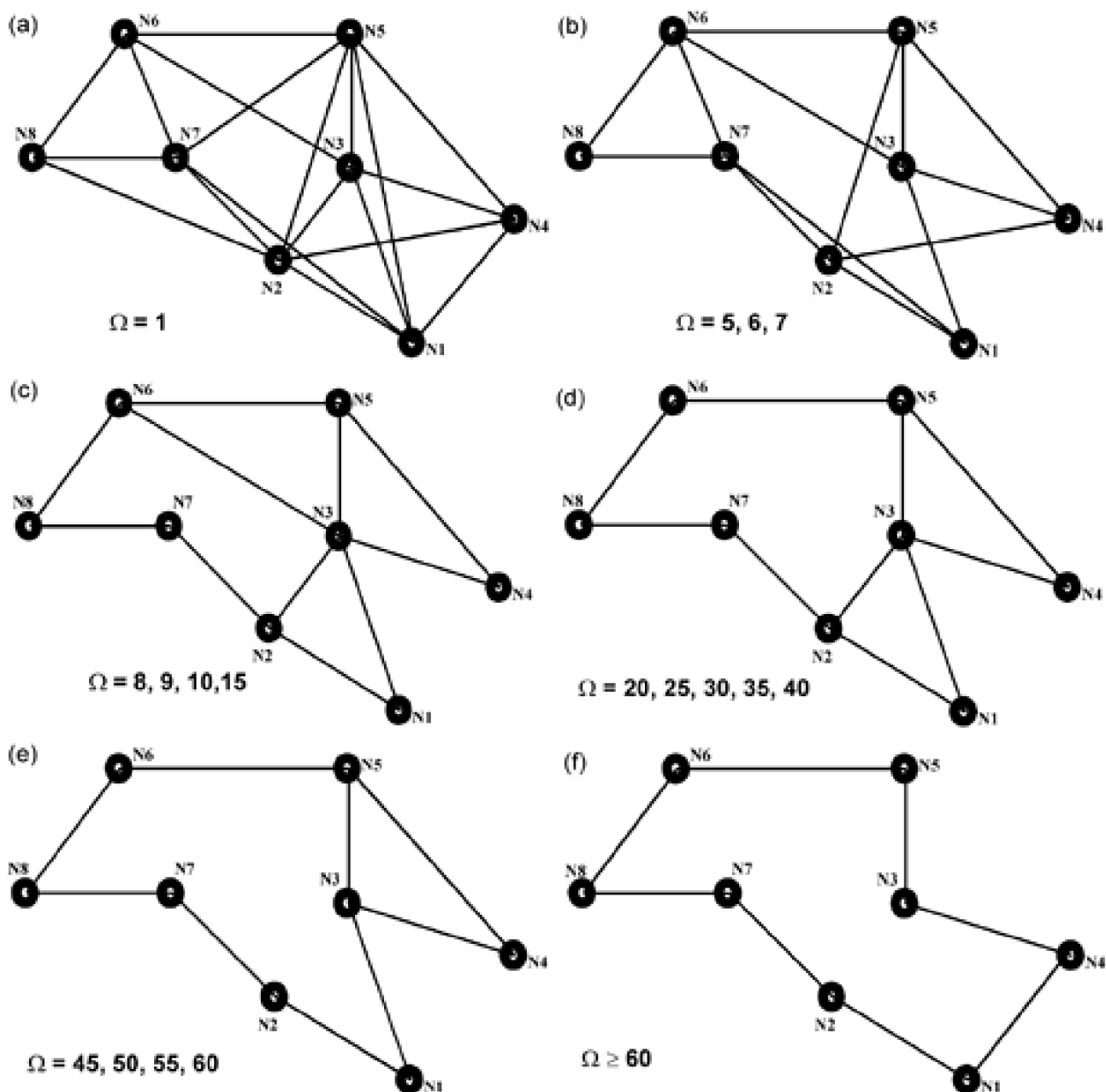


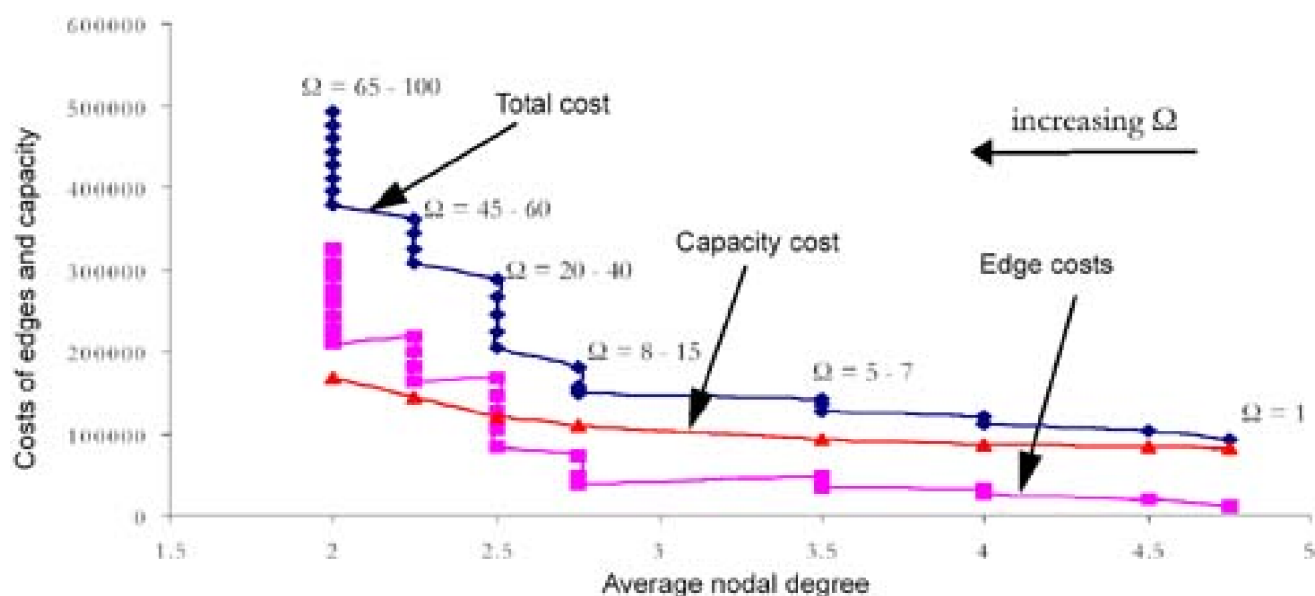
Figure 9-15 shows that total cost rises relatively uniformly as Ω is increased, but there are significant plateaus and jumps in the underlying breakdown of total cost between edges and capacity. These dynamics correspond to sudden jumps in the optimal graph topologies, samples of which are shown in Figure 9-16 for the ranges of Ω indicated on Figure 9-15.

As expected, MTRS always produces a closed graph. Note also in Figure 9-16 how the topology evolves from richly connected down to a Hamiltonian cycle as the cost of edges becomes increasingly important. Once the Hamiltonian cycle of Figure 9-16(f) is reached, total cost rises completely linearly with Ω because no choice of fewer edges is feasible from a survivability standpoint. If this was an FCR solution pushed to this extreme, the solution would be a minimum cost spanning tree of $N-1$ edges, rather than a cycle of N edges. More generally note that for each Ω or certain ranges of Ω , there is a specific optimal topology. This is the topology that balances efficient working routing and spare capacity assignment costs against the cost to establish itself. In this regard the optimal graph of $\Omega = 20$ to 40 in Figure 9-16(d) seems of special interest because it is the graph on which roughly equal edge and capacity costs arise (from inspection of figure 9-15) and is a graph with $\bar{d} = 2.5$ which is characteristic of North American long-haul networks.

Figure 9-17 accompanies these results showing how nodal the degree of the optimal solution responds to increasing Ω . This plot has axes of cost versus nodal degree but it does not have the general shape of Figure 9-6 as may be expected because, here, Ω is the independent variable and we are plotting the properties (cost and degree) of the optimal designs with Ω as a parameter. In a plot such

as [Figure 9-6](#), Ω is constant and nodal degree is varied in search of a single optimal solution. [Figure 9-17](#) clearly shows how capacity costs vary almost continuously but are fixed at each "stack" of points in the edge costs. Each of the stacks represents an unchanging topology whose set of edges has increased cost as Ω is increased, but does not change until suddenly some new configuration of fewer edges costs less than the associated increase in routing (and hence capacity) costs, and then a sudden jump to the new optimum topology takes place.

Figure 9-17. Total cost and nodal degree of optimal eight-node MTRS solutions as Ω increases.



The form of these results, undertaken for research, differ in two important ways from the way we encounter the topology problem in practice. One is that we are not often able to solve MTRS directly to reveal the optimal topology. MTRS run times with the universal edge set go directly from hours to solve the eight-node problems above, to weeks for nine-node problems. Also the real problem context is one where Ω is given by the costs of equipment and rights-of-way and is not under our direct control. In fact Ω may usually be different for each possible edge. Nonetheless, the eight-node results, which we know are perfect solutions let us appreciate what happens in the actual problem we face: if the topology is too rich we have excess cost from edges—and if it is too sparse then capacity costs are excessive. At the extremes where general Ω is very low, capacity costs dominate total cost and we can be guided in a topology choice by principles such as efficient working path routing. At the opposite extreme what would matter most to the solution is a minimum cost biconnected graph, ultimately a particular cycle at the highest of Ω values. It is in the midrange in fact that the problem is the most general and hardest to solve. CPLEX solution times exhibit this behavior quite strongly: at high or low Ω , the MTRS problem solves most quickly. In the results of [Figure 9-16](#), the highest run times were seen exactly in the region of $\Omega \sim 25$ where edge and capacity costs are in balance.

9.8.2 Effect of Demand Intensity and Demand Pattern

The comparative results so far presented on effects of Ω and nodal degree employ a common demand pattern across alternatives. So how does the optimal topology vary with the pattern and relative intensity of the demand? To discuss this, let us separate the notion of demand intensity from demand pattern.

By intensity we mean any bulk common scaling on all demands of one case relative to another. Let the multiplicative factor involved be called b . In considering the effect of any such bulk scaling of demand, intuition and the total cost function for MTRS ([Equation 9.16](#)), suggests that any increase in b is equivalent to a corresponding divisor on Ω . The intuitive reason is that if we just scale up all demand quantities (under the same routing rules for each O-D pair), then this implies b times more capacity use on each edge. Equivalently the effect from a topology standpoint is also indistinguishable from an increase in the capacity cost coefficients c_{ij} in [Equation 9.16](#) by a factor b .

at the same demand intensity. In summary, what we are saying is that the total cost function for edge (i,j) under variable intensity or edge to capacity ratio, based on [Equation 9.16](#) can be rewritten as:

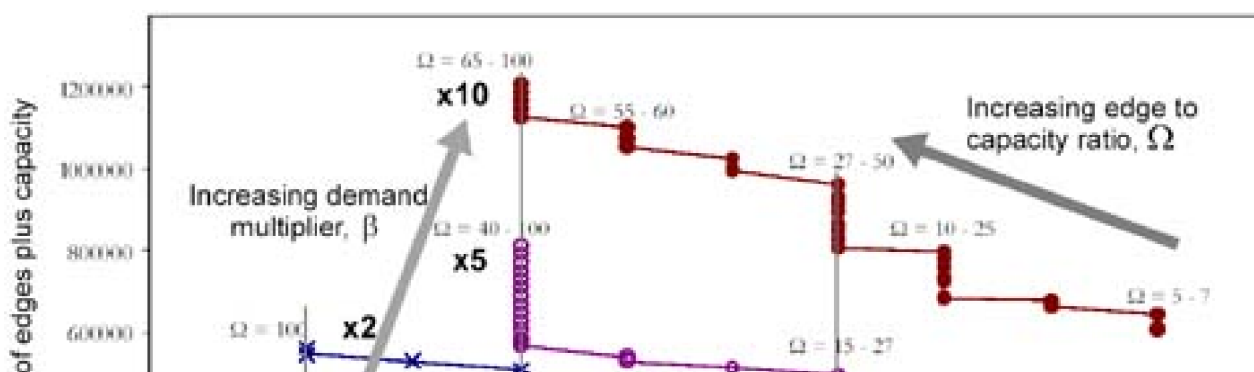
Equation 9.33

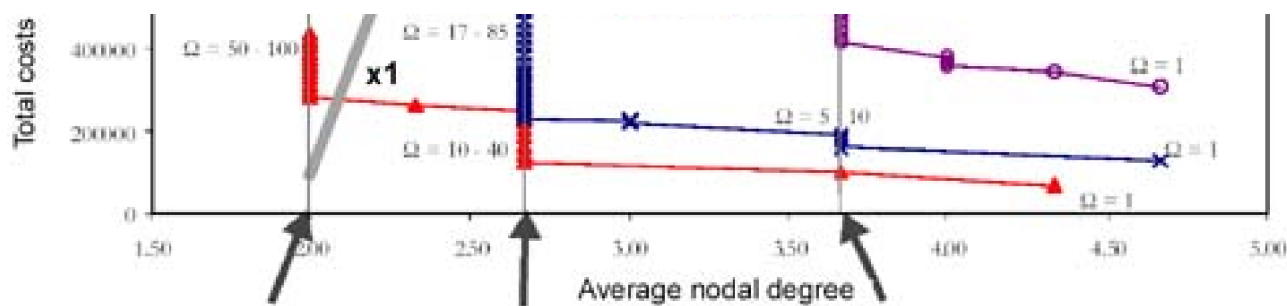
$$c_{ij} \cdot (w_{ij} + s_{ij}) + F_{ij} \cdot \delta_{ij} = \beta \cdot (w_{ij} + s_{ij})^0 + \Omega \cdot \delta_{ij}$$

where F_{ij}/c_{ij} is simply rewritten as Ω and $(w_{ij}+s_{ij})^0$ is the total capacity in edge (i,j) under some baseline demand matrix D^0 pattern relative to which the current demand is $b \cdot D^0$. The summation of total costs on all spans in [Equation 9.16](#) does not alter the logic. But under minimization no constant factor makes a difference to the solution details (only its absolute cost), so that relative to an initial problem based on D^0 , any other problem with the same Ω/b ratio has the same optimal graph topology. In other words, scaling demand intensity is equivalent to lowering the edge-to-capacity cost ratio in terms of the effect on optimal topology. This was also verified experimentally in [Ezem03](#). The only situation where this would not follow is where the working capacity routing and/or SCA process for spare capacity placement were not themselves invariant under b . Modularity and/or existing edges with fixed amounts of already installed capacity could both have this effect.

[Figure 9-18](#) presents a series of experimental design results with optimal solutions to MTRS using test cases created on 6 nodes from the original COST-239 topology and demand pattern. The plot is similar in nature to that of [Figure 9-17](#) but shows only total cost and includes families of curves where demand undergoes flat multiplication on all O-D pairs by the factors "x1" through to "x10" and at each demand intensity, Ω is varied to generate a family of solution topologies for each Ω, b combination. From [Figure 9-17](#) we understand that the stacks of rising cost at fixed degree are the signature of a graph topology that is robust under increasing Ω . Results confirm the reasoning above in that certain topologies reappear under equivalent Ω/b ratios. For example, the graph for $b=1$, has the same graph for $10 < \Omega < 40$ as appears for $b=5$ for $40 < \Omega < 100$. What the reasoning under [Equation 9.33](#) doesn't tell us, but [Figure 9-18](#) portrays, is just how often a single graph turns up as being optimal for a wide range of conditions. For instance [Figure 9-18](#) shows that there are three graphs (associated with the marker arrows) that emerge quite often. One, at $d=3.7$ serves well for a range of demand and edge cost factors centred roughly around $\Omega/b \sim 5$. The next, quite dominant, graph at $d \sim 2.6$ is well suited for a wide range of conditions centered on $\Omega/b \sim 10$. The most sparse graph of all $d \sim 2.01$ serves in the $\Omega/b \sim 50$ range. We also note from [Figure 9-18](#) that although Ω and b are interchangeable under [Equation 9.33](#), the "basins" of Ω, b combinations that map into a common best graph topology become smaller at high demand. At the lowest demand curve in [Figure 9-18](#) we see only five different topologies to cover the whole range of $1 < \Omega < 100$, and only two of these cover most of the Ω range. On the other hand at the highest demand (x10) seven discrete graphs are needed to cover the range $1 < \Omega < 100$, and it becomes much more important in total cost terms to be at the right topology for each smaller basin of Ω, b combinations before the jump to the next discrete topology.

Figure 9-18. Complete relationship between demand and edge-to-capacity cost ratio (Ω) for C239-6n in determining optimal topology, indicating three very dominant topology options.



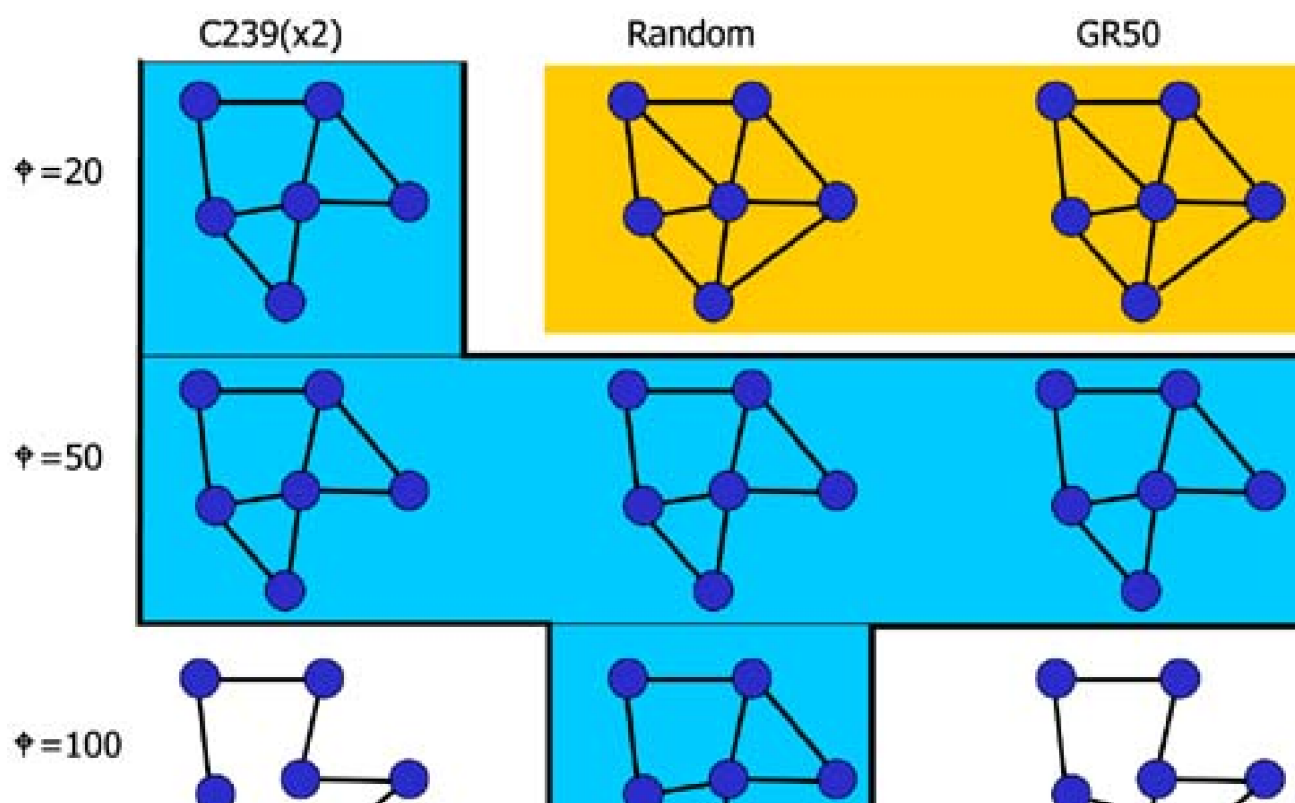


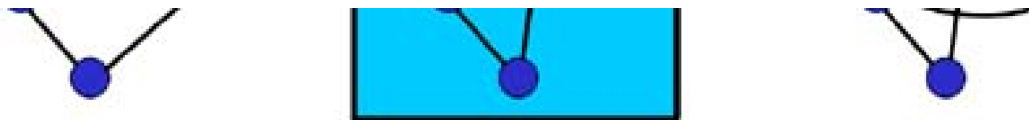
Much less can be said in general about how the relative *pattern* of demand between nodes affects the optimal topology. By and large this is where we must resort to actual solution attempts using methods that are the main topic of this chapter. Nonetheless one general insight is that significant variations in the pattern of demand can be "absorbed" without triggering or requiring a shift in the topology to remain optimal. Each variation in pattern affects $w_{ij} + s_{ij}$ costs in detail, but does not often trigger a shift in topology. This tendency to a stable (or "pattern-robust" topology) seems to be strongest when both capacity and edge costs are significant in the solution. In these cases a moderately rich topology tends to be selected, within which the routing and SCA subproblems have considerable latitude to absorb detailed differences in pattern without requiring an edge addition or deletion to remain optimal. Conversely, we observe that for a given total amount of demand, the optimum topology is more sensitive to the exact demand pattern at high Ω (unless Ω is so high that a limiting cycle has been adopted).

These effects can be seen in some tests of different demand patterns conducted on a further subset of 6 of the 8 nodes used above.

Three different demand patterns, denoted C239(x2), Random, and GR50 were tested at three values of Ω . The resulting optimal topologies are portrayed in Figure 9-19. The C239(x2) demand matrix uses the original published demand for the 6 nodes selected, scaled by two to bring its range (largest to smallest demand difference) to match the other two patterns at 20. In "Random" each O-D pair is assigned a demand quantity from a uniform random distribution of integers in the range [5,...,30] regardless of distance or nodal degree. The total demand in Random is 2.2 times that in C239(x2). "GR50" is an inverse distance-weighted gravity model with demands generated with a mutual attraction product term weighted by the inverse of the distance between the nodes. The importance factors for the mutual attraction effect were taken as the degree of the corresponding nodes in the actual COST-239 network. GR50 has 1.2 times the total demand of C239(x2). The important thing is that differences between solution topologies were analyzed with detailed reference to the actual demand matrices, with the following main findings.

Figure 9-19. How demand pattern alters the topology (6 node optimal solutions of MTRS).





The moderately connected 8-edge graph (given the dark shaded background in [Figure 9-19](#)) emerges as the optimal topology in five of the nine trials. Even though demand pattern and working and spare capacity differ in each design, this one graph is optimal for all patterns at the mid-range $\Omega=50$ and for C239(x2) at lower Ω as well as Random at the higher Ω . In addition at $\Omega=20$, both other patterns contain this hub and spoke structure but add the same two edges to it in a way that inspection shows correlated with greater direct demand between the extra node pairs connected in those cases. What seems to be so robust about this graph is that it uses only two relatively short edges, above the minimum required for biconnectivity and it provides for shortest routes through the center of the network which are nearly as low as the Euclidean distance between those node pairs. Note in particular that the evidently central node in this topology is "star" connected to four of the other five nodes. Thus, out of 15 O-D pairs in total on 6 nodes the central node and its four immediately attached neighbors ($2/3^{rds}$ of all the demand pairs) see relatively direct routing for their working demands. The general points we draw from this are that to find an optimum topology, and moreover a topology that is preferably stable over a wide range of pattern difference:

1. Efficiency for the routing of working flows remains of first order important. Although by itself working flow routing efficiency would never result in a biconnected graph, elements of the topology are clearly justified by their efficacy with respect to working paths.
2. While we may still be justified in asserting *a priori* limits on the overall network average degree, we need to permit the formation of higher than average degree "hubs" which can play a significant role in stabilizing the optimum topology under pattern variations.
3. Optimum topology is more sensitive to demand pattern at high Ω , where fewer edges are being chosen and edges dominate the total cost.
4. The more specific topology changes seen at $\Omega=100$ are again primarily attributable to differences in the working demand pattern, not primarily due to shifts in the strategy for placing spare capacity to protect such flows.

The main conclusion seems to be that working considerations are of more direct importance to determining the MTRS topology. As long as the topology is ultimately biconnected, protection arrangements seem to be of secondary dependence on the exact topology. Because spare capacity is shared, and because *any* set of restoration routes under the hop limit for each failure scenario suffices, achievement of an efficient spare capacity design is generally much *less* dependent on the topology. In contrast each working route consumes dedicated capacity and shortest routes *are* far preferred over indirect routes for working capacity.

The insights obtained about MTRS from these experiments inspire a simple hypothesis: that optimal graphs for MTRS may be approximated by first finding a good graph for the working-only problem and then just making that graph biconnected in an efficient way. This leads to a 3-part heuristic based on this approach to approximating MTRS.

9.9 A Three-Part Heuristic for MTRS

In this section, we describe a heuristic for MTRS based on the hypothesis just stated and a decomposition into three smaller MIP problems.^[5] The 3-step method is based on viewing MTRS as containing FCR-like aspects for working flows as well as mesh SCA- or JCA -like aspects (for restoration). The basic idea is to first identify a topology that is good from an FCR standpoint, and another set of edges that are meritorious from a mesh spare capacity viewpoint, and then solve the full MTRS model within the union of those specific edge sets, rather than the universal edge set. The three steps, each to be described further, are:

^[5]Material in this section is adapted from prior publication on this heuristic with J.Doucette^[GrDo01].

W1

Solve a (working-only) fixed charge plus routing problem (FCR). Edges identified at this stage are collectively sufficient for routing and our philosophy is that they are of special merit for consideration in a complete design by virtue of their special role in serving working demand flows. Any preexisting edges are represented as edges that have zero fixed charge for their establishment.

S2

Solve an artificial problem for the minimum cost of additional edges and capacity to ensure restorability of the working flows in W1. We call this the reserve network fixed charge plus spare capacity problem (RN-FCS). Additional edges identified by this step are sufficient to enable restoration by closing the graph and capacitating the needed spare capacity at minimum incremental cost. The philosophy is that such edges are therefore of merit for further consideration in a complete design by virtue of their special efficiency from a restoration standpoint. Any preexisting edges and new edges from W1 are asserted as inputs and the objective function excludes any fixed costs for the latter edges.

J3

Solve a *restricted instance* of MTRS where the set of candidate edges is the union set from W1 and S2, not the universal set of all possible edges. The idea is that since MTRS is exponential in $|A|$ there is high run-time leverage on reducing the number of candidate edges. But solution quality may not suffer greatly if the reduced edge set consists only of edges that are of special merit either from the standpoint of routing (W1) or restoration (S2). The final restricted MTRS problem instance selects a set of edges that are collectively sufficient and efficient for *both* routing and restoration.

The central hypothesis is that *within the union of the edge sets arising from W1 and S2 lies a subgraph on which a high quality approximation to MTRS can be found*. The labels W1, S2, J3 are meant to suggest: "step 1 for Working only, step 2 for Sparing only, step 3 for Joint reduced problem. It is important in this regard to realize that J3 and MTRS (or "full MTRS") have exactly the same problem structure. The only difference is that a "J3 instance" of MTRS has a restricted edge set.

The computational advantage of this approach is significant because the most onerous component step (W1) can be a partially relaxed

and/or time-limited FCR problem instance. FCR is itself still a difficult problem, but its role in the heuristic is only to help promote a candidate edge set that is of interest from a working standpoint, so we do not necessarily need to solve it to optimality. In comparison, S2 generally solves quickly and J3 is much faster than unrestricted MTRS because we typically have reduced the candidate edge space from $Y=N(N-1)/2$ to $\sim 2N$ or $3N$ (edge sets corresponding to networks of maximum degree of six, say). Although the result is approximate in the sense of global optimality, J3 is itself an exact instance of MTRS so its output is a fully feasible and constructible design in terms of all edges, routing and restoration details. In other words, there are no functional details that are approximate as a result this heuristic. The next sections define and discuss each step in more detail.

9.9.1 Step W1: Working-only Fixed Charge Plus Routing

W1 is an instance of FCR without regard to any survivability considerations, within the universe of *all* possible edges for the problem. As such, the formulation for this stage is unchanged from that for classical FCR in [Section 9.2](#). What we want going forward from this step is, however, only the topology selection outcome and the objective function value for later bounding use in J3. There is nothing in the FCR formulation that assures that a restorable topology emerges. In fact trees are very likely at this stage. Since the detailed routing associated with the FCR solution is not retained, the working flow variables are candidates for relaxation to speed up this step. The idea is only to produce a first topology that by itself is nearly optimal if the goal was only to serve the working demand flows. Nonetheless W1 remains the most complex stage of the 3-step method. This step may therefore also be time limited. An option discussed later for W1 is to use an artificially low Ω to identify not just the edges that would strictly be part of the FCR solution at the true Ω , but to also reveal edges that may have been close to this qualification.

9.9.2 Step S2: Reserve Network Fixed Charge and Sparring (RN-FCS)

This step augments the topology from W1 to become at least two-connected while simultaneously minimizing the fixed costs for additional edges and the spare capacity placed on all edges to achieve restorability. The result from W1 is an initial topology and set of working capacity w_{ij} values that fully serve the demand matrix. In S2 the topology from W1 is asserted as "already existing" edges. In S2 only edges from W1 are considered as failure scenarios whereas from a restoration flow standpoint all existing or possible edges are considered. Restoration flows are subject to the same fixed charges that apply in W1 for any edge that is added at this stage. Thus, new edges are added to the topology at this stage if they are justified on their combined merits of closing the graph and providing restoration capacity.

This step benefits computationally relative to MTRS in three ways: (1) The edge decision space is already reduced from Y possible edges to no more than $(Y-N+1)$ remaining edge choices (because at least $N-1$ edges were decided in W1). (2) All working capacity and working flow variables and constraints are eliminated. (3) Not all Y (i.e., $O(N^2)$) possible span failure scenarios have to be considered, only $\sim O(N)$ corresponding to the edges in the FCR solution from W1.

RN-FCS:

Equation 9.34

$$\min \sum_{ij \in \{A-E1\}} \{c_{ij} \cdot s_{ij} + F_{ij} \cdot \delta_{ij}\} + \sum_{ij \in E1} c_{ij} \cdot s_{ij}$$

Equation 9.35

$$\sum s_{ij}^{ik} = w_{ij} \quad \forall ij \in E1$$

$$ik \in A | j \neq k$$

Equation 9.36

$$\sum_{kj \in A | i \neq k} s_{ij}^{kj} = w_{ij} \quad \forall ij \in E1$$

Equation 9.37

$$\sum_{nk \in A | k \notin \{i,j\}} s_{ij}^{nk} - \sum_{kn \in A | k \notin \{i,j\}} s_{ij}^{kn} = 0 \quad \forall ij \in E1; \quad \forall n \notin \{i,j\}$$

Equation 9.38

$$s_{ij} \geq s_{ij}^{kl}; \quad s_{ji} \geq s_{ji}^{kl} \quad \forall (ij), (kl) \in A | (ij) \neq (kl)$$

Equation 9.39

$$s_{ij} \leq K \cdot \delta_{ij}; \quad \delta_{ij} \in \{0, 1\}; \quad \forall ij \in \{A - E1\}$$

Equation 9.40

$$\delta_{ij} = 1 \quad \forall ij \in E1$$

Equation 9.41

$$\sum_{ij \in A | i < j} \delta_{ij} \geq N$$

Equation 9.42

$$\sum_{ij \in A} \delta_{ij} = N$$

$$\sum_{k \in N: i \neq k} w_{ik} \leq c_i \quad \forall i \in N$$

For clarity the objective function is expressed in two parts. First is the edge and spare capacity costs for *additional* edges at this stage. The second sum recognizes spare capacity that may be added at this stage to edges already selected in *W1*. The set of edge selections already made (and "paid for") in *W1* are passed into *S2* in the set *E1* where they are directly asserted as part of the *S2* solution (in [Equation 9.40](#)). In [Equation 9.39](#) the edge space is correspondingly reduced to $\{A-E1\}$. Constraints [Equation 9.35](#) - [Equation 9.36](#) relate

the restoration flow variables s_{ij}^{kl} for each (i,j) failure scenario to the amount of working flow to be protected in the form of source-sink and transshipment constraints for each failure scenario. [Equation 9.38](#) dimensions the spare capacity variable on each new and existing edge. The remaining constraints define the edge selection variables and restate the added valid knowledge constraints.

9.9.3 Step J3: Restricted MTRS for Final Topology Selection

The last step addresses the global coordination of working, spare, and topology considerations that are inherent in the full MTRS problem but not present in the design at the end of *S2*. The augmented edge set at the end of *S2* can only be retained or reduced by this step. Note that *J3* does not adopt the union of edges from *W1* and *S2* as a given topology on which to solve routing and spare capacity. Rather the union of the prior edge sets is again viewed as a candidate edge space. Thus *J3* can either keep all edges promoted to it from prior steps, or further reduce the final set of edges employed. *J3* can also make use of a bound obtainable from the result of *W1*, described below. In addition, the *J3* objective value can be fed into an unrestricted instance of MTRS as an upper bound when attempting to solve a fully optimal MTRS problem. In summary, the three steps play the following roles:

- **W1** Finds a minimal topology and capacity as *justified by working flows* alone.
- **S2** Finds a min-cost topology augmentation as *justified by restorability* considerations.
- **J3** Revises the working flows of *W1* to exploit the augmented topology of *S2* and coordinates the assignment of spare capacity and final selection of edges.

9.9.4 Useful Bounds from Steps W1 and J3

At two stages in the heuristic we can also obtain bounds to aid in solution of either restricted or unrestricted MTRS problem. Specifically, if step *W1* is individually solved to optimality it follows that the objective value of *W1* is a *lower* bound on *J3*. Any feasible solution for MTRS, or a *J3* instance, has to cost more than an optimal solution for FCR alone. (It can only cost more to add protection.) Even more specifically, we can lower-bound a subset of the variables in the MTRS objective function by applying the same line of reasoning to the topology plus working capacity variables *only* within an MTRS problem. That is to say that:

Equation 9.43

$$\sum_{ij \in A} \{c_{ij} \cdot w_{ij} + F_{ij} \cdot \delta_{ij}\} \geq \text{obj. (step W1.)}$$

because the component costs for the fixed charge and routing solution (alone) that are embodied within a full MTRS solution can only

make compromises to accommodate the wider set of considerations in MTRS compared to the pure FCR solution from W1.

Similarly, J3 provides an *upper* bound on the cost of the full MTRS problem because it has considered only a reduced edge set. A full MTRS problem, by considering all candidate edges can only possibly improve on a J3 solution. Thus we can write the additional bounding relationship:

Equation 9.44

$$\sum_{ij \in A} \{c_{ij} \cdot (w_{ij} + s_{ij}) + F_{ij} \cdot \delta_{ij}\} \leq \text{obj.}(\text{step J3.}) .$$

We make use of the first relationship to help solve the J3 problem in the 3-step heuristic and the second one to help solve instances of unrestricted MTRS.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.10 Studies with the Three-Part Heuristic for MTRS

We now look at selected results from the study in [\[GrDo01\]](#) using the 3-step heuristic. This involved notionally small problem instances based on 9, 10, and 15 nodes to enhance the likelihood of obtaining full CPLEX terminations to provide optimal reference solutions. What really characterizes the size of an MTRS problem is, however, not the number of nodes but the number of edge decision variables. Thus, even a 10-node problem with its universal set of 45 edge candidates can be much tougher to solve than a 15-node problem with half as many candidate edges.

9.10.1 Method

All cases at nine and ten nodes used a set of nodes placed at random (x,y) coordinates in the plane. The 15-node cases are based on a previously published network [\[Bell93\]](#) and a limited set of candidate edges. The Euclidean distance between nodes was used as the length, l_{ij} , of the candidate edge between those nodes. The fixed charge for establishment of that edge in the topology would then be b_{ij} and the cost per unit of capacity added to an edge was l_{ij} . In the nine and ten node problems we allow all possible edges and there is non-zero demand value for every O-D pair. The 15-node problem uses 56 candidate edges, 28 of which are the actual edges published in [\[Bell93\]](#) to which we added an equal number of randomly chosen but plausible additional edge choices. By plausible we mean that the expanded set of candidate edges is predominated by planar edges to next-neighbors and neighbors that were not often more than twice the average internodal Euclidean distance away in the plane. The actual set of candidate edges in each case is shown by light lines in the background of the figures that follow. The resultant test problems either have a universal edge set (a full mesh of potential edges) or a set of candidate edges that is typically at least four times the characteristic degree of real transport networks. In the 15-node cases half of the possible demand pairs were also set to zero.

All steps of the heuristic and the MTRS master problem were prepared in AMPL and solved with the CPLEX 6 MIP solver on a four x 250 MHz Sun Enterprise processor with 892 MB of RAM. A CPLEX priority file was used directing it to first branch on topology variables, secondly on the integer capacity variables. For each test case the sequence of runs was to do steps W1, S2, J3 and then with the benefit of the objective values from W1 and J3 as bounds, the full MTRS problem would be attempted to obtain an optimal reference case. Consequently in cases where the full MTRS attempt runs to the time limit without finding any feasible solution at all, it means that CPLEX could not find a solution that *improved upon* the heuristic within the allowed time. Except where indicated (in two cases at 15-nodes), each step of the heuristic was individually solved to optimality. In the cases to be mentioned the W1 (FCR) subproblem was deliberately limited to 15 minutes.

9.10.2 Results

[Table 9-2](#) and [Table 9-3](#) summarize results on the 9-, 10-, and 15-node tests. The test case name, (for example 9n36s4 - 15) indicates the number of nodes (e.g., 9), the cardinality of the universe of candidate (bidirectional) edges (e.g., 36 is a full set of candidate edges for 9 nodes), the random instance number (where there are different random arrangements of the same number of nodes, e.g., 4) and Ω , the ratio of edge-selection cost to unit-capacity cost on an edge (e.g., 15). Where more than one instance of the same size problem is indicated, the location of the nodes and the vector of O-D pair demand magnitudes are re-randomized. [Table 9-2](#) first gives the test case details, then the objective function values and run times for each step of the heuristic, followed by details of the full MTRS reference solution attempt. In the S2 objective values, only the variable cost components of the S2 formulation are recorded (i.e., the cost of edges and working capacity from W1 are not repeated in the S2 objective values). The total cost of the network at S2 is the sum of the edges selected and working capacity costs from W1 plus the objective function for S2 (added edges for graph closure and spare capacity cost).

[Table 9-2](#) also records three types of MTRS reference solution results based on the type of CPLEX termination obtained. "IF" means that

in the time given, the solver was not yet able to find a feasible solution. "FT" results were solved to optimality (a full termination) by CPLEX, in the times shown. "TL" results are cases where the CPLEX did produce a feasible solution but the run was time-limited. The time-limited objective values may be lower or higher than the heuristic. In the latter cases we report the *gap of the heuristic against the time-limited CPLEX performance*. For instance, "within X% of the result from 6 hours of CPLEX time." In these cases we do not use the usual CPLEX report of a remaining gap to the best LP lower bound as lower bound against which to test the heuristic because the gap to the LP relaxation is virtually meaningless. The best LP lower bound in such cases typically shows the MIP solution having 50 - 60% gaps, even seconds before an optimal termination is reached by branch and bound. This is because relaxation of the edge variables removes the most fundamental structure of the problem. The third type of termination are cases where an allocated amount of time running the unrestricted MTRS problem on CPLEX did not yield any improvement over the heuristic solution and the J3 objective value was used as an upper bound. Such cases have the IF indication standing for no feasible solution found in the time given. These cases can be read as meaning that *"the heuristic result could not be improved upon in X hours of CPLEX run time"*. In other words, the full problem remained completely infeasible in the allotted time when given the J3 heuristic result as an upper bound. These were generally unexpected outcomes especially given the length of time the full problem would run without even reaching feasibility. Usually providing the result of a heuristic to upper bound the optimal solution is expected to reduce the search space and thereby speed discovery of superior or an optimal feasible solution. However, even with this benefit the full problem instances remain infeasible in the IF cases. It seems that in these cases the solver cannot find even one feasible low-weight edge vector (i.e., one that uses relatively few edges but still forms a closed connected topology) on which edge costs alone are under the J3 value. This is a significant observation that influences the further heuristics to follow.

Table 9-2. Test Case Results with Three-part Heuristic for MTRS.

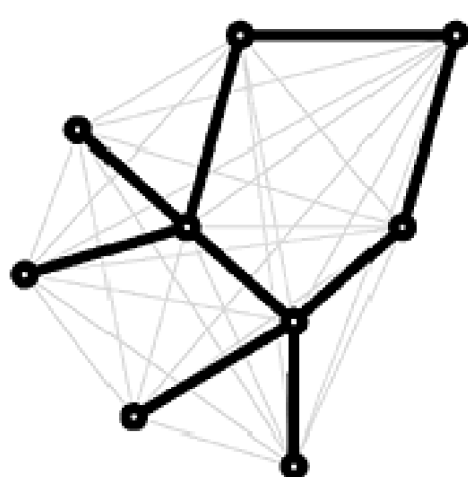
No.	Test Case	Step W1		Step S2		Step J3		MTRS Ref. Solution			Gap (%)
		Obj.	CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.	CPU	With Bounds	
1	9n36s1 -15	12 416	156.08	9 367	3.33	19 056	1.52	18 476	6 h (TL)		3.1
2	9n36s2 -15	11 996	177.33	11 329	14.54	18 809	1.81	18 811	6 h (TL)	18 h (IF)	0.0
3	9n36s3 -15	11 346	70.23	8 328	5.27	17 956	1.04	18 020	6 h (TL)	6 h (IF)	0.0
4	9n36s4 -15	12 621	300	9 954	10.98	20 560	1.39	19 094		73 h (FT)	7.7
5	10n45s1 -15	14 464	2608	11 050	28.09	22 964	3.85	23 691	6 h (TL)	6 h (IF)	0.0
6	10n45s2 -15	14 542	1595	10 100	40.58	23 300	2.42	23 471	6 h (TL)	6 h (IF)	0.0
7	10n45s3 -15	14 463	1985	12 340	22.76	21 160	7.11	26 416	6 h (TL)	18 h (IF)	0.0
8	10n45s4 -15	14 384	1077	12 400	105.74	22 850	12.33	29 309	6 h (TL)	18 h (IF)	0.0
9	15n28s1-20	16 459	402.75	15 617	57.73	27 841	51.08	26 131	477s (FT)		6.5
10	15n56s1 -20	13 933	900	10 069	244.40	22 225	8.21	25 248	12 h (TL)	6 h (IF)	0.0
11	15n26s1 -40	18 749	900	12 461	61.90	29 005	9.25	31 134	6 h (TL)	6 h (IF)	0.0

[Table 9-3](#) records information about the number of edges in the topology as it evolved under Steps W1-J3 in the heuristic and compares this to the number of edges in the attempt at a reference solution for the full problem. The S2 column in [Table 9-3](#) records the total number of edges in the S2 solution graph including the edges inherited *and used* from step W1. This is followed by the number of edges in the S2 graph that were *not in* the W1 solution. When an edge from W1 is not used (has zero spare capacity) in the S2 solution, the S2 edge total does not match the W1 total plus the number of "new" edges. Such cases are indicated with an asterisk. [Table 9-3](#) also records observations on the number and role of new edges added at S2 in terms of contributing to closure, and so on. The solutions of S2 could exhibit three types of edge changes relative to W1. An edge could be added in a way that provides both graph closure and bears spare capacity, or just bears spare capacity but does not contribute to closure of the graph. A third type of change from W1 to S2 is an edge that was present in the W1 topology, but is not logically present in the reserve network overlay design of S2. These are edges from W1 that bear no spare capacity in the S2 solution. [Table 9-3](#) also shows that, in all but one case, the J3 solution used all edges in the union of W1 and S2 edges. Only in problem 15n28s1-20 (Case 9) did J3 "rationalize" the edge sets from W1 and S2 in the sense of reducing from 23 candidate edges in the union of W1, S2 to 21 edges in the J3 solution. This was a somewhat unexpected tendency that is discussed below.

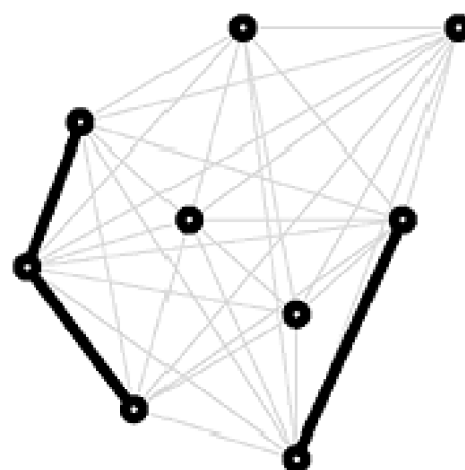
9.10.3 Discussion

The 9-node test cases yielded one fully optimal reference solution (case no. 4: 9n36s4 -15) with a gap of 7.7% for the heuristic. The heuristic, however, ran in about 5.2 minutes whereas 73 hours was needed to obtain the full termination reference solution. We also have a suggestion in [Table 9-2](#) that the heuristic solution was using too few edges (three less than the optimal solution). In two other 9-node problems, six and 18 hours were allocated to running the reference solution with the benefit of the upper and lower bounds from the heuristic, but no improvement was obtained over the heuristic within these times. In the remaining 9-node case (9n36s1-15), six hours of CPLEX time yielded a reference solution 3% better than the heuristic result which was obtained in 2.7 minutes. [Figure 9-20](#) illustrates the W1, S2, and J3 topologies for case number 4, and the optimum topology which is available for this problem. [Figure 9-21](#) gives the same portrayal for case number 6, and the optimal solution attempt after six hours.

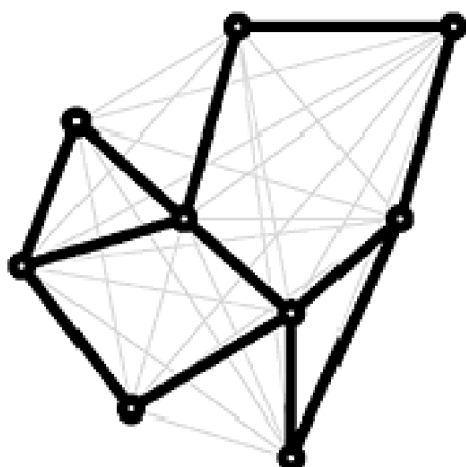
Figure 9-20. W1, S2, J3, and Optimal MTRS solution topologies from Case 4: 9n36s4-15.



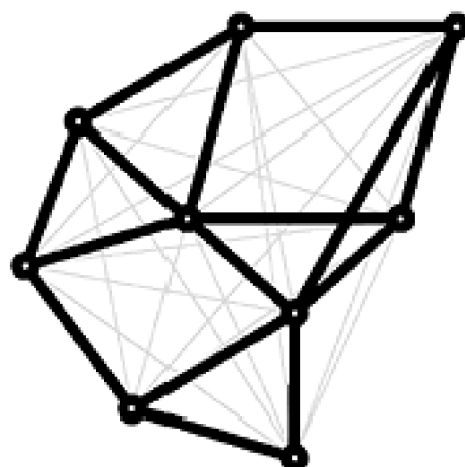
(a) - Step W1 (9 edges)



(b) - Step S2 (new edges only) - three edges added



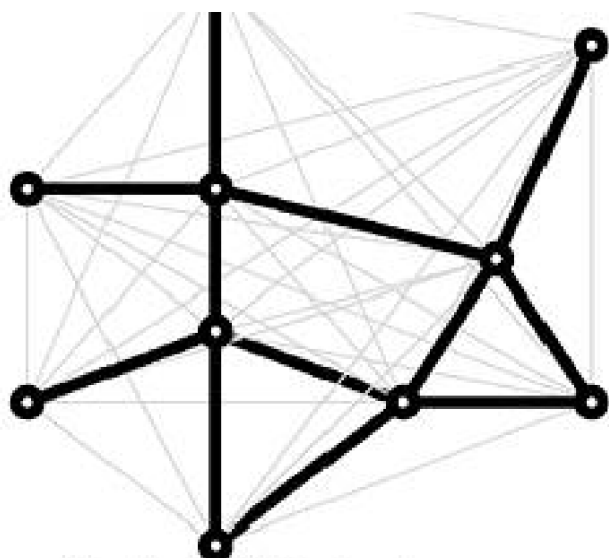
(c) - Step J3 (12 edges)
5.2 minutes, Obj = 20 560



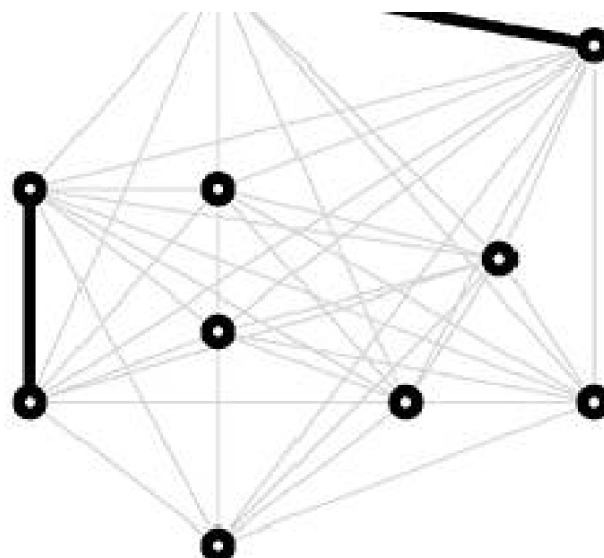
(d) - MTRS (optimal) (15 edges) 73 hours,
Obj = 19 094

Figure 9-21. W1, S2, J3 and J0 Topologies from Case 6 (10n45s2 -15).

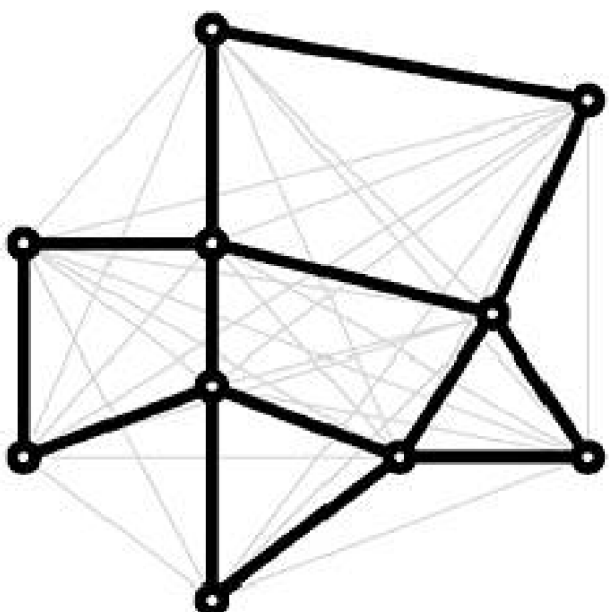




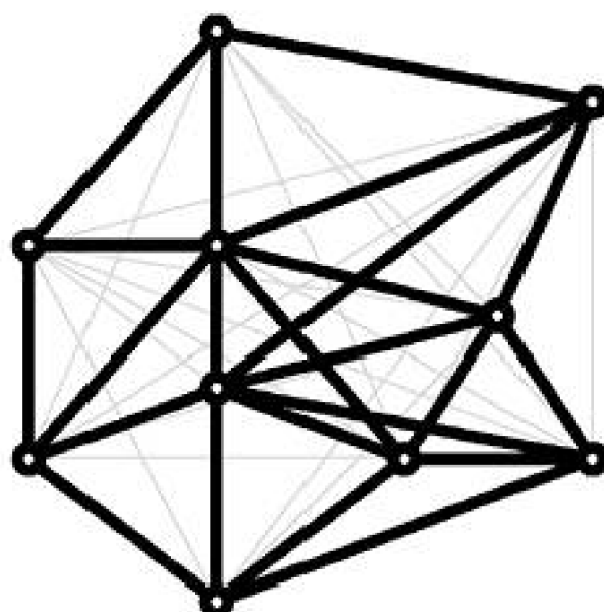
(a) - Step W1 (12 edges)



(b) - Step S2 (two new edges)



(c) - Step J3 (14 edges)
27.3 min, Obj = 23 300



(d) - MTRS (sub-optimal) (23 edges)
after 6 hours, Obj = 23 471

The initial set of 9-node problems confirm as expected that S2 and J3 run very quickly while the overall time is dominated by FCR in W1. This suggests the strategy of time-limiting W1, which is tested in the 15-node problems. It was also noted that in all four 9-node cases the J3 solution uses *all* edges promoted for its consideration by prior steps W1 and S2. This was somewhat unexpected as it was thought initially that the set union of edges from W1 and S2 would tend to overpopulate the candidate edge set, leading to a reduction in J3. Related to this is the observation in [Table 9-2](#) that the reference solutions, where obtainable, consistently use more edges than the heuristic employs. This suggested further tests (not included here) where we artificially lower Ω to try to inflate the number of edges used in W1 and S2. However, detailed inspection of results at $\Omega = 15$ shows that the design tends to be dominated by capacity costs not edge costs, so there may not be too much significance in the topology differences between full MTRS and J3 in this case.

Table 9-3. Number of Edges Employed at W1, S2, J3 Stages of the Three-part Heuristic for MTRS

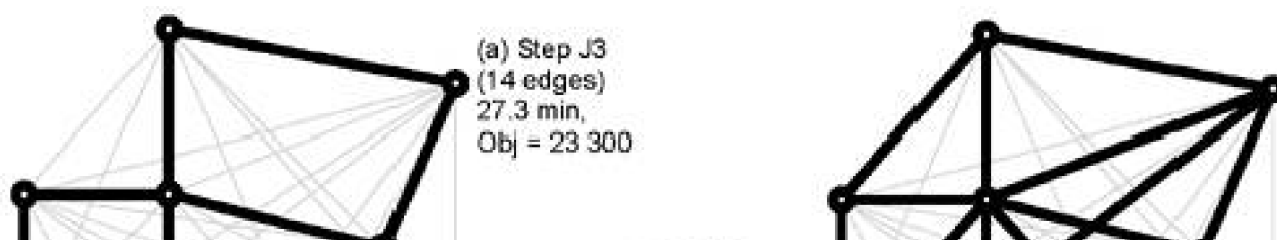
No.	Test Case	Step W1	Step S2 (tot,new)	Step J3	MTRS Ref. Sol'n	Notes on edge evolution W1-S2
1	9n36s1 -15	9	13, 4	13	17	all added edges at step S2 were for closure.
2	9n36s2 -15	9	14, 5	14	n/a ^[a]	only 4 edges of the 5 added edges needed for graph closure.
3	9n36s3-15	10	13, 3	13	n/a	S2 3 edges added, all effect graph-closure.
4	9n36s4-15	9	12, 3	12	15 (optimal)	S2 3 edges added, all effect graph-closure.
5	10n45s1-15	11	15, 4	15	n/a	S2 2 of 4 added edges effect closure.
6	10n45s2-15	12	14, 2	14	n/a	S2 2 edges, both effect closure.
7	10n45s3 - 15	10	13*, 6	16	n/a	S2 5 of 6 edges effect closure, 1 is non-planar, 3 edges from W1 disused.
8	10n45s4-15	11	13*, 5	16	n/a	all 5 edges effect closure, 1 non-planar, 2 edges from W1 disused.
9	15n28s1-20	15	22*, 8	21	20 (optimal)	6 of 8 edges added for closure, 1 from W1 disused.
10	15n56s1-20	16	20*, 5	21	26 (at 12 h)	4 edges added for closure, 1 from W1 disused.
11	15n56s1-40	15	19*, 5	20	22 (at 6 h)	4 of 5 edges added for closure, 1 edge from W1 disused.

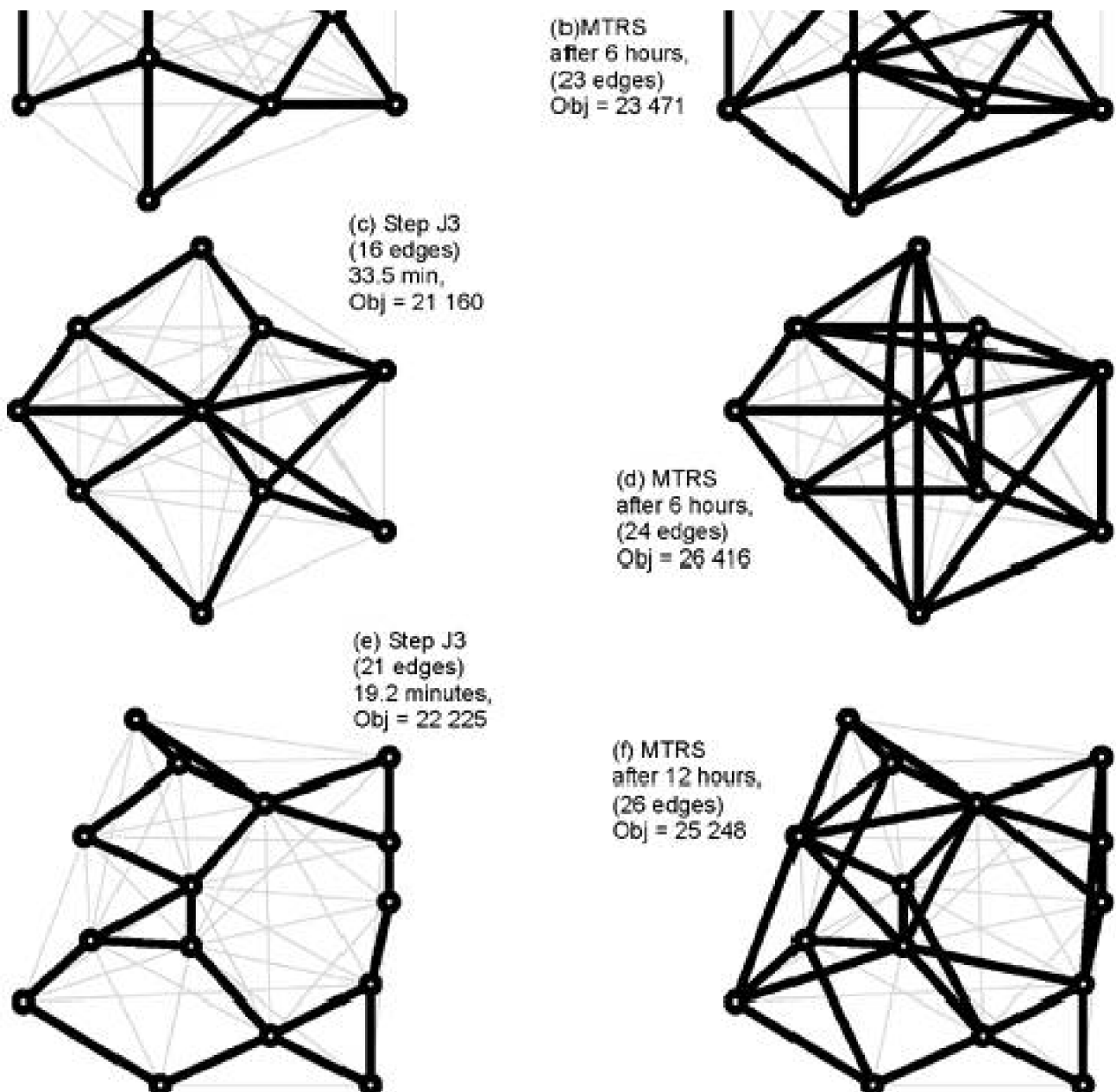
^[a] n/a = not available because no feasible solution was found by CPLEX that improved upon the heuristic result in the time available when MTRS was provided with J3 objective as an upper bound.

Cases 5 to 8 are the 10-node problem instances with complete edge sets and demand matrices. In these cases, we again see that J3 employed all edges provided to it from W1 and S2, suggesting that the first two heuristic steps may not be promoting a large enough set of edges to consider in J3. However, all four 10-node results were cases where six to 18 hours of CPLEX time on the MTRS reference problem (with bounds) could not improve on the heuristic results. When the upper bound was removed (for validation purposes) the MTRS problem immediately found feasible solutions but given six more hours reached objective values that were still 7% to 28% above the corresponding heuristic result! It is only in this sense that the "gap" for these results is reported as 0.0 because the gap is against a time-limited MTRS attempt.

In [Table 9-3](#) the full MTRS problem was run with the benefit of the J3 objective as an upper bound, in an attempt to give CPLEX the head start toward a fully optimal termination. In these circumstances, however, we obtain no feasible result at all. On the other hand, we wished to view the unrestricted sub-optimal solution as it neared the six to 18 hour time limits, so we repeated the time-limited full MTRS runs without the bounds. A sample of the results comparing topologies is shown in [Figure 9-22](#). The most consistent and striking observation is that, even after six to 12 hours, the best feasible MTRS solution costs more than that found by the heuristic in minutes. Moreover, in every case tested the time-limited attempt at full MTRS clearly has too many edges to be even plausibly near optimal. The optimal solution attempt is still searching in high regions of the topology space after six to 18 hours. In contrast the heuristic, by its nature, is directly guided into a region where topologically feasible arrangements of a not-excessive number of edges are immediately at hand.

Figure 9-22. Example topologies from the 3-part heuristic compared to best feasible solution from attempt on optimal MTRS solutions after six and twelve hours.





The fact that the unrestricted problem fails to even reach *feasibility* in six to 18 hours when the J3 objective value is supplied as a bound suggests that simply finding a closed connected graph on the relatively few edges (the basic condition for feasibility) may be the most difficult purely combinatorial aspect of the entire problem. With the bound present, the solver appears to be searching almost at random for a low-weight combination of edge variables that describe a closed connected graph. The space of edge combinations that are not even plausible swamps the ability of the MIP solver to progress toward finding even one closed connected graph on the typically few edges that are associated with near-optimality. Without the upper bound, the computational prospects are even worse. The solution becomes feasible but now a vast number of graphs are enumerated at far too high an edge count, and an investment of time is made in each of those for routing and capacity considerations.

Our interpretation of this repeatedly observed effect is that the MIP solver gets bogged down in high-weight edge-vector space. Simple discovery of low-weight edge vectors that describe closed connected graphs (where the optimum solutions really lie) is quite difficult for a branching search on the edge variables. And because the full LP relaxations are so weak as lower bounds, the solver's progress is relatively unguided. Hence the solution tree nodes it visits tend to be high weight edge vectors simply because these are statistically much more frequent in the population of all possible closed connected graphs.

To explain further: combinatoric principles would have it that there are vastly more arrangements of closed connected graphs with many edges, than there are graphs that have relatively few edges *and* also describe a biconnected closed graph. In other words, there are simply far more arrangements of graphs with—for example—24 edges, such as in [Figure 9-22\(c\)](#), than with 16 edges, like [Figure 9-22\(d\)](#). For reasoning only, consider the MIP search on the edge vector as an almost random walk through the edge-vector space. (It is never really random but with such weak LP bounds to a 1/0 problem such as this, this could be close to the net effect). Consequently, the probability that any given branching on the edge variables is even plausible as solution graph is extremely low. While this is a

simplification, it seems to describe the solver's inability to escape the combinatoric dominance of the overly connected graphs in the topology space to find even one instance of a low weight biconnected edge vector. Using [Figure 9-22\(c\)](#) and [\(d\)](#) as examples, it is clear that the MIP solver has yet to even discover *one* instance of a graph similar to those in to (c) after six and 12 hours, respectively. The solver is wading around in a huge combinatoric space of edge combinations in which lightweight closed and connected graphs (like [Figure 9-22 \(c\)](#)) are extremely rare. Any guidance effect the solver is getting from the loose LP relaxation is swamped by the combinatoric dominance of graphs like [Figure 9-22\(d\)](#) as opposed to [\(c\)](#).

This line of reasoning was checked in some experiments by Ezema [[Ezem03](#)]. Two sample results convey the general point. First consider the universal edge set on a 16 node problem. There are 2^{120} possible edge vectors on 16 nodes, obviously too many to enumerate by any means. However, if an unbiased sample of 10^6 possible edge combinations is considered, it is found that only 16,339 combinations (1.6%) are biconnected graphs of 40 or fewer edges. Forty edges is taken here as a criterion for a graph being "lightweight" in that this corresponds to $d=5$ for 16 nodes. More interestingly, when 20 nodes are considered, the sample of 1,000,000 edge combinations yields zero graphs that even have fewer than 40 edges, let alone are connected or biconnected. At 28 nodes, the probability of a randomly generated edge vector having $2 < d < 6$ is $\sim 10^{-30}$.

Cases 9 to 11 are 15-node problems sharing a common set of node locations and demand matrix. Only the number of candidate edges and Ω are changed. The Case 9 problem included as candidate edges only the 28 edges of the actual network from [Bell93](#). Case 9 has an optimum reference result obtained in 477 seconds. In this case the heuristic showed a solution gap of 6.5% and a total run time nearly equal to that for the optimal result. This suggests that when the candidate edge set is highly constrained, one can attempt to solve MTRS directly rather than use the heuristic. As before, the heuristic runtime is dominated by the FCR instance in W1.

Cases 10 and 11 are on the same 15 nodes but have a total of 56 edge candidates. This represents 53% of the universal edge set. Now the solution space is on all combinations of 15 or more edge choices from 56 candidate edges. Based on this increase, and initial indications of high run times for W1, we time-limit the W1 step to 15 minutes. The results in [Table 9-2](#) suggest that this is a viable tactic within the heuristic framework: with time-limited W1 stages, the heuristic results, obtained in under 25 minutes, were 7 to 14% *better than* the unrestricted MTRS solution attempt at 6 and 12 hours, for $\Omega = 40$ and 20^* respectively. A more extensive study of the heuristic can be found in [[GrDo01](#)].

9.10.4 Insights from the Three-Part Heuristic

Results show that the 3-part heuristic is an effective tool in its own right. But our experience with it also gives us the some further insights. These are summarized as follows:

The runtime of full MTRS problem, and the dominant W1 FCR step of the heuristic, depends above all on the number of candidate edges. Every additional edge variable doubles the size of the solution space. Hence, anything that can be done ahead of time to limit the number of edge candidates (without affecting solution quality) is highly effective. This leads to the highly-effective edge-limiting preprocessing heuristic that follows and can be used in conjunction with the 3-step heuristic, with J3 directly (on the reduced edge set that it provides), or in principle with any topology solution method.

Computational evidence (and combinatorial explanations above) both indicate that what overwhelms the MIP solver is the difficulty of even discovering biconnected graphs that use as few edges as are typical in a transport network, e.g. typically $d < 6$ with certainty for real networks. Separate assessments confirm that the vast majority of edge vectors that describe a biconnected graph also include far too many edges. Conversely for a network of moderate size, edge vectors that specify a plausible number of edges have almost no chance of also describing a single biconnected graph component. And yet while the MIP has to branch rather blindly in this $1/0$ space, it is relatively easy for us to generate such graphs directly by procedural means. Therefore the search of topology space is perhaps the best aspect for us to take away from the MIP. In subsequent methods we procedurally control the topology search subproblem, not even leaving the space of qualified graphs. This search of the qualified graph space is coupled to a MIP solver only for routing and capacity cost evaluation on each candidate graph.

A final point is about the number of edges used in J3 solutions relative to the total set promoted by W1, S2 stages and the insight this gives us about the initial hypothesis that was made regarding MTRS topologies. It was not initially expected that J3 would often retain all edges or remove only one or two. It was initially thought the union of W1, S2 edge sets would significantly overpopulate the J3 edge set allowing J3 to make extensive and significant final cuts. Certainly a large edge set being strongly reduced by J3 would be nice to see intuitively because it gives greater confidence of the scope of final optimization that occurred in J3. Indeed we initially viewed it as a failing that this was not happening and pursued methods to deliberately "overpopulate" the W1-S2 edge set. But doing so did not lead to significant improvements in solution quality. We now appreciate, however, that it is actually consistent with the original hypothesis of the

3-step heuristic for J3 to eliminate few edges. The point is that if the original hypothesis is in fact strong (i.e., that MTRS is close to an FCR solution that is just made biconnected), then the J3 solutions *should* tend to use every edge from W1-S2. Observing this simply confirms the initial hypothesis that the solution is closely approximated by good FCR edges complemented by a least-cost addition for biconnectivity and restorability.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

9.11 Ezema's Edge-Limiting Criteria

We have seen abundant evidence that what dominates the complexity of MTRS is the size of its set of candidate edges. Let us reconsider the 10-node problem mentioned in [Section 9.7.3](#). As mentioned there are strictly $2^{45} = 3.5 \times 10^{13}$ possible edge decision vectors if we consider the universal edge set. If, for arguments sake, we could evaluate a million of these combinations per second, it would take over a year to consider all possible edge combinations. But say, for some reason we knew *a priori* that nine of the 45 edges could be eliminated with confidence that they would never appear in a good solution in any case. The reduction to 36 candidate edges reduces the number of edge vectors to $2^{36} = 6.9 \times 10^{10}$. Now the same postulated computer can evaluate all combinations in only 20 hours. Thus, anything we can do *a priori* to eliminate candidate edges from even being considered should be highly effective. In this regard, two things we do know with considerable empirical confidence are:

- It is extremely unlikely that an optimum topology for any transport network contains single spans that reach directly across the entire network diameter. For example, Edmonton is much more likely to have direct physical layer connections to Calgary and Vancouver, than to New Orleans.
- It is extremely unlikely that any one node would have nodal degree higher than some practical value that experience dictates such as, for example, $d_{max} = 8$.

These two considerations can be combined into a family of simple *a priori* edge-elimination criteria that can greatly benefit solution times with almost no effect on solution quality. The simplest edge elimination procedure is to visit each node in arbitrary order, select the d_{max} incident edges at that node which are of shortest length, and remove all other edges incident on the node from the edge set. For example

with $d_{max} = 5$ we keep only the edges to the five nearest neighbors in Euclidean space. This results in an edge set that had ≤ 5 for every node in terms of the candidate edges remaining. Under this procedure individual nodes can wind up being left with less than d_{max} incident edge candidates because an edge retained at one node may later be rejected as not being in the five shortest edges at the other attached node. This most simple of criteria nonetheless yields impressive performance gains. [Table 9-4](#) shows its effect on solution quality and run times on problems up to 11 nodes using both MTRS and the 3-step heuristic.

Table 9-4. Performance of MTRS and the Heuristic with a $d_{max} = 5$ Edge Limitation

N	MTRS (all edges)		MTRS (reduced edge set)		3-Step Heuristic (all edges)		3-Step Heuristic (reduced edges)	
	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)
7	191358	47	191358	9	191358	4	191358	4
8	224389	5721	224389	28.7	224389	7	224389	5
9	299673	333451	299673	84	304383	19	304383	6
10	n/a	IF (10 days)	372723	92	378567	153	378567	15
11	n/a	IF (10 days)	430347	800	418057	602	430347	36

In [Table 9-4](#) the optimal or best found solutions are shaded for reference. MTRS solutions with universal edge sets were only obtainable up to 9 nodes. Such full MTRS problems at 10 and 11 nodes had still not reached feasibility after 10 days of attempted runtime. With simple $d_{max} = 5$ edge limitation, the edge decision spaces were reduced to 23, 21, 19, 18, 17 and 15 edge variables, respectively, for $N=11$ down to $N=7$ nodes. Note that most of the edge-limited edge sets have somewhat fewer than $N d_{max} / 2$ entries. This arises as explained from the elimination of one or more edges that would be retained from one node's local viewpoint, when revisited from the neighboring node. At $N = 11$ the candidate set of 23 edges from edge-limiting is a reduction of the edge space by 32 edges, or a factor of 2^{32} in the

solution space and yet we can see there was no impairment whatsoever in the 7, 8, and 9 node problems where fully optimal MTRS reference solutions are available. MTRS with the reduced edge set also produced the best solution at $N=10$ nodes, but at $N=11$, the 3-step solution with all edges is lower in cost than MTRS with reduced edges, so the solution quality was affected by edge limiting in this case. The reason is explained shortly, but before leaving, note nonetheless the impressive performance of the 3-step heuristic combined with a *priori* edge limitation. All problems up to $N=11$ were solved in under a minute, either optimally or with a gap of less than 3% to the best solution found.

We can, however, easily improve the edge limiting heuristic. Inspection of the case above where MTRS on the reduced edge set was suboptimal showed that a necessary edge had been eliminated because the application of the elimination of candidate edges at nodes is local and not coordinated with eliminations at other nodes. An important retained edge at one node may be later eliminated when revisited at another node, making the degree of the first node too limited. Another issue with the simple reduction method is that it does not allow for the kind of "natural hub node" that we saw in the discussion of MTRS solutions ([Section 9.8.2](#)). In that context we saw the emergence of a $d=4$ "hub" in a 6-node problem. In real transport networks we occasionally see such hubs with individual degree considerably higher than the network average, often with a central location and many short edges to other nodes. These two considerations lead an improved edge limiting procedure. First we sort all candidate edges in ascending order of cost. We then determine the number of edges m that corresponds to our view of the maximum average connectivity of a plausible solution for the network as a whole (no longer for individual nodes), d_{max} . Then we remove edges longer than the m^{th} edge (starting with the longest) as long as each end node of an eliminated edge have an individual number of incident candidate edges greater than some number k . This is called (m,k) edge limitation. In summary the process is:

Edge Limit (m,k)

Sort the candidate edge set in ascending order based on

length (or cost) of edge;

Select the $m = d_{max} |N| / 2$ shortest edges;

For all other edges, starting with the longest:

Remove if both end nodes have at least k neighbors within the candidate edge set.

This procedure makes sure that every node has the potential to be at least of degree k , while allowing individual "hub" nodes with considerably higher degree. Nodes with short edges are free to keep all their edges. At the same time it exploits our *priori* knowledge about a plausible d_{max} for the network as a whole, to achieve significant reduction of edge sets. Through experimentation in [Ezem03](#) an overall policy for edge limitation that can be recommended is:

Equation 9.45

$$m = 0.3 \cdot |N| \cdot d_{max}$$

$$k = 3 \text{ for under 11 nodes}$$

$$k = 4 \text{ for 11 or more nodes}$$

Using this policy with $d_{max} = 5$ the $N=11$ to $N= 7$ problems considered above have reduced edge sets of 26, 19, 17, 15, 12, 10 edges, respectively, and the reduced MTRS solution at $N=11$ in [Table 9-4](#) improves to match that of the 3-step heuristic—and is believed to be optimal. In addition, all of the prior 7-, 8- and 9-node test cases used for the study of the 3-stage heuristic in [\[GrDo01\]](#), for which optimal MTRS solutions were obtained, are exactly reproduced by MTRS with the reduced edge sets here. The solutions are, however, obtained in under a minute with the limited edge set whereas the same results took hours or days with the full edge sets for MTRS. On the 10, 11, 15, 20, and 26 node test cases unrestricted MTRS solutions are not available but edge-limited MTRS solutions were obtained in at most two hours that outperformed four hour time-limited attempts at full MTRS by up to 35% in solution quality.

[Figure 9-23](#) shows the effect of edge limitation with $k = 3$ and $k = 4$ ($d_{max} = 5$) for a 7-node problem. No edge from the strictly optimal solution found with MTRS is omitted from the reduced edge set. [Figure 9-24](#) shows the effect of applying the same edge limitation policy to

the prior 15-node, 56 edge, test case used for trials of the 3-step heuristic. The 56 initial candidate edges are reduced. All edges of the near optimal solution (found from the whole edge set) are included in the reduced edge set.

Figure 9-23. Edge limitation applied to the universal edge set for a 7-node problem.

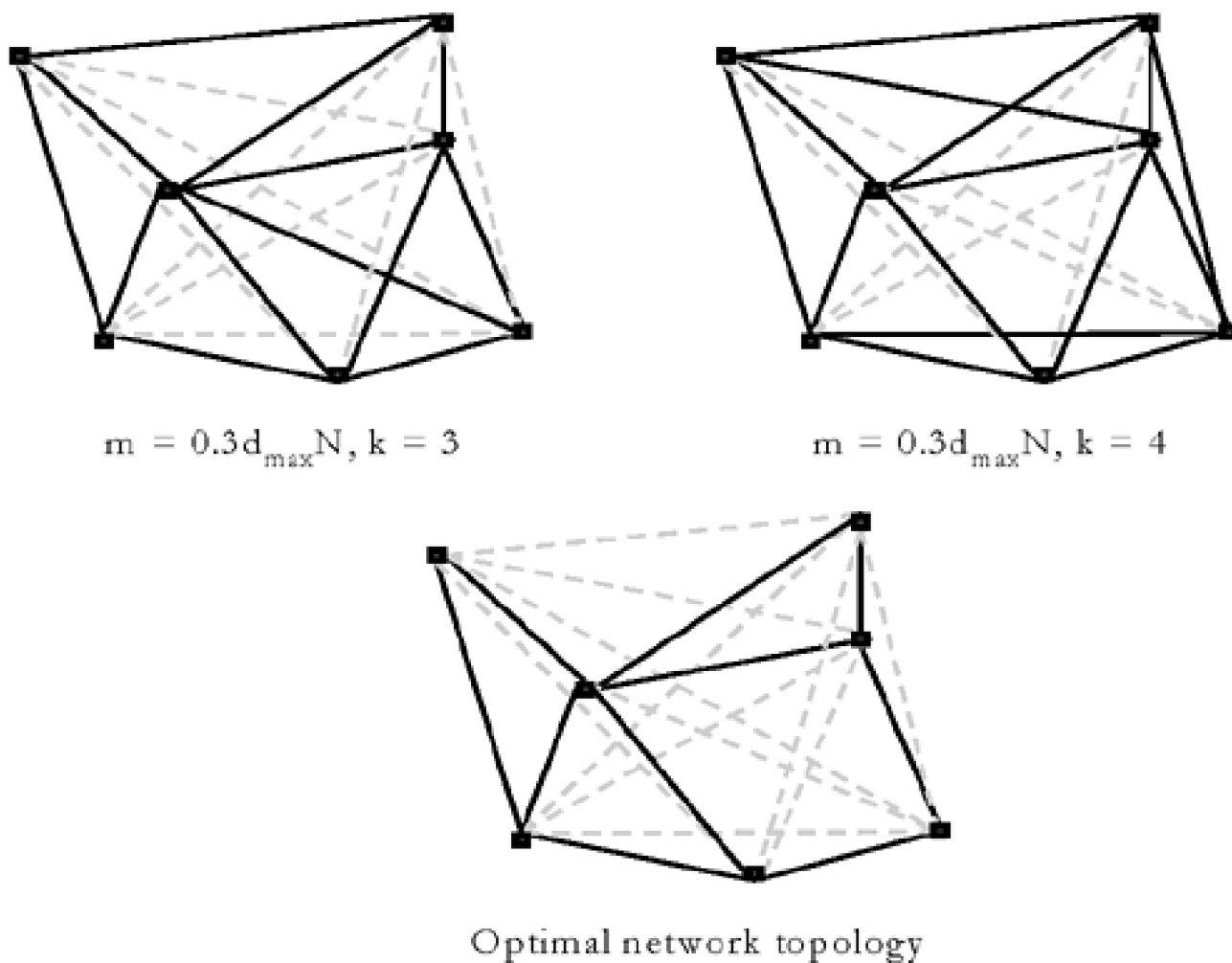
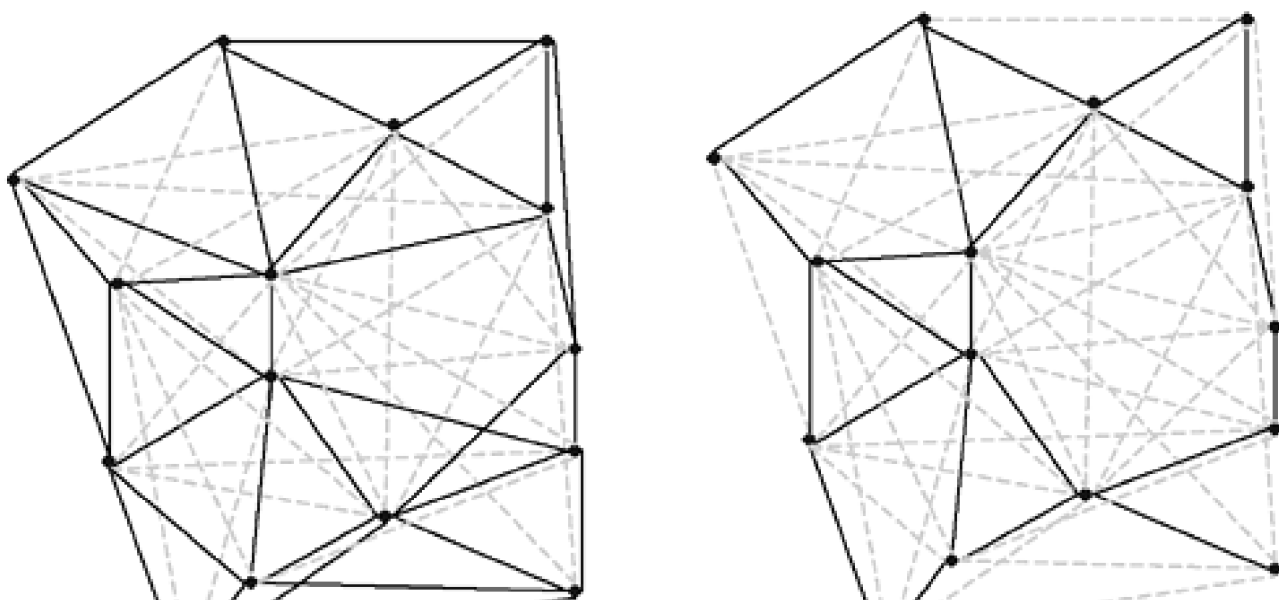
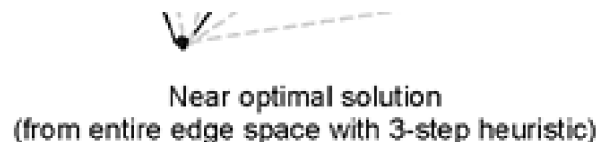
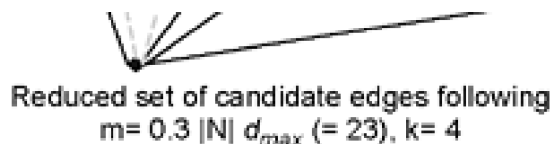


Figure 9-24. Example of edge limitation applied to a 15-node problem [Ezem03].





The edge reduction guideline of [Equation 9.45](#) can also be viewed simply as an initial criterion that can be followed by any further limitations necessary or warranted. For instance with $k=4$, $d_{max} = 5$ the edge space of a 26 node problem is reduced from 190 candidate edges to 76. Using heuristic methods that follow we may be able to work with 76-edge candidates, but often that will still be too many to be practical. Seventy-six candidate edges corresponds, however, to an actual ratio of candidate edges to nodes of 7.6. Even though we used $d_{max} = 5$, the large number of extra candidates are admitted by the $k=4$ criterion, so $k=3$ could be asserted instead. Ultimately of course, case by case inspection of the edges for "plausible appearance" in an optimal solution can be used to further reduce the edge set.

Additional criteria for acceptance or rejection can be based on absolute properties of each edge candidate. All edges below a certain low absolute distance or cost may be considered worthwhile exceptions to the basic rule. Conversely, if the maximum geographic diameter of a network region was say 1,000 km, or the average nodal distance to a nearest neighbor was (say) 200 km, then one might assert that in addition to any eliminations by the basic d_{max} criterion, any single prospective edge over 800 km will not be considered.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

9.12 Successive Inclusion Heuristic

The Successive Inclusion (SI) heuristic (initially called Iterative Sampling in [Ezema03]) offers a way to check or partially guard against greediness effects from edge limiting and/or to permit more aggressive initial edge limiting as may be necessary on larger problems. The idea is to provide an initial set of candidate edges and solve for the optimum topology within that edge set to start. Then, each edge that was part of the true universe of edges for the problem but was not included in the initial design is readmitted for consideration in a series of relatively fast subsequent runs of the topology design problem. The overall process is still suboptimal but, in practice, it performs quite well. In effect when each initially excluded edge is readmitted individually, we are asking the solver, "Would you change your mind about anything if you had *this* edge to work with too?" Of course this misses the possibility that edge x would only be chosen in the presence of edge y , both of which were initially excluded edges, but it does "mop up" on any first order opportunities to include good edges that were initially omitted. To the extent that certain elements or choices tend to be "good ideas" in combinatorial problems, this heuristic will be effective. What we mean by this is that if an edge tends to be a "good idea"—in and of itself, set in almost any backdrop of other edge combinations—then Successive Inclusion will work well to backstop or improve on the quality of edge-limited designs, because it makes sure that the edge limiting process did not exclude any really important (i.e., useful) edge.

Successive Inclusion can actually use any initial (biconnected) topology design and universe of edges. We assume that MTRS with reduced edge sets, or the 3-step heuristic for MTRS are used as the basic solution technique in conjunction with Successive Inclusion. First, let us define:

- E_{univ} = the set of *all* candidate edges for the problem.
- E_{lim} = the edge-limited set of edge candidates. Obviously $E_{lim} \subset E_{univ}$.
- $E_{opt}(n)$ = the set of edges included in the n^{th} iteration.
- $MTRS(E)$ = the MTRS solution procedure given edge set E , returns $E_{opt} \subset E$.

Then the procedure is:

Successive Inclusion:

```
n:=1; Eopt(n):= MTRS(Elim);
for every edge i in {Euniv-Elim}: {
    Eopt(n+1):= MTRS(Eopt(n) ∪ i);
    n:=n+1}
```

The reason this works well is the understanding that an edge solution found from an edge set E can never worsen when presented with a larger edge set $\{E+i\}$; it can only improve. Thus, in the first step above, an initial topology is found within the reduced edge set. In the main loop that follows, each edge that was initially excluded is individually reintroduced. The new edge could be included if its effect is to reduce capacity costs more than its own cost, or MTRS may revise its prior edge choices given the opportunity to consider $\{E_{opt} + i\}$ and adopt the new edge i , while dropping one or more other edges from the prior solution. If edge i is not adopted we can be satisfied that its addition

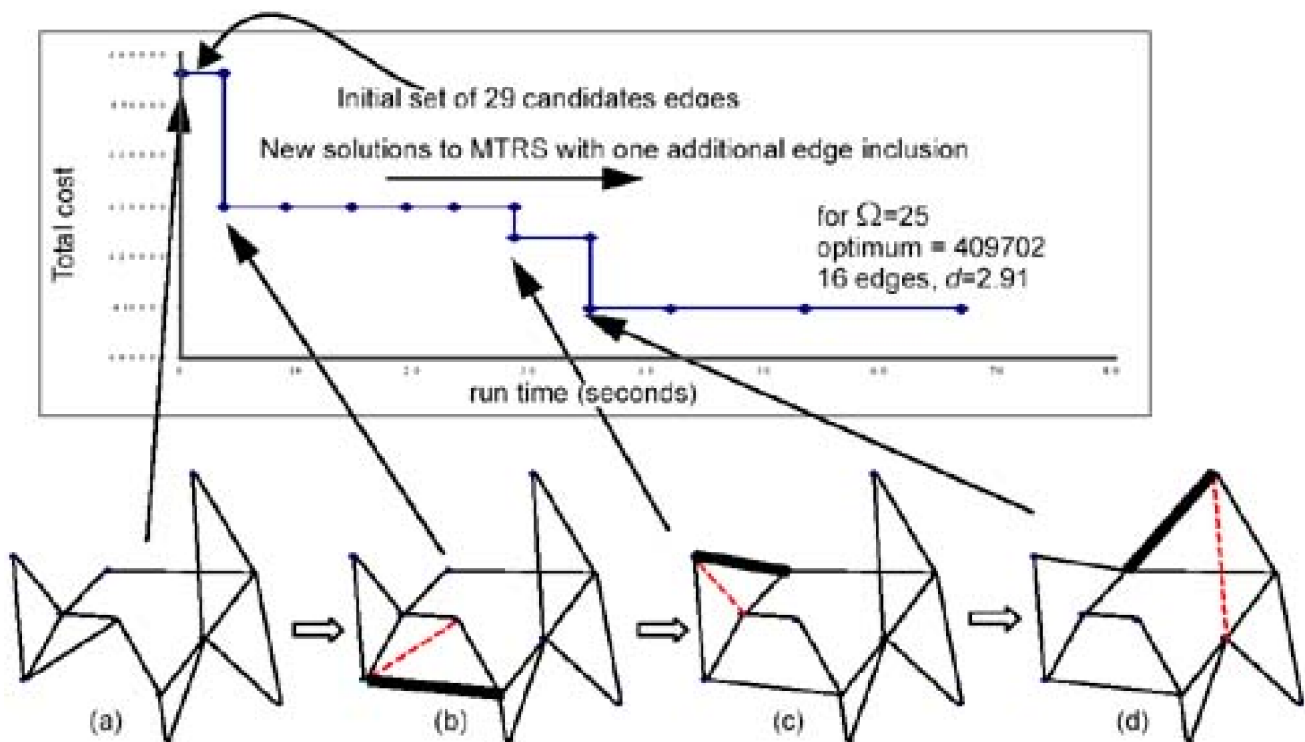
(individually) to the initial edge set would also not have had any effect because the solution cost of $MTRS(E)$ is always $\leq MTRS(E+i)$. Any improvements found in the main loop are implicitly retained in $E_{opt}(n)$.

To use Successive Inclusion it is still always best to solve the initial MTRS problem on the largest reduced edge set that is practicable. But SI allows us to approach larger overall problems than we otherwise could tackle at the same solution quality. For instance an 11-node problem with its universe of 55 edges might be edge-limited to 28 edges. Twenty-eight edges is equivalent to the universal edge set for an eight-node problem, which we can solve directly with MTRS in about a minute. Say this results in selection of an initial topology with 20 edges (i.e., $\bar{d} \sim 3.6$). The initial MTRS run is then followed by $55-28 = 27$ further runs of MTRS with (nominally) 21 edges (in principle the number of edges could go either back up or down). Assuming the average edge size of subsequent runs stays about 21, these MTRS problems are roughly $2^{(28-21)} = 2^7$ times faster than the initial MTRS run. The overall procedure is thus vastly faster than the initial problem involving 2^{55} edge combinations. And although strictly suboptimal, in practice it often does match the fully optimal MTRS solution where the latter are available.

Note that there are several options for the initial design step in SI. It could be a full MTRS solution on the reduced edge set, or, an application for the 3-step heuristic for MTRS in which case a larger reduced edge set can usually be handled at the initial stage. Ezema used a minimum spanning tree, repaired for biconnectivity, with addition of a set of further candidate edges selected from edge-limiting as the $E_{opt}(1)$ initial edge set and observed SI to accumulate the added edges and make other revisions to the solution topology as needed to reach the optimal solution. [Figure 9-25](#) shows two selected results for the COST-239 network of 11 nodes and universe of all 55

possible edges. The problem uses the published COST-239 demand matrix, Euclidean distances as edge costs and $\Omega=25$. A set of 26 candidate edges was assembled from a repaired MST graph (17 edges, $d=3.09$) plus 9 edges from the universal edge set for this problem, chosen by edge-limiting. The topology shown at the end in [Figure 9-25](#) is the optimal topology with final $d=2.91$ (16 edges). Topology changes occur only when a new edge inclusion is used in the solution or the new edge permits removal of some existing edges, as in [Figure 9-25\(c\), \(d\)](#). The broken lines indicate edges that were part of a prior solution but were deleted following addition of a new edge. Several different sets of 26 edges were tested and all quickly converged to the 16-edge topology that we believe is actually optimal at the total cost shown based on separate runs of the 3-step heuristic on 39 edges out of the entire edge-set of 55.

Figure 9-25. Trajectories of SI runs on COST-239 11 node problem starting from repaired MST.

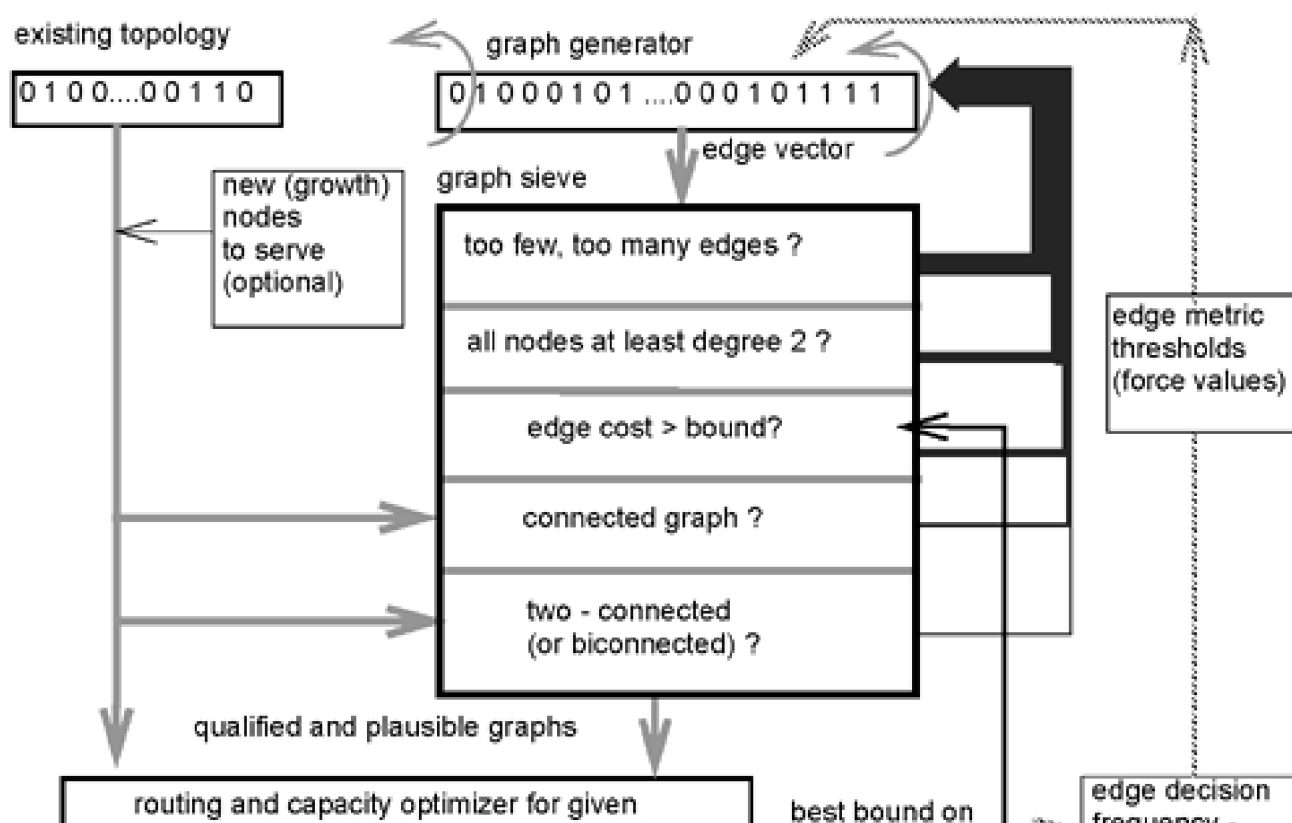


9.13 Graph Sifting and Graph Repair for Topology Search

The edge-limiting heuristic can be applied ahead of time to any MTRS-type problem. Whatever solution method is used, the inherent problem complexity is reduced, benefitting runtime for any type of subsequent solver. Indeed edge set reductions alone can be combined with the 3-stage heuristic to obtain high quality solutions to MTRS problems of up to around 20 nodes in practice, or 30 nodes if more aggressive edge set reduction is applied. But what if we want to solve even larger topology problems and/or we want a separate method to use as a cross-check on larger designs? To go to larger problem sizes it is quite clear that we have to entirely remove the topology search subproblem from the sphere of the MIP solver. In this section we explain a procedure to generate "qualified" random graphs to drive a search on topology in which the MIP is used only to evaluate the capacity costs associated with each candidate graph. The important difference is that our search will be over the space of qualified graph candidates directly, not single-edge decision variables. By no longer operating on the space of *edge* candidates we avoid the vast number of edge vectors that do not even specify a qualified graph, but which ultimately completely defeat the 1/0 MIP at larger problem sizes.

The basic idea is called "graph sifting." It is based on the realization that among the space of all possible free edge decision vectors, only a small fraction of these are even plausible or qualified for detailed consideration as the basis for a cost minimal transport network. More specifically we know that simply to be qualified, a candidate graph must be biconnected or two-connected. We also know that to be even plausible as the basis for an optimal solution in practice, the graph must be relatively lightweight. The initial sifter idea is thus to implement a series of extremely fast and simple procedures to apply increasingly targeted but easily implemented sifting criteria, so that all that falls out at the bottom of the sifter are plausible topologies for detailed routing and capacity design. The graph sifter can work directly to enumerate and discover all qualified graphs directly by sifting on suitably sized problems. At larger problems, however, so much sifting time can be spent before finding even one qualified graph that a better strategy is to "repair" unqualified graphs, thereby producing a series of candidate graphs. The edge cost of each candidate is trivial to evaluate and the MIP solver can quickly assess the capacity cost of any given candidate graph. The concept of the sifter alone is portrayed in [Figure 9-26](#). The logic of the graph sifter coupled with a graph repair process is illustrated in [Figure 9-27](#).

Figure 9-26. Concept and architecture of graph sifter approach to optimizing topology.



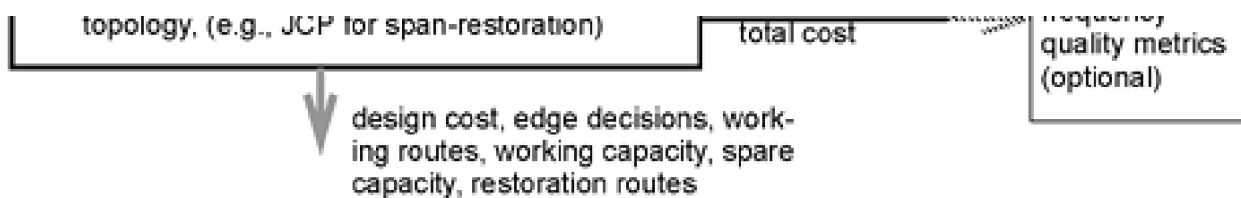
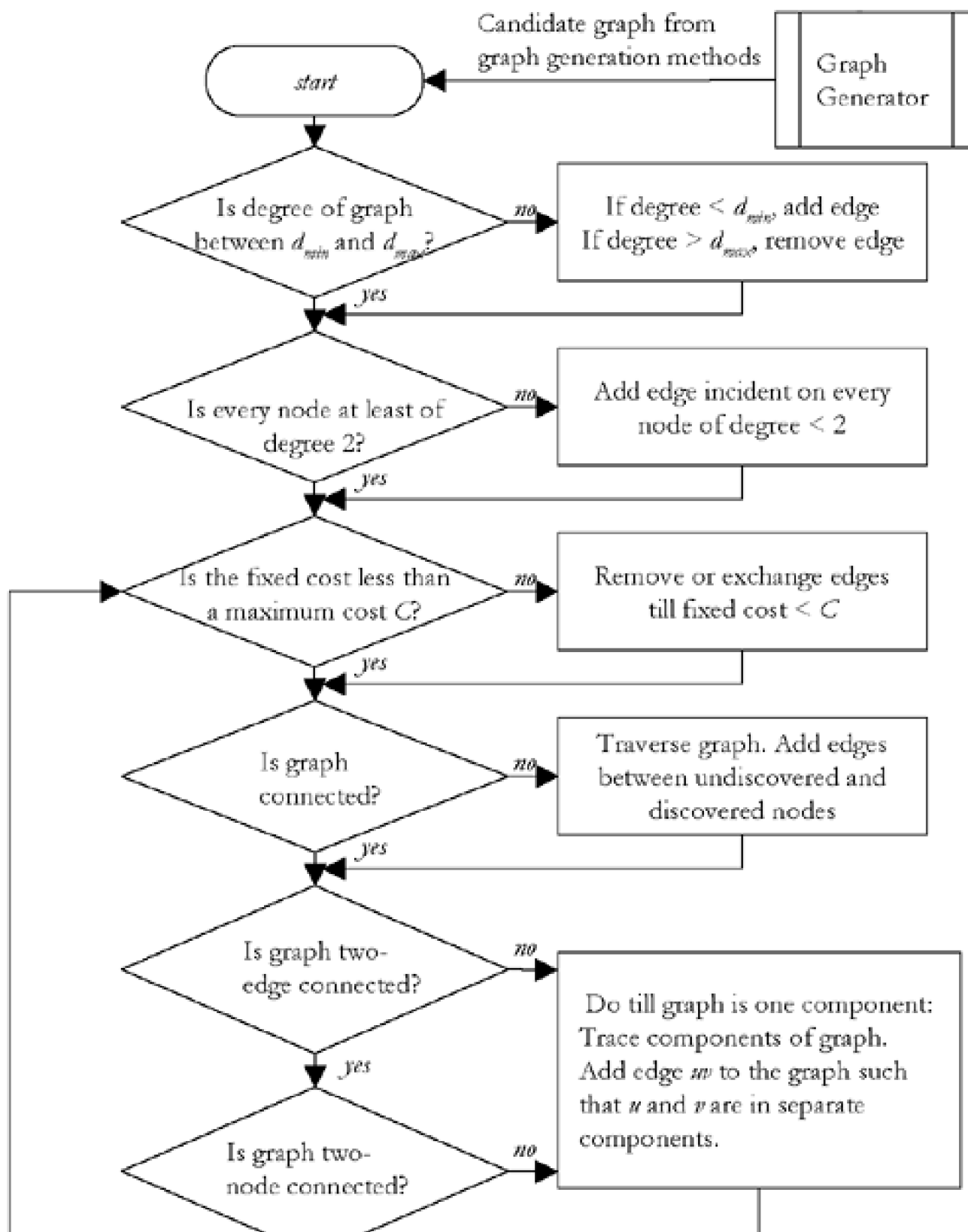
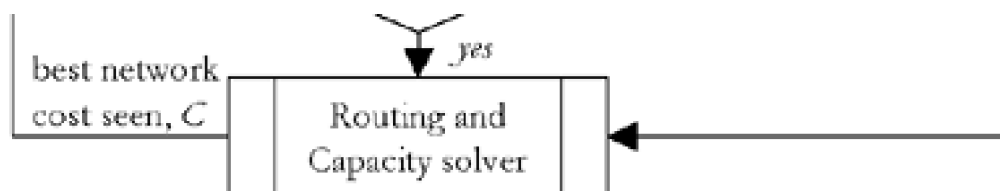


Figure 9-27. The graph sifter with repair process to enumerate graph candidates [Ezem03].





9.13.1 Graph Generation

The output of the graph generator is a binary word of width equal to the number of edge decision variables. Existing edges are represented as hard wired bits (set to 1) in the "counter." Each time the binary counter advances, its' word is processed by the graph sieve. In this role a binary counter has the property that it would enumerate every graph combination, although at sizes of interest this may be impossible in practice. A binary counter also tends to specify graphs with high average weight. If exhaustion of the graph space is not the aim, then a large random sample of edge combinations is more desirable and bit-wise random generation is preferable.

For bit-wise generation each edge is considered individually and set to one with a probability p . For each slot in the edge vector, a random number in the range $[0, 1]$ is generated. If the number is less than or equal to p , the slot is set to 1, otherwise it is set to 0. The function used to determine p leads to the different models for generating random graphs.

In a pure random model p can be set based on plausible maximum nodal degree considerations, so as to directly generate edge vectors with weight biased into the region of interest. If we want graphs of average nodal degree of 5 for a 10-node network (i.e., 10 nodes with 25 edges on average) out of a possible maximum of 45 edges, we would set $p = 25/45 = 0.56$. The fixed probability model does not, however, reflect the structure of real network topologies because any two nodes are as likely as others to become connected.

Other random graph generation methods include Waxman's model [\[ZeCa96\]](#),

Equation 9.46

$$p(u, v) = \alpha \cdot e^{-d/(\beta \cdot L)},$$

where d is the Euclidean distance between nodes u and v , L is the network diameter (maximum Euclidean distance between any two nodes) and $\alpha > 0$ and $\beta < 1$ are model parameters. Increasing α raises the number of edges. Increasing β raises the probability of a long edge. If we set $\beta = (L-d)/L$ we get a pure exponential random graph model where the probability of a diameter-long edge goes to zero. One problem with these random models is that unless biased to high average edge probabilities, they are unlikely to produce a connected graph. For instance at $\alpha = 0.2$ and $\beta = 0.15$, graphs with $d \sim 3.5$ result but are almost never connected. This is one of the reasons for considering algorithmic graph generation methods.

Algorithmic graph generation methods provide a way for us to generate graphs that are at least connected. Some such methods are described below.

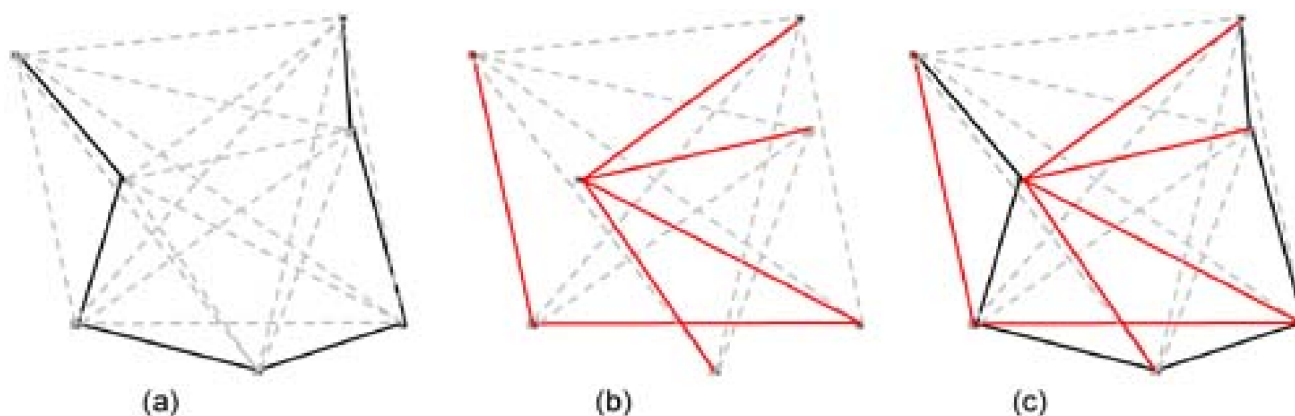
Node Linking

A vertex n is selected at random from all nodes in the graph as the start node and is added to the record of nodes visited. At n , the shortest candidate edge is added to an adjacent node v . Node v becomes the next node to be visited and is added to the record. At v , the shortest candidate edge not yet in the graph is added. This process is continued until the record of nodes visited contains all nodes in the graph, at which time a random connected graph has been produced. A tie occurs if the edge selected from a vertex v has its other vertex n already "linked" in the graph or vertex n has all its candidate adjacent neighbors already linked. To break a tie, we choose another vertex u randomly from the linked vertices such that u has a candidate adjacent node w that is not yet linked. From vertex w , the nodal linking process continues.

Two Trees

This graph generation method is the only graph generation method that directly produces a qualified graph candidate without the need for applying a repair heuristic. It is a variant of the Two Trees Dense graph construction heuristic described in [MoSh89]. The method involves combining two trees formed on graph $G=(V,E)$ to form a graph $G'=(V,E')$. The first tree is a minimum cost spanning tree $T=(V,E')$. The second tree T' is formed on graph $G'=(V,E-E')$. Where all nodes have the same survivability requirements (i.e., biconnectivity or two-node connectivity), we choose the second tree as a minimum cost spanning tree formed on the graph $G'=(V,E-E')$ with no edge of T reused. This method generates two-node connected (but dense) networks and is illustrated in Figure 9-28. In Figure 9-28(a), the solid lines show a minimum spanning tree T for the 7-node, 21-span network. Figure 9-28(b) shows a minimum spanning tree T' for the graph after the edges of T are removed. Figure 9-28(c) shows the resultant graph $G'=(V, T \cup T')$.

Figure 9-28. Example of two-trees method of random generation of biconnected graphs.



To obtain random biconnected graphs using the two trees method, each tree need not be a minimum spanning tree for its subgraph. Any tree-forming procedure, including one based on simple node linking can find a first tree T . Numerous complementary T' can then be found for each T . Note that in most actual transport graphs if we find and remove one tree T , a second tree T' is often not feasible as removal of T would often disconnect the graph. It is important, therefore, to realize that when used in our context the initial graph $G=(V,E)$ is based on the universal set of edges, or, more generally, E at least contains all *candidate* edges for the problem.

9.13.2 Graph Testing

The principle for ordering the sieve tests is to apply the simplest to execute first, so that by the time required in the more complex qualification tests, such as for biconnectivity, is invested only for a small fraction of all graph candidates. When a qualified graph is identified it is passed to a separate process for routing and capacity cost evaluation. We first describe the basic sifting criteria, then comment on the repair strategy for each criterion if it is failed.

Tests for Minimum and Maximum Edge Count

A graph that is either biconnected or two-connected always contains at least N edges. This tells us that the minimum number of edges a qualified graph has should be N . So any edge vector containing fewer than N logic ones is rejected. There is also a maximum edge weight disqualifier. Based on knowledge that no real network (that we know of) has a physical-layer $\bar{d} > 4.6$ or so, we can fairly safely assert, let

us say, $\bar{d}_{max} = 6$. We can be confident of this requirement having no deleterious effect on the solution quality but it sieves out an enormous fraction of graphs. Moreover it is in the nature of the problem that if this limit was having any effect, we would notice solutions coming out around $\bar{d} \approx \bar{d}_{max}$ and could then shift the search window up accordingly. Once a view of \bar{d}_{max} is adopted, we set $S_{max} = \bar{d}_{max} \cdot N/2$ as the maximum number of ones in the edge vector. In an evolutionary planning context the \bar{d}_{max} criterion would be set with the complete network of both old and new portions in mind as a whole.

Tests for Individual Nodal Degree

In a two-connected or biconnected graph every node will individually be of at least degree 2. Otherwise it is a singly-connected stub node and there is no point bothering with the later more complex tests. If all nodes in the graph formed of the existing edge set, any added nodes, and the current variable edge vector have degree 2 or more, the edge vector goes on to the next sifting stage. Otherwise it is discarded and the graph generator is incremented, or repair is effected before continuing.

Test for Edge Cost Above Best Bound

The next simplest test on the candidate edge set is to ask if the sum of the fixed costs of the edges it embodies already exceeds the lowest cost yet seen for a complete design. This test is available after the sifter solver system has evaluated a number of candidate edge sets but it takes advantage of the fact that if the fixed charges of the current edge set exceed the lowest cost of any complete design yet seen, then the candidate vector can be rejected because it cannot possibly be the basis of a lower-cost complete design solution.

Test for a Connected Graph

At this stage we know the candidate edge vector has a reasonable number of edges, each node is individually degree 2 or more, and the sum of fixed edge costs is not above bounds. These tests get rid of many candidates and are extremely fast as they involve nothing more than counting and comparison tests. But we do not yet know if the edge vector is describing a connected graph in its own right (for green fields use) or describes a connected graph in conjunction with the existing network (for evolutionary planning). At this point we are forced to start applying more complex disqualification tests and one could ask why not apply the test for two-connectedness or biconnectivity (as desired) right at this stage. The test for simple connectivity is still quite fast, however, and rejects many candidates. In addition, the subsequent tests for biconnectivity or two-connectedness are based on algorithms that look for bridge spans or nodes and therefore requires a confirmation of basic connectivity before they can be properly applied. A rapid procedure to test if the graph G is a single connected component is to start at any node as a root node, and step out to each of its adjacent nodes called its neighbors. Each span traversed to reach a neighbor is marked. From each neighbor node, its own neighbors are visited also, but only unmarked edges are traversed. A record is kept of all nodes visited. When all connected nodes have been visited, the visited node record is checked to ensure that it contains all nodes in the graph. If it does, G has passed the test and goes on to be tested for two-connectivity. Otherwise it is discarded.

Only one of the final two tests need be implemented, depending on whether the design criterion is for two-connectivity (may contain a bridge node) or biconnectivity (excludes bridge nodes). Each of the following tests can be implemented with space and time complexity of $O(n + m)$ where n = number of nodes and m = number of edges, using the algorithm for that purpose from [BaGe00](#) which was reviewed in detail in [Section 4.8](#).

Test for Two-connectivity or Biconnectivity

For a connected graph G , its edge connectivity $I(G)$ is the size of the smallest cutset in G . In general G is k -edge connected if $I(G) \geq k$. A graph that is not two-edge connected contains a cutset with only one edge, called a bridge. This test checks the graph G for the existence of a bridge edge. If none exists, G has passed the test and is released as a qualified graph on which to solve the routing and capacity design problem.

The concepts to test for biconnectivity are analogous, but using nodes instead of edges. If G is connected and not a complete graph, its (node) connectivity $K(G)$ is the size of the smallest separating set in G . A separating set in G is a set of nodes whose removal disconnects G . If the separating set contains only one node, the node is called a cut vertex (or bridge node). We then say that G is k -connected if $K(G) \geq k$. A biconnected graph is one that is connected and contains no cut vertex. A graph that is not two-connected contains a separating set with only one node, the cut vertex. This test checks the graph G for the existence of a cut vertex. The number of bi-components in G signifies the presence or absence of a cut vertex. A bicomponent is defined as a maximally connected subgraph of G . If G has more than one bicomponent, a cut vertex exists and G is discarded. Otherwise, G represents a valid graph that can be passed on to the final solver for further optimization of routing and capacity.

9.13.3 Repair Procedures

If sifting with repair is implemented, the repair at each stage only corrects the "local" deficiency being tested for, and does not lead directly to a fully qualified graph. For instance if a stub node is found, an edge is added to correct this deficiency, even though other flaws may remain to be found by further criteria in the sifter. The repair heuristics are simple specific routines that deal with each particular graph property. For example, if a graph fails the test for maximum or minimum span count, the routine to repair the graph would be to add or remove edges to/from the graph respectively, until the weight of the graph is within the desirable range. If the graph fails the stub node test we add an adjacent edge to each such node connecting it to a new nearby node. For the tests of connectivity and so on, the repair routine first finds the components in the graph. Then an edge is added between nodes in different components. The respective test is repeated and new components (if any) are identified. Edges are added between nodes in different components until the graph is connected, two-edge or two-node connected as desired. For the best bound test on edge costs, the repair routine exchanges high cost edges with lower cost edges until the graph cost is at least below the total cost of the best complete design yet seen.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

9.14 A Tabu Search Extension of the Graph Sifter Architecture

Another tactic that can be applied within the graph-sifter framework draws from ideas of "tabu search" ([Section 4.18.3](#)) to accelerate the process of eliminating candidate edge combinations. This is suggested by the block entitled edge decision frequency-quality metrics in [Figure 9-26](#) and the outer dashed loop suggesting how certain edges might gradually be pinned to either 1 or 0 as an increasing number of complete solutions are observed. The basic principle is that in many combinatorial optimization problems, certain elements or choices are frequently associated with being part of a good overall design. Here, the idea would be to keep a running count of how frequently each edge appears in low cost designs and how often a high design cost correlates with its absence. A frequency quality metric for this can be of the general form:

Equation 9.47

$$M_j(k) = \sum_{k \in \text{last } K \text{ designs completed}} \delta_j(k) \cdot [C - (\text{design cost})] + (1 - \delta_j(k)) \cdot [\text{design cost} - C]$$

where $d_j(k)$ is the 1/0 decision outcome regarding edge j in the k^{th} complete design and C is just an arbitrary constant, characteristic of a typical design cost. $d_j(k)$ values apply in evaluating $M_j(k)$ whether edge j is currently a free variable or if, as a result of the strategy below, it has been forced to 1 or 0. In [Equation 9.47](#) the merit of span j , $M_j(k)$, is updated on every completed design. Either the edge was used and it was part of a good design, in which case the first term weights it with the cost savings of that design relative to the bias constant C , and vice-versa. Conversely, a judgment is made about the quality of the designs in which edge j is *not* employed, in the second term. If there is no correlation either way with overall design quality and the employment of edge j in the design, $M_j(k)$ will bobble around close to C . On the other hand if edge j is usually a good or a bad decision $M_j(k)$ will drift up or down, away from C , respectively. Thus, a TS-like policy could be implemented to structure the search as follows:

Equation 9.48

$$\begin{array}{ll} M_j(k) > 2 \cdot C \rightarrow \delta_j(k+1) = 1 & \text{i.e., assert edge } j \\ -2 \cdot C < M_j(k) < 2 \cdot C \rightarrow \delta_j(k+1) \in (0, 1) & \text{i.e., edge } j \text{ is a decision variable} \\ M_j(k) < -2 \cdot C \rightarrow \delta_j(k+1) = 0 & \text{i.e., exclude edge } j \end{array}$$

Under this policy the complete edge decision search space would be reduced and the search guided when there is no accumulating evidence or experience of particular edges being either effective or not, they remain as free edges in the enumeration of graph candidates. But some edges build a history and get firmly asserted or deleted from the graph candidates inspected in the ongoing search. Note that the $M_j(k)$ values continue to evolve even after an edge crosses the threshold where it is forced in or out of the graph generation, so a given edge may come in and be firmly held through a certain range of the enumeration space, then dropped out again later. The thresholds of $\pm 2C$ for forced inclusion or exclusion of an edge in [Equation 9.48](#) is an arbitrary example of how the thresholds could be set. More formally the thresholds should be set based on observing some criterion like the 1-s cost offset from the mean. Note that while the basic sifter architecture, run to exhaustion of the free edge counting range, can claim optimality in its result, addition of the outer loop of frequency-quality metrics and edge assertion thresholds, converts the entire procedure into a form of tabu search algorithm which may find good solutions more quickly than the exhaustive sifter, but will no longer be an optimal solution algorithm.

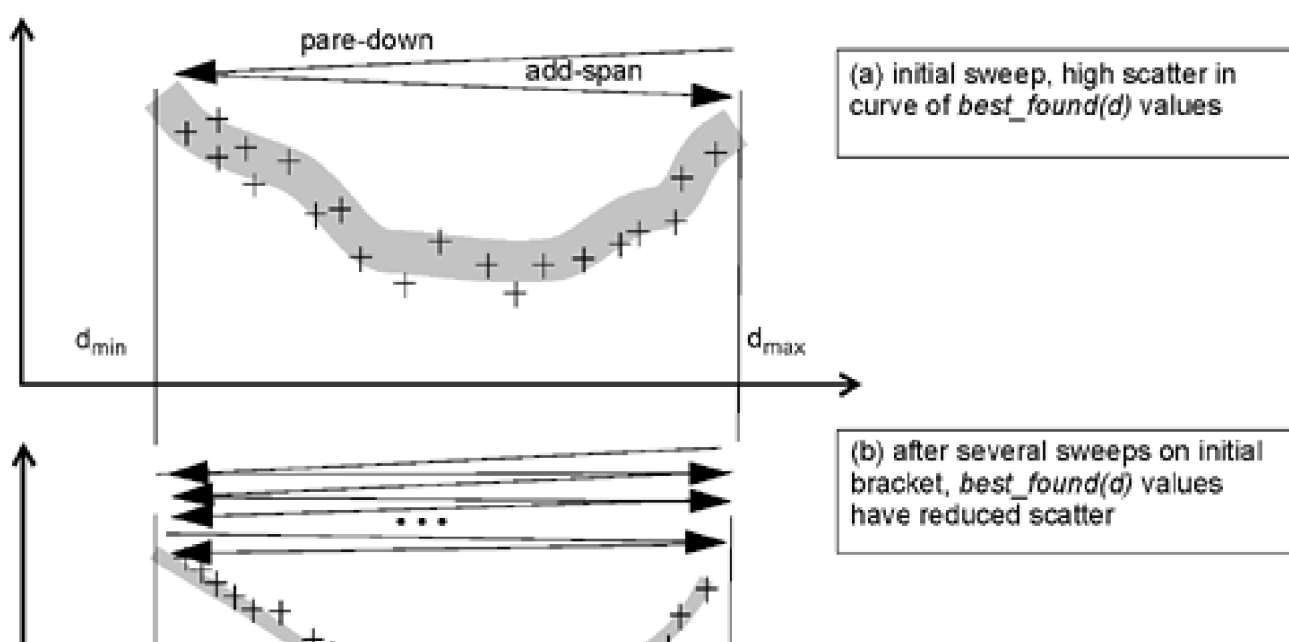
9.15 Range Sweeping Topology Search

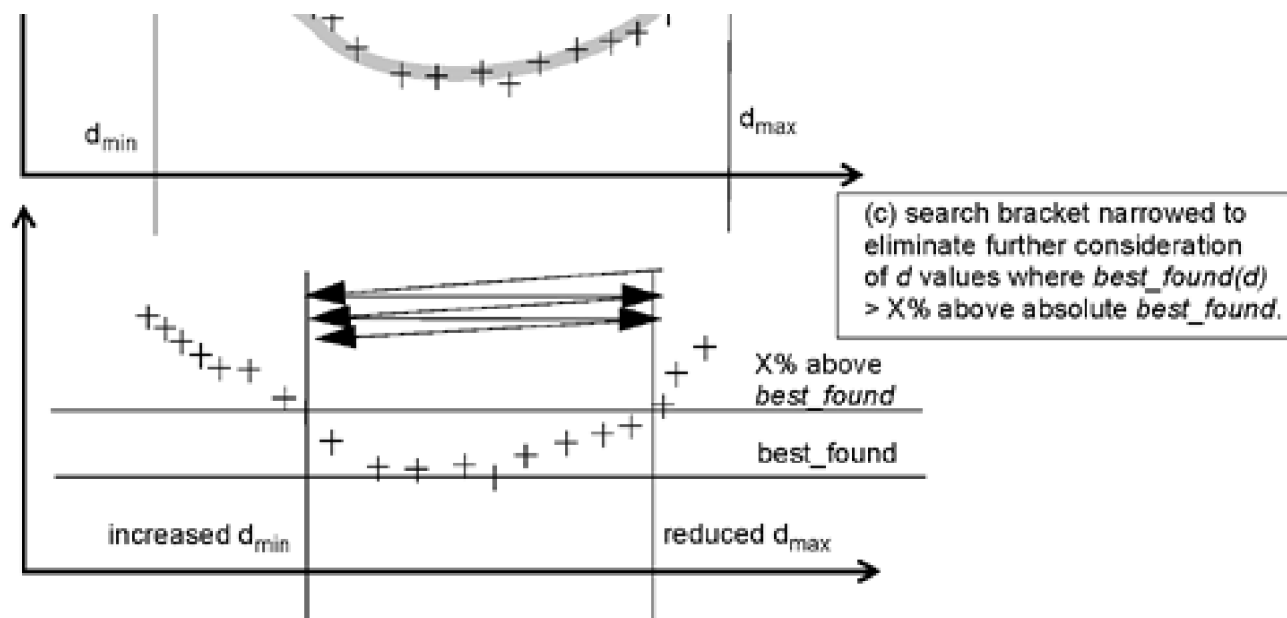
We close the chapter by describing a technique for searching systematically (rather than randomly) through a plausible region of nodal degree for a detailed topology construction that yields minimum MTRS cost. Using the nodal degree and cost information derived from the ongoing search, we adaptively narrow the search bracket to search even more closely over the region within which we believe an optimal graph topology may lie. Called "Sweep Search," the idea exploits the fact that we know from first principles and preliminary studies that there is a nearly continuous curve of total costs versus connectivity with a unique minimum basin seen in curves such as in [Figure 9-6](#). If we presume bounds on nodal degree d_{min} to d_{max} then sweep search can be used to traverse this region, for as many passes as desired, enumerating detailed graph constructions within the plausible region.

The procedure has two main phases. One phase is the *pare-down* procedure already outlined in [Section 9.4](#). In the previous *pare-down* the edge with the least working flow crossing it is removed at each iteration, if its removal does not violate biconnectivity. Here, however, random selection of an edge for removal is a preferred means to explore different graphs on each sweep. The process is repeated until no edge can be removed from the graph without violating biconnectivity constraints or the average nodal degree of the graph is d_{min} . The process reverses here and starts over with edge additions replacing edge removals. To the sparse graph we now have, we add an edge, and pass the graph to the solver for routing and capacity design. We continue to do this until the graph has an average nodal degree of d_{max} . Thus, we never lose biconnectivity throughout the search but are always evolving the actual graphs being considered, walking them back and forth through the region where the optimum is thought to lie. The test for biconnectivity uses the $O(m+n)$ algorithm of [Section 4.8](#).

When we walk from a highly connected graph to a sparse graph, we call it "pare-down." Similarly, walking from a sparse graph and adding spans is to "pare-up." A pare-down plus a pare-up sequence is called a "sweep." For each sweep, the nodal degree d of the graph that produces the lowest edge plus capacity cost and its cost are noted. The search sweeps back and forth through the "plausible region" as long as desired, constructing different detailed topologies and evaluating their total design cost. The overall search can be progressively narrowed by adding an observer process that records the *local_best_found* solution at each d value in the sweep range and using this information to periodically narrow the d_{min} to d_{max} search bracket. The idea is to observe the minimums of total design cost achieved at each d value in the sweep range and pull in and re-center the actual search bounds as enough *local_best_found*(d) data is observed to produce sufficient monotonicity and contrast between the absolute *best_found* at the extreme values of the search ranged. [Figure 9-29](#) illustrates the overall idea.

Figure 9-29. Concept of oscillating sweep search strategy for detailed topology construction with adaptive narrowing of the search bracket.





In the process of pare-up and pare-down, edges may be added and removed randomly or with deterministic criterion. The criteria we use for pare-down include moving the edge with the least amount of total capacity placed, removing the edge with the least amount of spare capacity placed, removing the edge with the least amount of working capacity placed, and removing the most expensive (longest) edge in the network. And for pare-up: adding the shortest candidate edge not yet in the network.

We also intersperse the deterministic edge selection criteria with a random selection of edge for addition or removal to avoid being stuck in local minimum. Branch exchange routines are also included in between sweeps to provide a toggling or mutation of the solution. The implementation is such that after each sweep (*pare_down* and *pare_up*), there is a random number between a minimum of 1 and maximum of 3 edge exchanges before the next sweep. The nodal degree of the graph remains unchanged. An alternative to branch exchanges at the turning around point in each sweep is to call the graph sifter for a whole new qualified random graph as a starting point for the next *pare_up* or *pare_down* pass.

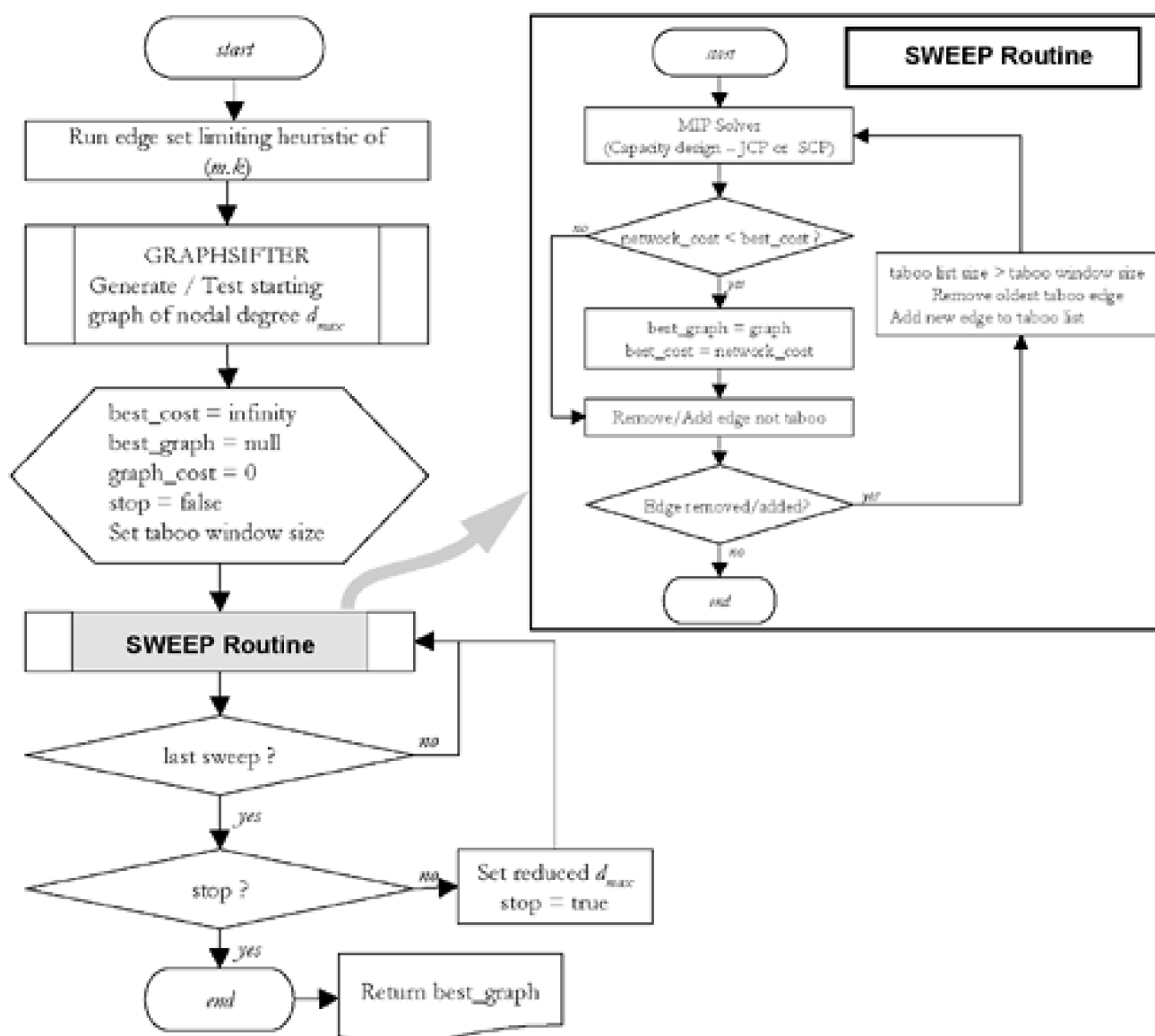
In experimenting with the sweep search method we have also found it helpful to employ a tabu mechanism such that an edge, once added (or removed), cannot be removed (or added) until a certain number of further edge operations are completed. This, in addition to the change of graph or branch exchanges at the end of each sweep direction, makes it unlikely that we ever consider the same graph twice. The pseudocode is as follows:

Sweep_Search Tabu Logic

```
windowSize = x;
tabuList addedEdges, removedEdges;
pareDown:
select edge for removal from graph;
if edge is not tabu (included in addedEdges)
    remove edge from graph;
    add edge to removedEdges
    if (size of removedEdges = window Size)
        remove oldest edge from removedEdges.
pareUp:
select edge for addition to graph;
if edge is not tabu (included in removedEdges)
    add edge to graph
    add edge to addedEdges
    if (size of addedEdges = window Size)
        remove oldest edge from addedEdges.
```

The stopping criterion is a preset time limit or number of full sweeps completed. [Figure 9-30](#) shows the composite sweep search strategy with inset detail of the sweep routine.

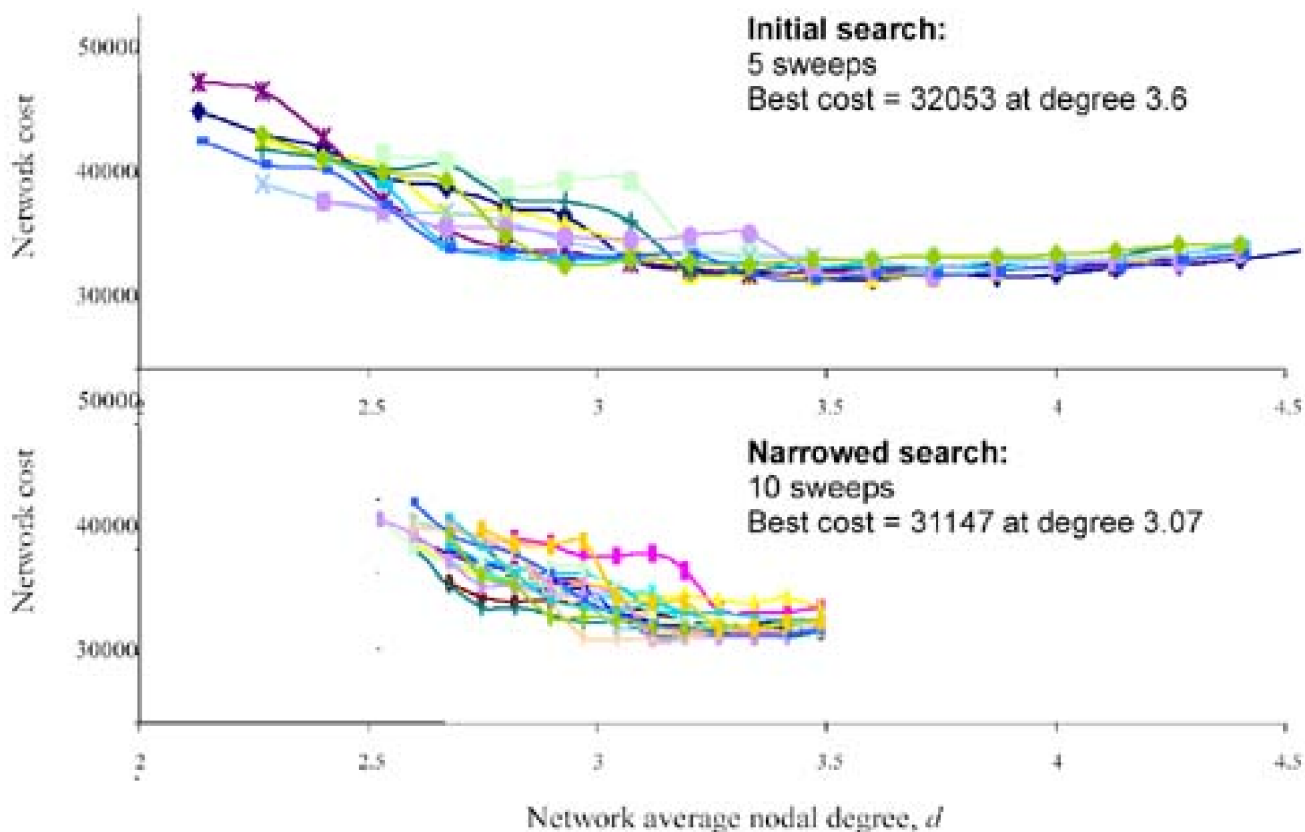
Figure 9-30. Overall Sweep-Search procedure and the core Sweep routine.



9.15.1 Sample Results with Sweep Search

As a test case we present the 15n56s network used previously in the 3-step heuristic and edge-reduced MTRS trials. Its full complement of 56 candidate edges are first reduced to 34 by the edge-limiting heuristic giving an initial $d_{max}= 4.53$ and initially we set $d_{min} = 2$. An initial set of 5 sweeps yielding a best-cost graph in the vicinity of $d=3.6$ and identified the region from $2.5 < d < 3.5$ for narrowed searching. Ten sweeps were allowed in this region yielding an improved best-cost of 31147, which is within 1.3% of the best known solution for this problem found by a 4-hour time-limited run of MTRS on a 4-processor machine. The total time for the sweep search result was 3.6 hours, but on a desktop PC. [Figure 9-31](#) illustrates the actual sweeping trajectory.

Figure 9-31. Sweep-Search trajectory on 15 node network with 34 candidate edges [Ezem03].

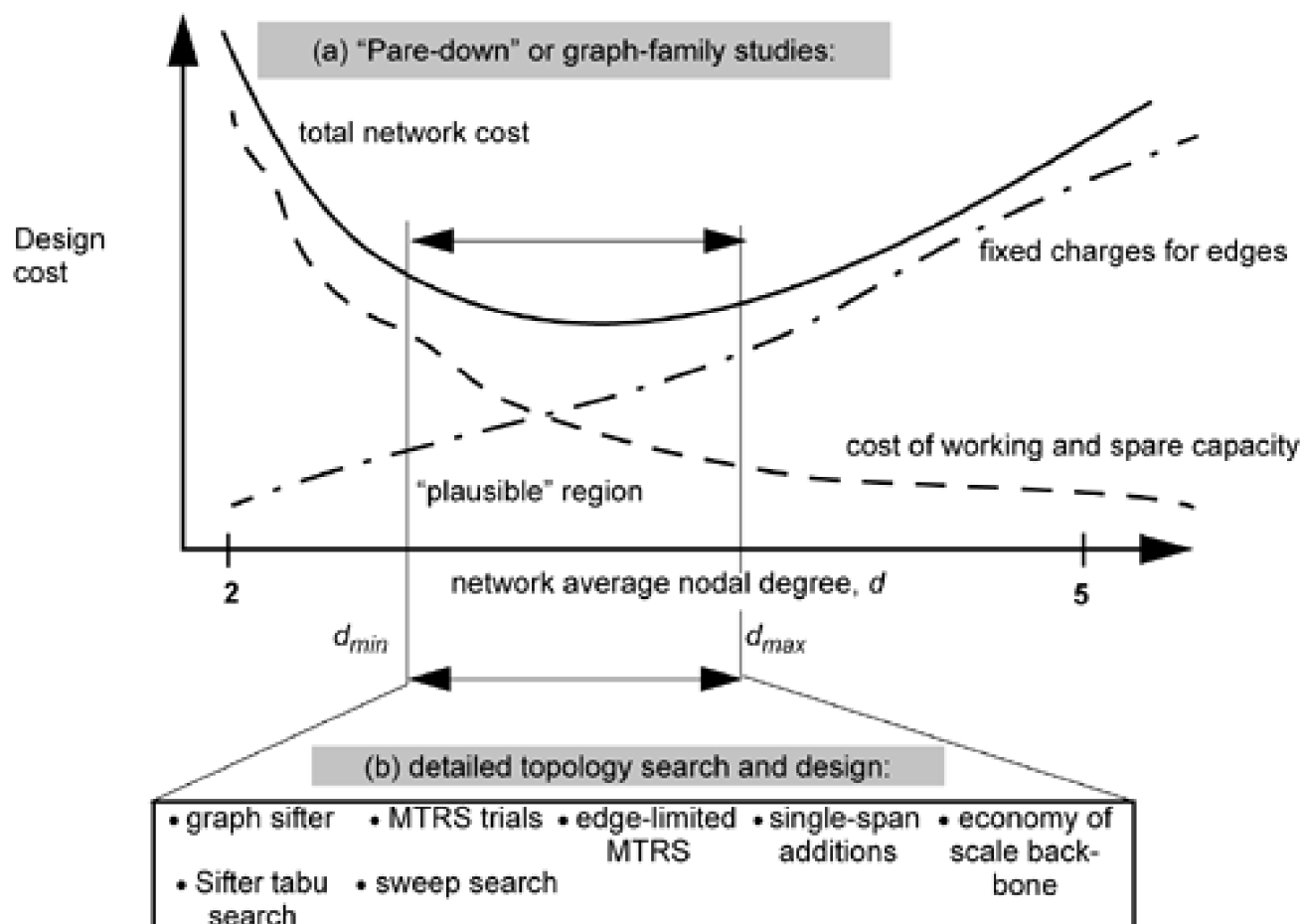


While the performance of time-limited MTRS and sweep search were not greatly different in this case, sweep-search clearly scales much better than MTRS, or the 3-step heuristic (or any MIP-based method) as the number of candidate edges increases. While the MIP solution time grows exponentially with $|E|$, the main "outer" loop of sweep search has no dependency at all on the number of candidate edges. It merely keeps walking up and down through biconnected graph space in the stipulated region as long as requested. The "inner loop" process of evaluating capacity cost is where sweep search has a dependency on the number of *nodes* in the problem, although again, no direct dependency on the number of candidate edges. The dependency on number of nodes arises from using a MIP solver or a routing and spare capacity assignment heuristic to evaluate capacity costs of each graph that the sweep process visits. Each such graph is presented as a "given topology." Routing and SCA times increase with the number of nodes and spans in each such "given topologies" from the outer loop, but in practice can still be solved quickly (minutes) for as many as 60 to 100 nodes. Being procedurally-based, as opposed to an AMPL MIP model, sweep search also offers the greatest general scope for internal enhancements in coding for speed and further enhancements in search strategy.

9.16 Overall Strategy and Applications for Topology Planning

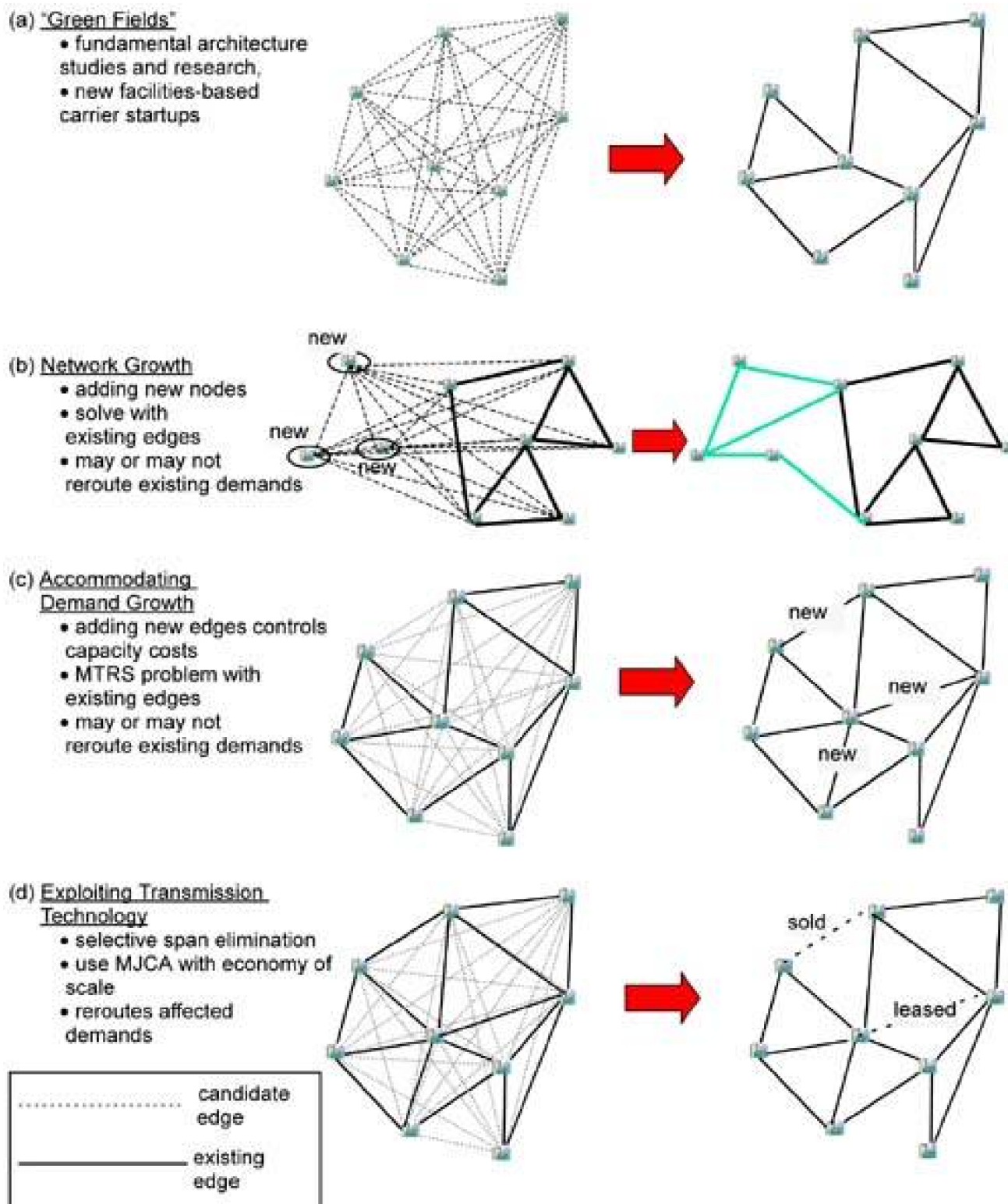
This chapter has presented some basic insights and a range of tools and ideas to use in approaching topology-related questions and topology-oriented network planning. What most distinguishes topology design for mesh-based survivable networks from prior methods is the requirement of biconnectivity and the presence of spare capacity and shared protection routing considerations at the same time as dedicated capacity and routing considerations also apply for working flows. While working flows are generally still well-treated by shortest path routing, what is notably different about spare capacity in terms of topology design is that the spare capacity wants to be situated *away from* the working capacity that it protects. As a distillation of many considerations in this chapter, [Figure 9-32](#) suggests an overall two-phase approach to topology planning for mesh networks at larger sizes where tools like MTRS or the 3-step heuristic cannot be directly applied. Initial studies can illuminate where an existing network is roughly situated relative to the optimum trade-off between edges and capacity or, given a demand pattern, capacity and edge costs a few sweeps with *pare_down/pare_up* can give a general indication of the region where an optimum topology appears to lie, typically often in the $2.3 < \bar{d} < 3.5$. With a bracket established for the range of plausible solutions, this allows a d_{max} to be set for the edge-limiting heuristic reducing the candidate edge set. At that stage any of the more detailed solution strategies of this chapter can be applied to identify a specific graph construction (and routing and capacity placement plan) that is near-optimal. In practice, having a repertory of techniques through which to approach the problem can be valuable to provide various planning insights and options for further consideration in aspects that the topology solvers themselves cannot encompass, such as related considerations of business strategy, financing, regulatory issues, and so on.

Figure 9-32. Overall strategy for studies to optimize topology: (a) determine the right characteristic nodal degree, (b) apply detailed topology search / construction methods.



We should stress in closing that topology is not a problem that only needs to be solved once for a given network. On the contrary, topology assessment and evolution planning should be an ongoing process. [Figure 9-33](#) illustrates the four main roles that we see for ongoing use of the topology-related capabilities in this chapter.

Figure 9-33. Four main uses for topology design methods of this chapter in network planning.



First is the "green fields" context that we have mainly considered. This is the most natural context for development of basic insights and techniques through research, and is also relevant to planners in asking fundamental questions about future architecture or how existing topology and network cost compares to the ideal "green fields" topology. And now and then in the real world, there are actually green fields design situations with new operator start-ups that could use these methods almost directly.

Another context is adding new nodes to the network. This can be addressed by any or all of the methods developed on the green fields problem with the assertion of existing edges. The problem complexity is almost unaffected by existing edges and because the number of edge candidates to consider in adding one or a few nodes at a time is relatively small, many node addition problems could be directly and almost perfectly solved with MTRS. Depending on operator policy, such network extension planning may allow rerouting of existing demands in conjunction with the new edge addition decisions. An area for further study would be to consider the topology extension problem with a number of candidate nodes to add (based on revenue forecasts) as well as candidate edges to weave the chosen new node(s) into the existing network fabric.

In [Figure 9-33\(c\)](#) we note that topology planning is also required to monitor or adjust the balance between edge and capacity costs if the carrier undergoes a period of significant demand growth. We saw that multipliers on demand were equivalent to a reduction in Ω and might therefore call for new span additions to keep the network near its minimum cost realization.

On the other hand, technology advances may outstrip demand growth over a period of time in terms of the capacity versus cost characteristics of available transmission systems. Such technology advances can effectively reduce Ω suggesting a sparser topology. However, in an existing network the "edge costs" may typically already have been incurred. If they are sunk costs, the MTRS orientation is not so directly applicable because the opportunity to not spend on the edges in question has already past. The more future-looking opportunity in this context is to view the existing graph as one from which certain spans might be "eliminated" (possibly sold or leased off) under a strategy of changed routing and protection arrangements that take advantage of large transmission module capacities and economy of scale in purchasing such capacity to eliminate certain network spans. MJCA (and extensions to include prospective sale or lease costs of existing spans) is the tool for this kind of topology-related planning study.

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)

Chapter 10. p -Cycles

p —Cycles are remarkable and interesting because they combine the real-time switching simplicity and speed of rings with mesh-like efficiency and the flexibility and freedom of a mesh in the routing of working paths. Up until the discovery and understanding of p -cycles about 1998, if you had asked the author or many other researchers and transport planners if a protection scheme was even theoretically possible that would combine the properties of ring "50 ms" switching and mesh redundancy in the 30 to 50% range you would likely have received the considered reply that "No, it seems that 100% redundancy is the inherent "price" one has to pay to support 50 ms switching. Only rings, ring covers, or 1+1 APS can support that type of fully preconfigured switching, and they are all at least 100% redundant." But in fact with p -cycles it *is* possible to have these properties together at once! This fortuitous combination of features i.e., "ring-speed with mesh-efficiency" in one scheme lay undiscovered through more than a decade of the ring *versus* mesh debate. It seems especially pleasing that out of that environment of pitched contention and competition something should emerge that has only the *best* features of each of these seemingly irreconcilable prior alternatives.

p -Cycles are easily misunderstood to be a form of cycle cover and yet it is fundamental to the efficiency of p -cycle networks that they are *not* simply a form of cycle cover. We see in fact that they bear the $1/(p-1)$ mesh-like signature for the lower limit on achievable redundancy, reaching levels as low as 36% redundancy in examples in the chapter. The key difference with p -cycles is the concept of *straddling spans*. These are spans which themselves are not part of the p -cycle but they can bear *two* working channels for each protection channel on the p -cycle which they straddle. And straddling spans themselves bear no protection capacity. In addition p -cycles permit working paths to use the entire mesh-like facilities graph for shortest path routing, unlike rings in which working paths are ring-constrained. In contrast, any form of cycle cover or ring design is, at best, 100% redundant.

What is fascinating is that in one sense p -cycles are based on such a minor technical variation on rings. Indeed, technically p -cycles are not difficult to develop as an extension to ring technology. An easy mistake would have been to rely on initial intuition and dismiss the change as having only minor potential effects and not pursue the idea further. But when the effects of this "one small change" are fully worked through, it turns out to have sweeping implications. This chapter is filled with material about p -cycles and an initial pursuit of those implications. The investigation of networking ideas, design methods and heuristics related to p -cycles is, however, still relatively young. Much more networking science on p -cycles is expected to follow. The main aims of this chapter are to introduce the concept, to make clear its differences from prior methods, to show how and why such high efficiency arises, and to treat a variety of further p -cycle concepts such as Hamiltonian p -cycles, node-encircling p -cycles, and straddling subnetworks. We also consider the basic design problems with both ILP and heuristic algorithmic approaches. In doing so it is important to realize that the p -cycle concept is a generic logical scheme that can be implemented at the fiber level, the WDM or OC- n channel level, or at the MPLS/ATM layer. In some cases we need to adopt one or the other framework, often the DWDM layer, as a discussion vehicle, but the inherent applicability to other layers or contexts should not be overlooked.

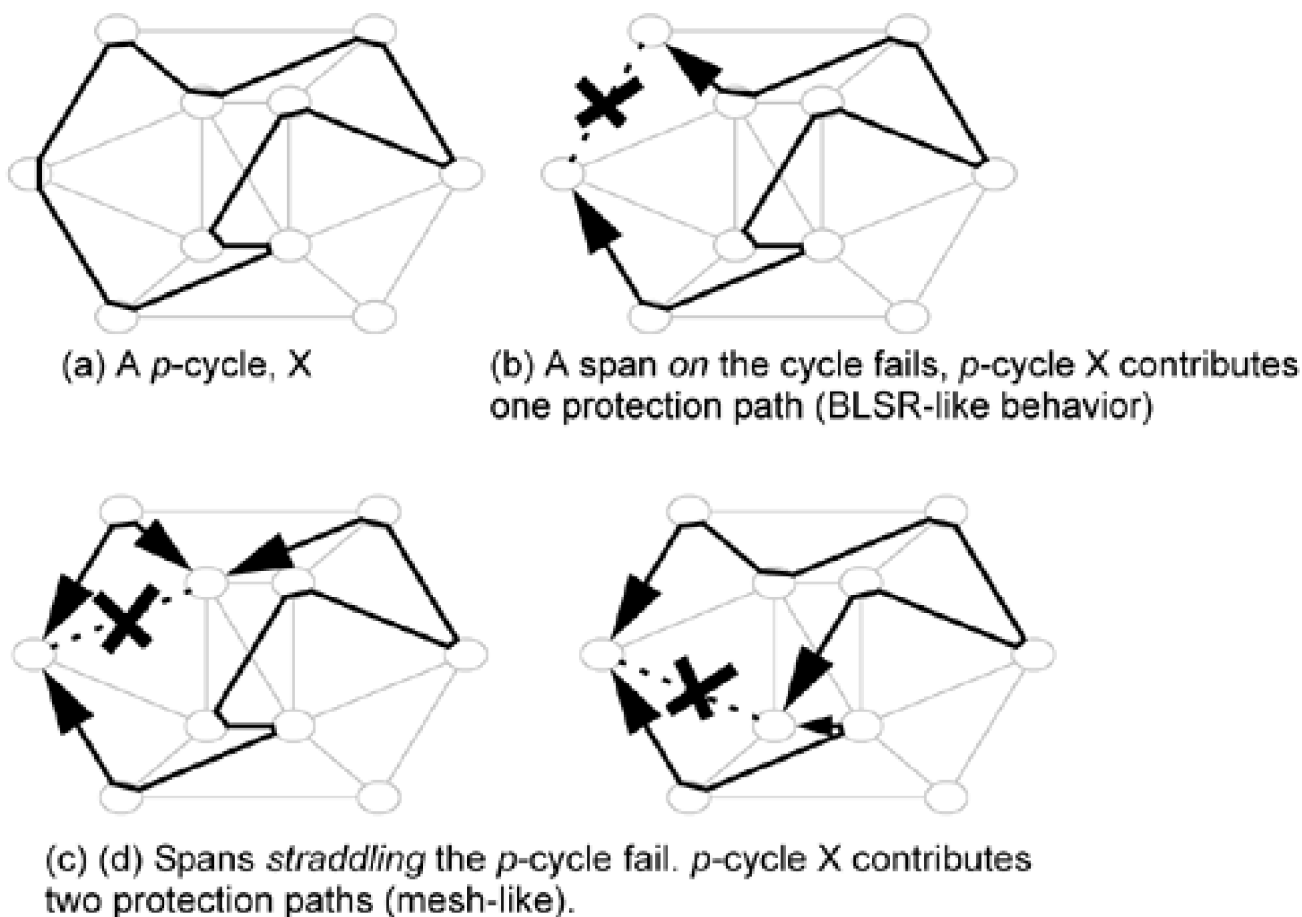
10.1 The Concept of p -Cycles

In [Chapter 3 \(Section 3.5.4\)](#) we gave a brief overview of p -cycles in the context of an initial survey of known survivability schemes. Let us now develop that description further, setting the common technical basis for further treatments in the chapter.

In protecting a network with p -cycles, one forms cyclic preconnected closed paths of spare capacity while allowing working paths to take the shortest direct route over the facilities graph. p -Cycles are formed in advance of any failure and the switching actions required in real-time are completely preplanned and essentially the same as that of a line-switched ring. Despite similarity to rings in that both use a cycle on the graph for their topology, p -cycles are unlike rings or cycle covers^[1] in that they protect both *non-cycle* and *straddling* failures, illustrated in [Figure 10-1](#). [Figure 10-1\(a\)](#) shows an example of a preconfigured protection cycle (a " p -cycle"). In (b), a span on the cycle breaks and the surviving arc of the cycle is used for restoration. This action is functionally like a unit capacity BLSR. In (c) and (d), however, the same p -cycle is accessed to support restoration of working paths that are *not* on the cycle. In fact, cases (c) and (d) are the more advantageous circumstances in general because *two* restoration paths are available from the p -cycle for such failures. Any conventional ring provides at most one restoration path per unit of protection capacity and protects only against failures on the spans of the same ring, not on "straddling" spans.

^[1] As a group rings and cycle covers includes 1+1 APS, BLSR or OSPR, UPSR or OPBR, FDDI rings, resilient packet rings (RPR), oriented cycle double covers (O-CDCs), and cross-connect managed logical rings.

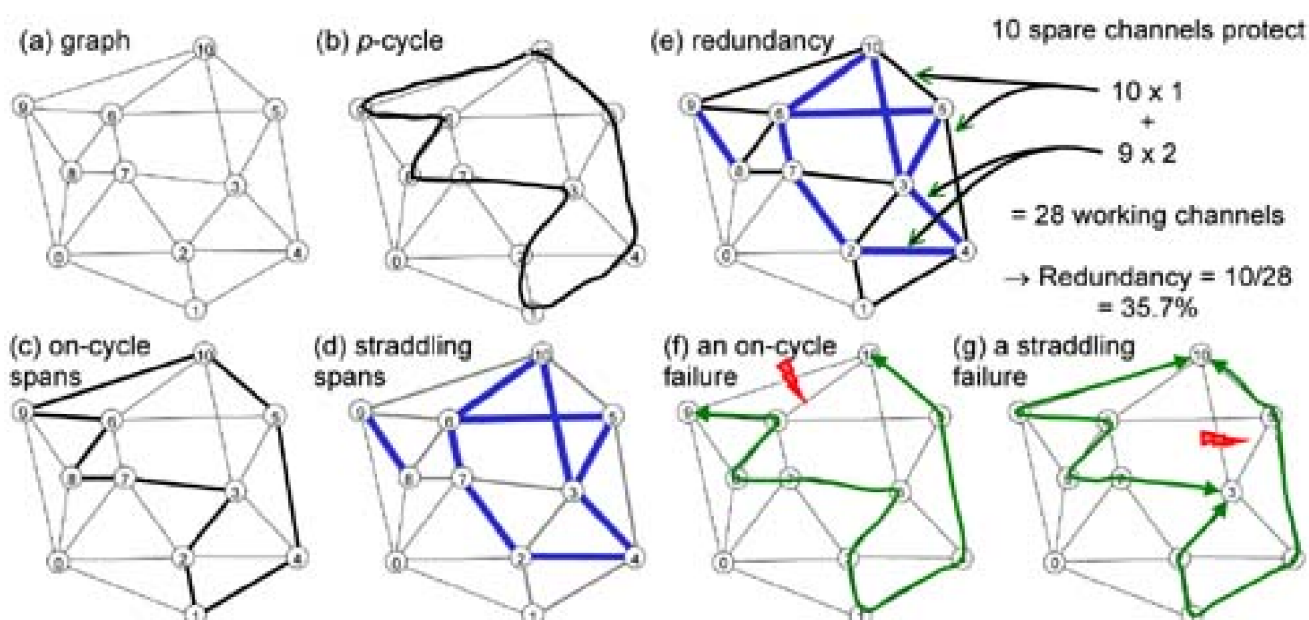
Figure 10-1. Use of p -cycles in span protection.



10.1.1 Why Straddling Spans Are So Significant

Straddling spans make a significant difference to the failure coverage provided by the same investment in spare capacity in a ring compared to a p -cycle. Consider as an example the network in [Figure 10-2\(a\)](#) of 11 nodes and 24 spans. In [Figure 10-2\(b\)](#) a cycle of 10 hops is configured as an example. If this cycle is used as a BLSR then each unit of protection capacity on the ring protects the same amount of working capacity on the 10 spans underlying the ring itself, shown in [Figure 10-2\(c\)](#). These are the so-called "on-cycle" failure spans. But if we use the same unit capacity cycle in (b) as a p -cycle we also access it for protection of spans such as (6-10) which is said to straddle the p -cycle. A straddling span has both its end nodes on the p -cycle, but is not itself part of the cycle. In the example, the p -cycle has nine such straddling span relationships, shown in [Figure 10-2\(d\)](#).

Figure 10-2. Example comparing the fault coverage and redundancy of using the same protection capacity as a ring, but in the form of a p -cycle.



Straddling span relationships have *twice* the leverage of an on-cycle span in terms of efficiency because when they fail, the cycle itself remains intact and can thereby offer two protection paths for each of unit of protection capacity. Note, in the case of a span such as (9-8), that the straddling spans are not limited to being only "inside" the p -cycle perimeter on the graph. [Figure 10-2\(e\)](#) shows the resultant impact on efficiency, or conversely the redundancy. The example p -cycle can achieve 36% redundancy if each of its straddling spans has two units of working capacity equipped. The corresponding ring is 100% redundant. In [Figure 10-2\(e\)](#) and (f) the real-time operation is illustrated. In (e) we see a BLSR-like protection reaction to an on-cycle failure. In (f) the p -cycle is accessed to protect a straddling span failure. Note the significant increase in protection coverage provided by using the same investment in spare capacity as a ring, but simply accessing it as a p -cycle. The single p -cycle of 10 hops provides protection to 19 spans and because it protects two times its own capacity on each straddling span it actually covers 28 units of working capacity, compared to the ring which protects only 10. Without any particular effort this simple example thus yields a protection structure of $10/28 = 35.7\%$ (logical) redundancy.

Note also a "signature" feature of p -cycles: that straddling spans have working capacity but *no* spare capacity. Spans (3-4), (2-4), and all other straddling spans in [Figure 10-2](#) may bear either one or two units of working capacity, but in either case require *zero* units of protection capacity. In a complete design of many p -cycles such spans may in practice bear protection capacity associated with other p -cycles overlying them, but the very possibility of spans with *no* protection capacity is a unique property of p -cycle based networks. In fact this gives a simple "acid test" to determine, regardless of other complexities, whether a given scheme is or is not equivalent to p -cycles. No form of enhanced ring, cycle cover, or generalized loopback design exhibits such "working only" spans—spans which bear zero protection capacity.

10.1.2 Historical Origins: Preconfiguration and the "Clamshell" Diagram

As a digression, readers may find a synopsis of how we came to discover and understand p -cycles interesting, as well as containing some useful lessons. Philosophically, it is an interesting case in point about how one seemingly minor technical difference can turn out to have sweeping implications when followed through. In this case, it would have been all too easy to pre-judge the implications of "just adding straddling span failures to rings" as being quite minor, and dismiss the idea. But this "one small difference" turns out to have major networking efficiency implications. Fortunately, we did not have the chance to mistakenly dismiss the idea of adding straddling spans early on, because that was never directly proposed. Instead, the efficiency of what we now call p -cycles was first encountered—and initially suspected to be a mistake—in the course of thesis research by Demetrios Stamatelakis. Stamatelakis was continuing a general line of investigation on the idea of spare capacity preconnection at the cross-connects of a mesh-restorable network [GrMa94] [MaGr97] [Stam97] [StGr99]. The problem we were tackling was that it was—by then—recognized that restoration path-sets in a mesh network could in fact be spontaneously and efficiently formed (self-organized actually) by a DRA in well under 2 seconds. While it took years to convince a critical mass of industry planners that this was itself possible with methods such as the SHN, and this was verified by others, the next problem was that key players in the industry then objected that distributed mesh restoration would *still* be too slow because the cross-connects (then SONET-era DCS) were slow in *putting the cross-connections into place*. If so, the situation would be one of every node knowing rapidly *what* to do, but taking too long to do it—yet another obstacle for the mesh alternative to overcome.^[2] So the hypothesis of preconfiguration was that some patterns of prefailure cross-connection of the spare channels in a mesh network could put the network in a state of maximal "readiness," where some or most of the cross-connections required in real time would be already made when needed upon failure.

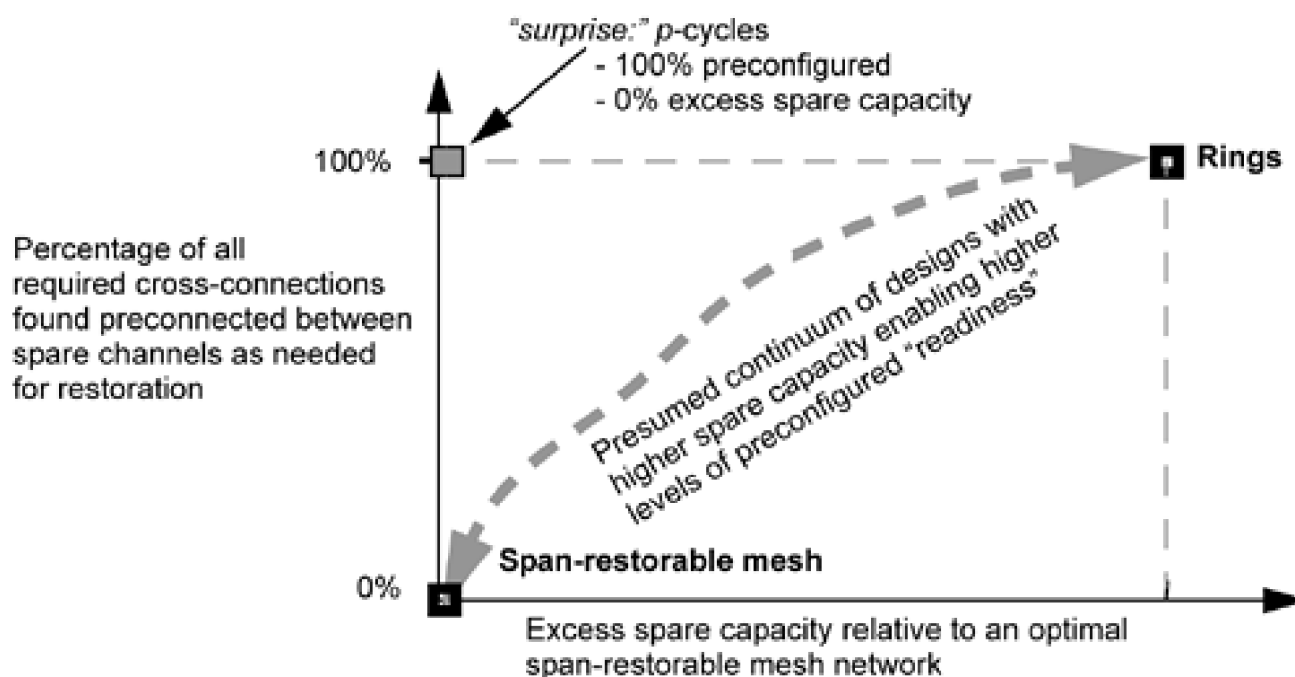
^[2] The perception that cross-connects would be too slow in making cross-connections for restoration stemmed from a Bellcore specification that *allowed up to 1 second* per cross-connect action for remote provisioning applications. In an era when DCS machines were even called "slow switches," this was widely interpreted as saying that DCS nodes *required 1 second* per cross-connection. Indeed some SONET DCS systems were quite slow. But even at the time, DCS vendors interested in the restoration application were able to implement 10 ms "internally commanded" cross-connections. The whole issue was technically a red-herring—just another roadblock against the mesh alternative. Ironically, however, it forced our thinking on toward preconnection to mitigate the issue, and this led to p -cycles. Today electronic and optical cross-connects implement millisecond or faster internal switching functions for restoration or even mass parallel matrix changes. For more on the concept of a "prompt internal" cross-connection requirement, see the discussion in [Gro94].

Readiness meant that once the required pattern of restoration paths was known, the real-time workload of actually making the required cross-connections would be reduced by certain patterns of prefailure connections already made between spare channels. In the general idea of preconnection, the patterns formed in the spare capacity could be arbitrary patterns or restricted to linear segments, trees, cycles, or mixtures of any of these. (Trees are still being investigated in some groups as preconfigured protection structures. They may have natural advantages for protecting multicast sessions, but as we came to understand in our preconfiguration studies, they are much less efficient than cycles as a class of preconfigured protection structure [StGr00b].) The initial paradigm of the research was to investigate different strategies of preconnection and study the trade-off—which we presumed to exist—between the total amount of spare capacity required and the percentage of prefailure "readiness" that could be achieved.

Rings, at 100% redundancy and 100% preconfiguration, were conceptually recognized as anchoring one end of the trade-off curve. An efficient span-restorable mesh where all spare channels are held in an unconnected pre-failure state was conceptually the other extreme of the capacity vs. preconnection relationship. To investigate the curve that we thought must exist between these extremes, Stamatelakis was running a Genetic Algorithm that was free to breed and combine populations of all arbitrary patterns of preconfigured spare capacity subnetworks. Preconnected linear segments, trees, partly closed trees, cycles, cycles with bars across them or pigtailed, and so on, were all valid individuals in the GA. Any preconfigured pattern on the graph was allowed. The GA embodied no *a priori* idea about the particular merits of any one elemental pattern types, but simply bred and selected populations seeking to enhance a fitness function which was to maximize the overall restorability level of the network using only preconfigured spare capacity structures. Any structure where no new cross-connections are required in real-time to effect a restoration path, other than the two always required at each end for traffic substitution into the restoration path, is said to be 100% preconfigured or "failure-ready." This was to be done within the pool of spare capacity provided from an optimal solution for SCA in the same network. The idea was that we would increase the total spare capacity budget and observe the preconfigured restorability level of the resulting designs. By preconfigured restorability we mean the fraction of all cross-connections for all restoration scenarios that would be found already made in the spare capacity (i.e., ready) when needed. This research framework is summarized in [Figure 10-3](#).

Figure 10-3. Research paradigm of the preconfiguration studies where p -cycles emerged: 100%

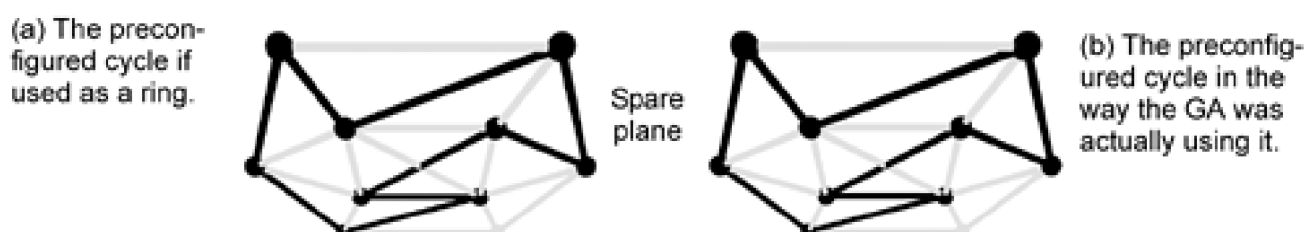
preconfigured protection structure but with *no* more spare capacity than a mesh.



The "surprise" is that such a continuum doesn't exist. Results were saying that even *without* adding any spare capacity above the strict minimum for span restoration we had reached 100% preconfiguration and...*all the patterns were cycles!*The first thought was that there might be some mistake; a bug in the program or the data summation routines. After all—how could we be fully pre-wired for switching that would be as fast and simple as in a ring, but only need the spare capacity of a mesh, not 100% or more spare capacity? When checking turned up no errors, the next thought was that somehow there must actually be a deep wisdom to the idea of ring-based networking that we had missed. (We too still only pictured cycles being used in a ring-like way at the time as well.) We were at a loss to explain the result, or find an error in the GA, until we drew what we call the "clamshell" diagram. The clamshell diagram chooses one cycle from the design and shows it in the "spare capacity plane" of the network. Below it, in the working capacity plane, we draw each of the spans that has working capacity associated with that protection structure.

The first clamshell diagram that we made of a *p*-cycle from the GA design is reproduced in [Figure 10-4](#). It showed us immediately that the finding was *not* a mistake, and made it clear why we were seeing the mesh spare capacity level *combined with* 100% preconfiguration of the protection structures, i.e., why we were sitting at the point labeled "surprise" in [Figure 10-3](#). The key was that the GA was *not* using the preconfigured cycles as rings. The GA had nothing to tell it that it could not use the cycle as we now understand and it was choosing cycles that (as we would now say) had a large number of straddling span relationships. If used as a ring (the way we and everyone else had always presumed a cycle would be used), [Figure 10-4\(a\)](#) shows that 9 hops of spare capacity protects 9 hops of underlying working capacity. [Figure 10-4\(b\)](#) shows, however, that the GA was using the same 9 hops of spare capacity to protect working capacity on 19 underlying spans in the working plane. This did not yet completely explain the low spare capacity requirement, however. The final penny dropped when we realized that on each of the 10 straddling spans in [Figure 10-4\(b\)](#), there were *two* units of working capacity being protected. So the GA was actually gaining a total of $9(1) + 10(2) = 29$ protection relationships out of only 9 units of spare capacity. As a individual subsystem the one *p*-cycle in [Figure 10-4\(b\)](#) and its associated working channels are together only 31% redundant. Thus, the clamshell diagram finally showed us how it was that the GA was achieving 100% preconfigured restorability with no more than the amount of spare capacity needed for a span-restorable mesh on the same graph.

Figure 10-4. The "Clamshell" diagram: discovery of the *p*-cycle idea by a Genetic Algorithm for breeding high efficiency preconfigured structures of spare capacity.





10.1.3 Other Important Properties of p -Cycles

There are a number of other important properties of p -cycles. p -Cycles can either be cross-connect based or based on an ADM-like "capacity slice" nodal device structure. In the OXC-based approach p -cycles are formed from individual spare wavelength channels, offering the greatest flexibility to adapt and evolve the p -cycle configuration. On the other hand, the ADM-like p -cycle nodal element offers the "pay as you grow" advantage of conventional rings. In either case, p -cycles avoid the structural association between the routing of working demands and the configuration of protection capacity. Unlike rings, p -cycles are formed only within the spare capacity layer of the network, leaving the working paths to be routed freely on shortest paths, or any other route desired. The configuration of the p -cycles is adapted to the working flow, not the other way around. It is important to note that when p -cycle network redundancy values are stated in various studies, this is already stated with respect to the most efficient shortest path routing of working demands. In other words, relative to rings there remains a further capacity saving just due to replacing ring routing with mesh-like shortest path routing. Some in the industry attribute as much as 30% working capacity savings overall due to this effect alone.

This freedom in the routing of working paths also makes p -cycle networks highly amenable to dynamic operation under the working capacity envelope concept described in the context of span-restorable networks in [Section 5.2.5](#). Under this approach, it is always known ahead of time what ultimate levels of working capacity (w_i values) can be protected on each span. As long as any new service path is routed over spans where the protection capabilities are not already fully employed (i.e., the path is routable within the current "protected working envelope," then the provisioning process need not have any workload related to ensuring survivability of the dynamically added path. We outline later how the configuration of p -cycles within the spare capacity of the network can also be self-organized adaptively to provide a protected working envelope suited to constantly evolving patterns of dynamic demand flow. Note also that the average length of protection paths in a p -cycle is half that of the corresponding ring for straddling spans, and the same as a BLSR ring for on-cycle spans.

[Table 10-1](#) summarizes these and other aspects of the p -cycle concept, as distinct from rings. p -Cycles can be formed from individual spare channels whereas rings commit a whole module of working and spare capacity to the same cycle. Each p -cycle can contribute up to two paths to a wider range of restoration scenarios than a ring. Rings also have a structural association between the working demands that they protect and the protection capacity in the same ring, while p -cycles are formed only within the spare capacity layer of the network leaving the working paths to be routed freely on any route desired. The p -cycles formed in the spare layer adapt to suit the working path layer. A deployed p -cycle design may also be easily modified by the OXCs that form it, whereas ring placements are essentially permanent structural commitments of both working and spare capacity to which the routing of new working paths must conform.

Our next aim is to address some concurrent developments that superficially may seem the same as p -cycles, but are not. Covering these now avoids later confusion between the concepts of enhanced rings, cycle double covers, and p -cycles.

Table 10-1. Comparison of p -Cycle and Ring Technologies

Attribute	p -cycles	SONET or DWDM rings
Modularity	Any add/drop or cross-connection signal unit.	OC-n or DWDM 40, 80, 160I, etc.
Protection Yield	Up to two restoration paths per unit of p -cycle capacity.	One restoration path per unit of ring protection capacity.
Protection Flexibility	Restores failures on the cycle and on cycle-straddling spans.	Each ring protects only spans contained in the same ring.
Routing and provisioning of working paths	May proceed without regard to protection structures, with p -cycles adapted to suit.	Routing must conform to deployed ring structures and inter-ring transition restrictions.

Attribute	<i>p</i> -cycles	SONET or DWDM rings
Network Redundancy	Typically that of a span-restorable mesh network. $1/(\bar{d} - 1)$ lower limit.	Typically well over 100% including protection and working routing inefficiencies. 100% lower limit.
Average Length of protection paths	One half of the <i>p</i> -cycle circumference for straddling span failures. Same as rings for on-cycle failures.	N-1 hops of the ring (essentially the full ring circumference outside the failed span itself).

10.1.4 Enhanced Rings

Having just compared *p*-cycles to conventional rings, this section is aimed at making the distinction between *p*-cycles and two other recent developments that at first glance seem similar. The first of these is what are sometimes referred to as "enhanced rings," although it is not known if this has become a standard term for the technique or not. The basic idea, US Patent [EIBu00], is for two otherwise conventional BLSR-type rings to be able to share a common protection channel on spans where the rings run in parallel, that is to say where, if the rings were tiling the plane, where two tiles are edge to edge.

To appreciate the motivation for this enhancement to normal ring-based networking, consider first the view of a capacity cross-section along the right-of-way indicated in the example of Figure 10-5, which is based on a partial zoom-in view of a North American IXC network with two postulated rings that have facing edges on the Salt Lake City to Los Angeles facility route. Each ring individually comprises a 100% matching of working and protection fiber capacity. If, as is not unusual, the working fiber "fill" is say, 40% or so (and it may tend to be even lower than this on north-south mid-western US routes due to demographic considerations), then the overall redundancy^[3] of each system may be 400%. In practice, the network operator sees two completely functioning, maintained and operating transmission systems, perhaps at OC-192 capacity, and several thousand miles long which are both completely idle and provided for protection only. This is a major investment that is in a sense doubly redundant.^[4] So the idea of enhanced rings is to make arrangements, in our example at Salt Lake City and Los Angeles, to at least improve the situation somewhat by allowing the two facing rings to share a single allocation of protection bandwidth. This is done through provision of a simple 2:1 selector switch at each of the sites bordering the common span, and arranging an exchange of signaling between the two rings so the availability of the common protection span can be coordinated. See [EIBu00] for more details. The effect is that the revised cross-section diagram becomes as shown in Figure 10-6.

^[3] By overall redundancy here we mean the ratio of total protection capacity plus unused working capacity to used working capacity.

^[4] This effect (side-by-side protection spans) is the main flaw with the often tempting notion in ring planning to simply "tile the plane" with the obvious rings that map onto faces of the network graph.

Figure 10-5. Example of the motivation for enhanced rings—side-by-side BLSR ring spans have 400% redundancy at 40% working fill.

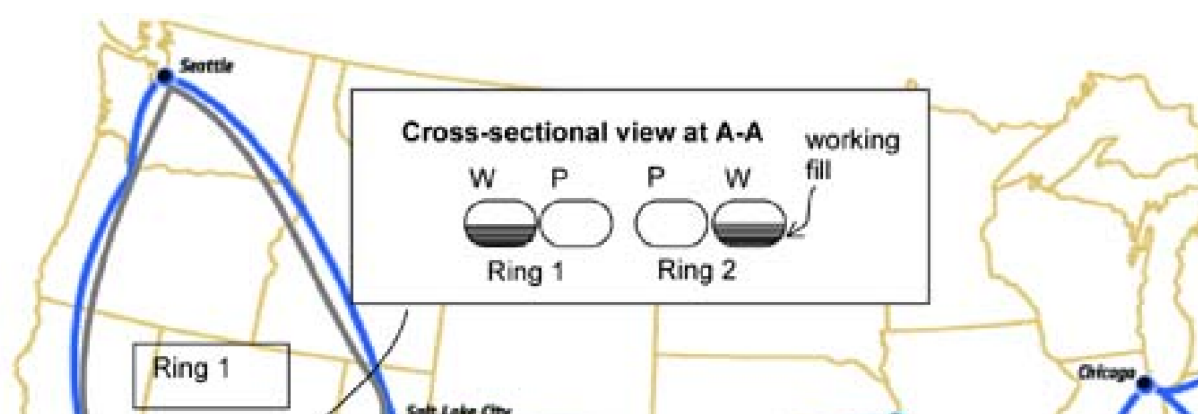
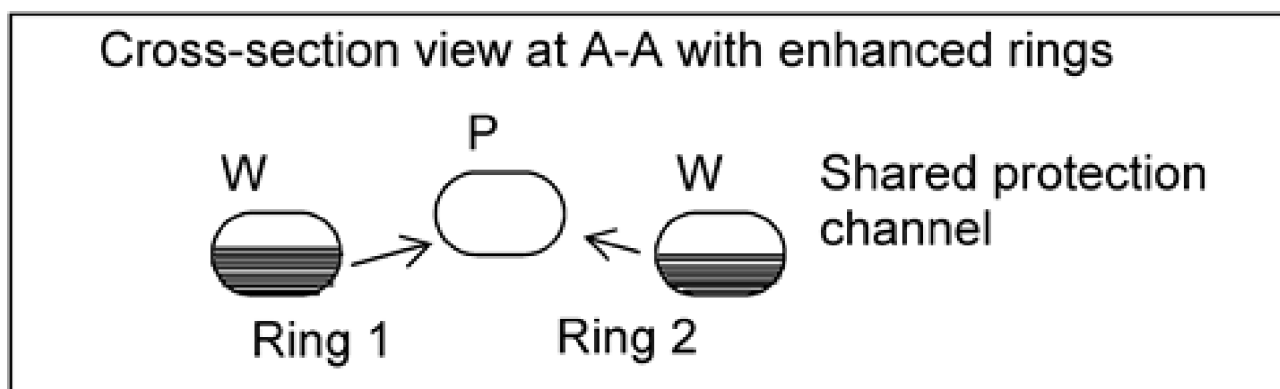


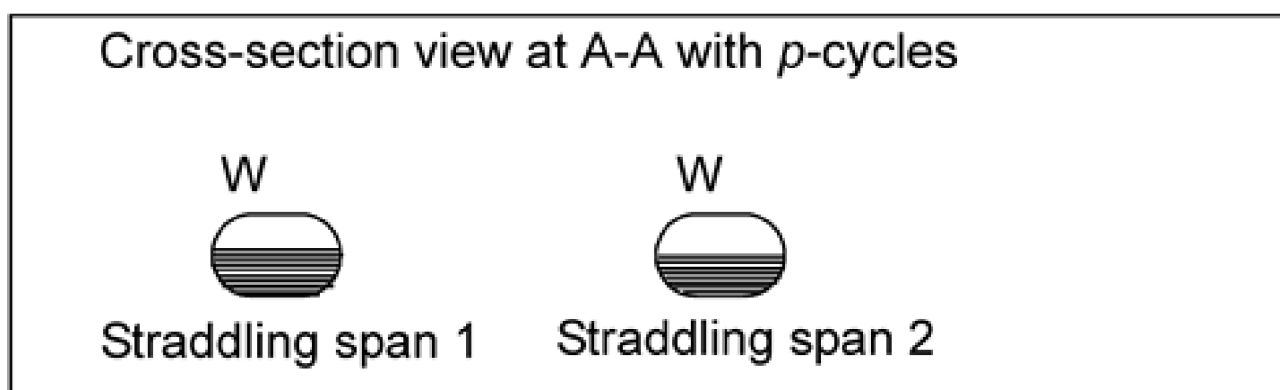


Figure 10-6. Cross-section view with enhanced rings—redundancy of 275% at 40% working fill by coordinating access to a common protection span.



So, does this amount to having constituted a p -cycle? The answer is no because these remain coupled BLSR rings with an allocation of working and protection capacity on every span and the ability only to protect against on-cycle failures. The key difference is again the aspect of straddling spans. If a true p -cycle was formed on the outer perimeter of the two rings above, the corresponding cross-sectional view in the same example would become that in [Figure 10-7](#).

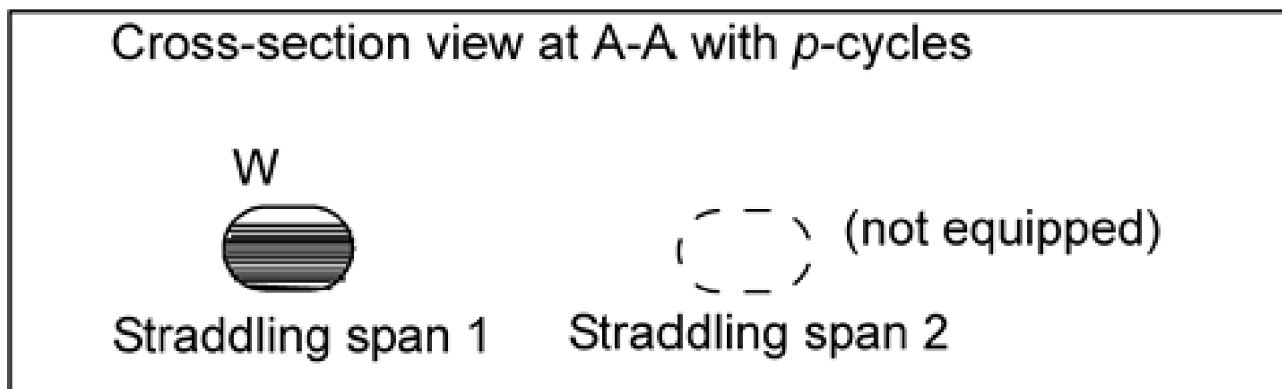
Figure 10-7. Cross-section view with a p -cycle if both straddling spans are equipped—no protection capacity on straddling spans at all, redundancy of 67% at 40% working fill.



In fact, however, the p -cycle approach allows an even further efficiency in this example that is categorically not available to the coupled ring scheme because each ring in the latter case must retain its complete cyclic integrity. Thus even if all the Salt Lake City to Los Angeles traffic was groomed over to one or the other ring only, the overall redundancy would not change because the other working fiber would still have to be there, completely intact even if empty, to complete its ring.

In contrast, however, although the p -cycle is *able* to protect up to two line-rate straddling spans, there is no need to do so if demand does not require it. Thus, in the same example, the further option available to us on the Salt Lake City to Los Angeles route, when protected by a p -cycle, is to have only one working-only line system, straddling the p -cycle. So the final view of the cross-sectional capacity investment becomes that in [Figure 10-8](#).

Figure 10-8. Cross-section view of a p -cycle with only one straddling span equipped—redundancy of 25% at 80% working fill.



Finally, it may be argued that the coupled rings should exhibit better availability because each ring individually has a smaller perimeter distance (i.e., exposure to line failures) than the single p -cycle we discussed. This is not necessarily an advantage in favor of the coupled rings, however, because with the sharing of access to a single protection span, the coupled rings in essence have the same total mileage exposure to a dual failure situation on the outer perimeter of themselves as a pair as does the p -cycle. A failure on the straddling span of the p -cycle, combined with a failure on the perimeter of the p -cycle, will cause outage, but so does the same combination on the coupled rings with shared protection spans. This is, however, a subject for detailed comparative availability analysis, beyond the present scope. In assessing p -cycles from an availability standpoint, however, a main consideration is that any scheme that protects against all single failures has already achieved the most significant availability enhancement over non-protected operation.

10.2 Cycle Covers and "Protection Cycles" per Ellinas et al.

Another area of work that could be superficially confused with p -cycles is the topic of cycle covers. In particular, a rather coincidental similarity in terminology between p -cycles and work on *oriented cycle double covers* by Ellinas et al. [EiHa00] needs to be explained. Let us start by explaining a cycle cover and cycle double covers. This makes the substantive differences between the latter and p -cycles apparent. Following that we only have to try to sort out the terminological confusion that has arisen in this area.

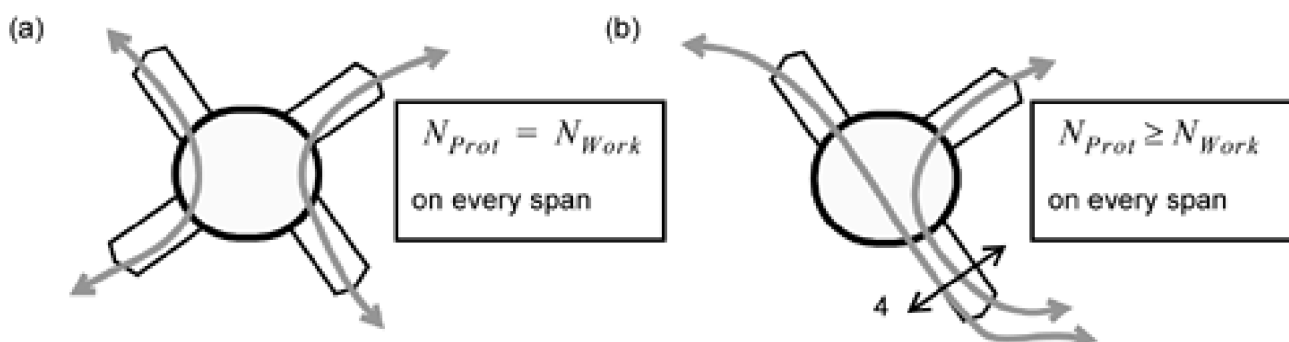
Given a graph $G(V, E)$ a cycle cover is any decomposition of G into elementary cycles such that each edge is present in at least one cycle. Conventional ring network planning is in a strong sense an exercise in finding a low-cost cycle cover. The cycles have to take on a capacitated nature in the general view, and in some cases demands may be rerouted to off load an edge from the cycle coverage requirement, a practice known as span elimination. But fundamentally every ring-based transport network comprises a cycle cover. As mentioned, in practice where the ring systems actually operate at the SONET OC- n or DWDM I- managed level, the problem is not just one of finding a set of rings that cover all spans of the facilities graph. In addition one needs to provide for various different capacity requirements on each edge, and wants to do so generally with a minimum number of ring systems and arranged in a near minimum cost configuration.

But there is a technical context where the survivability design problem can be almost reduced to that of finding a single unit-capacity logical cycle cover of the graph without further considerations of capacity. This is the paradigm of whole-fiber protection switching. Specifically, in a line of work by Ellinas [EiHa00] and others [MeBa02], the aim is to improve upon physically diverse 1+1 fiber-level APS arrangements. Ellinas sets the goal of achieving exactly 100% fiber-level redundancy in a network by trying to find cycle covers so that every span of working fiber is part of at least one covering cycle formed of dedicated protection fibers.

Another framework in which the cycle-covering approach has been developed is where a set of unit-capacity logical rings are established on a cross-connect based network. The logical operation is identical to that of a set of conventional discrete ring systems, but for flexibility and finer-scale capacity management the set of rings is created, switched, and managed at a unit-capacity channel level on cross-connects instead of fixed ADM nodes [Fee99]. In operating a set of unit-capacity logical rings created this way on cross-connects one can also define a set of rings that accommodates different numbers of working channels on each span.

Whether a single cycle cover of the graph is needed, or a set of multiple logical rings to cover a capacitated graph, an interesting thing which ring network planners know well, is that in general one cannot find a set of bidirectional covering cycles that overlaps every span only once. Some spans unavoidably wind up having two or more cycles overlapping on them in order to support a set of cycles that cover all spans. The basic reason for this can be seen by considering any node of odd degree. More generally if a graph is non-Eulerian, then more than one covering cycle is needed on at least one span because the covering cycles are degree 2 structures and yet at least one node in a non-Eulerian network is of odd degree. Figure 10-9 illustrates.

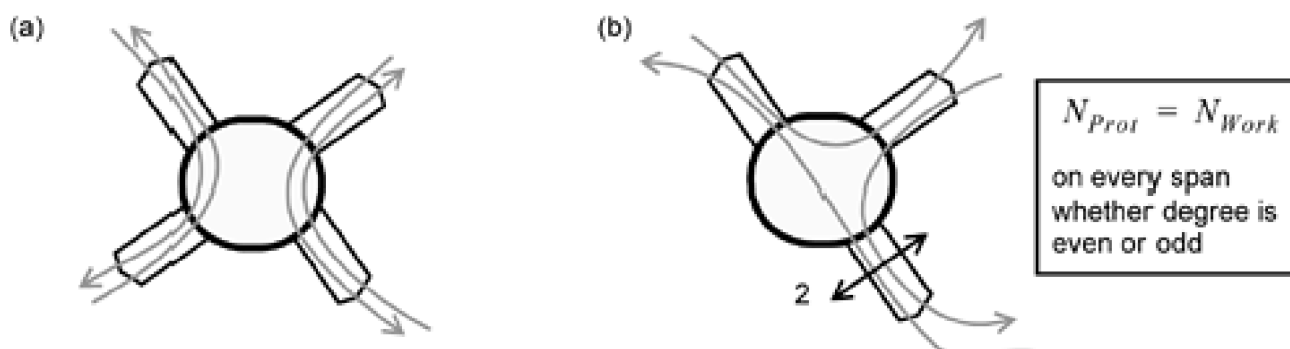
Figure 10-9. With bidirectional fiber pairs, a cycle-cover that overlies each span only once is possible (a) if all nodes are even-degree (an Eulerian network); (b) not possible if even one node has odd degree.



However, this is really only true if one is working with bidirectional logical rings, as is commonly the case because rings are either

bidirectional transmission systems or bidirectional associated channel pairs. What Ellinas observes is that the bidirectional nature of the actual transmission systems is usually realized by two fibers, one transmitting in each direction. Thus, at least at the single fiber level one could take a unidirectional or *oriented* view to the formation of single fiber cycles. One can then show that if the graph is planar it is always possible to find a set of *unidirectional cycles that covers every edge of the graph exactly twice*. Such a cycle set is called an *oriented cycle double cover* (O-CDC). The significance is that being covered exactly twice by oriented cycles is capacity-equivalent to being covered once by a bidirectional cycle. [Figure 10-10](#) illustrates how an odd-degree node can in fact have all spans covered only once (in the sense of bidirectional equivalence) by a suitably arranged set of *unidirectional* cycles.

Figure 10-10. With a directed (or oriented) cycle double cover it is possible to arrange that every span is covered by exactly two unidirectional cycles (in a planar graph).



At the odd degree node in [Figure 10-9](#) it is impossible to find a span cover with bidirectional systems that covers each span only once. But in [Figure 10-10](#) it is possible to find a set of unidirectional system layouts that does cover each span exactly twice. While [Figure 10-10](#) illustrates the central issue for one node, the important advance in [\[ElHa00\]](#) was to prove that this form of unidirectional cycle cover is feasible at all nodes of a (planar) network simultaneously. Thus, from a total fiber count standpoint, the goal of achieving exactly 100% matching (and no more) of protection and working fibers (for bidirectional services and on a planar graph) is achieved over a whole network.

Unidirectional "protection cycles" are, however, clearly not the same as p -cycles, notwithstanding the naming similarity. Rather coincidentally the principle of oriented cycle double covers to achieve 100% fiber-level protection redundancy [\[ElHa00\]](#) appeared in the same special issue of the *Journal on Selected Areas in Communications* where we first described p -cycle applications to the IP networking layer [\[StGr00\]](#). Unintentionally, both papers use the term "protection cycles" in their titles, sowing the seeds for confusion—especially as the editorial comments of that special issue also stated that we were both studying the same concept. But O-CDCs are clearly not p -cycles. The O-CDC approach only avoids the requirement to have more than 100% overall redundancy in a cycle cover. The protection action remains logically ring-like in all respects. There are no straddling spans that bear zero protection capacity anywhere in an O-CDC design. [Table 10-2](#) is a more comprehensive breakdown of other differences between O-CDCs and p -cycles.

^[5] At that time in work on p -cycles we had referred to the concept variously as *protection cycles* (in [\[StGr99b\]](#) [\[StGr00\]](#) for example on the MPLS layer applications of p -cycles) and also as *preconfigured cycles*, such as in [\[GrSt98\]](#) [\[GrSt98b\]](#) [\[StGr00b\]](#). We subsequently adopted " p -cycle" as the standard term—"p" being chosen in reference to both aspects of *protection* and *preconnection* and avoided further use of "protection cycle" per se to avoid confusion with cycle covering methods under the latter name.

Table 10-2. Comparative Properties of "Protection Cycles" and p -Cycles

Attribute	"Protection Cycles" [ElHa00]	p -Cycles [GrSt98] and "Virtual Protection Cycles" [StGr00]
Redundancy	Exactly 100% fiber-level redundancy.	Under 100%—essentially equivalent to a span-restorable mesh.
Protection Capacity	Found on every span of network that has working fibers.	All straddling spans bear up to two working systems with no protection capacity on the same span.
Conceptual Basis	O-CDC can achieve unit capacity ring cover without span overlaps.	Most efficient way to preconnect spare capacity in advance of failure in a mesh network [StGr00b] .

Attribute	"Protection Cycles" [ElHa00]	<i>p</i> -Cycles [GrSt98] and "Virtual Protection Cycles" [StGr00]
Application Domain	Whole-fiber level protection switching, or strictly unit capacity logical planning problems.	Applicable to fiber level, DWDM, SONET, MPLS, or ATM contexts. Deals with variable capacitation aspects, and amenable to oversubscription-based planning for MPLS/ATM.
Special Restrictions	Facilities graph is planar. Equal working capacity on all edges. Biconnected graph.	Biconnected graph.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

10.3 Optimal Capacity Design of Networks with p -Cycles

We now cover two basic MIP formulations for designing sets of p -cycles that were first developed and tested in [Stam97](#). Both of these are capacity-oriented models implying that in a DWDM context, wavelength conversion capabilities may be required at each OXC. This is also called the Virtual Wavelength Path (VWP) context, where λ -assignments may have to be changed along a service path to access required capacity in the next span on a route. In a SONET context the ability to change timeslots in the cross-connect node is the corresponding functionality. Later sections show how these basic capacity-based models can be extended to consider wavelength assignment as well.

The first formulation "*max Rp | spare*," produces a p -cycle plan with maximum restorability within a given amount and placement of spare capacity, such as from an existing mesh restorable design. Results with that formulation show restorability levels often at or very close to 100% with p -cycles formed within the existing disposition of spare capacity for an optimal span-restorable mesh. The second formulation "*min spare | Rp=1*" determines the set of p -cycles that minimize the total spare capacity required for strictly 100% p -cycle restorability. The high restorability achieved within a mesh capacity plan in *max Rp | spare* and conversely the minimal excess sparing above a mesh needed for *min spare | Rp=1* show that optimal p -cycle assemblies and span-restorable meshes are certainly close cousins, but not in some theoretical way exact equivalents.

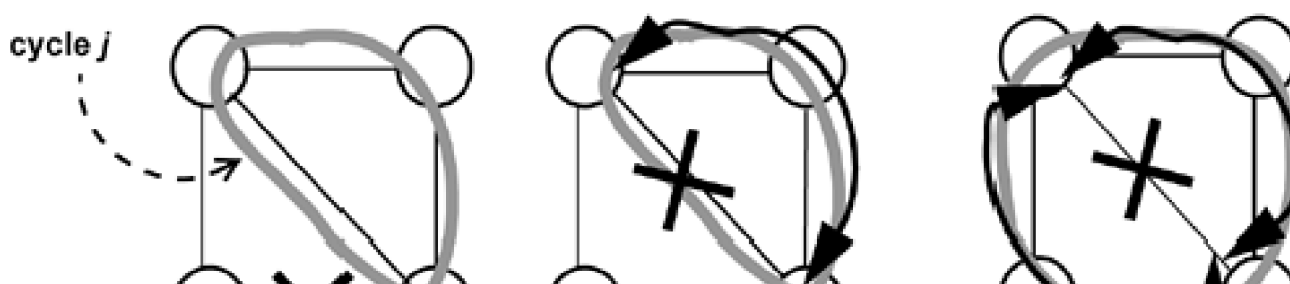
A preliminary step for both formulations is to generate the set P consisting of all elementary cycles of the network graph, or, more generally, the set of eligible cycles from which p -cycles may be formed in the design. By "elementary" cycles we mean cycles that may cross over themselves span-wise but do not "figure-eight" through any node. The set of cycles may be hop- or distance-limited and can be found with methods in [Chapter 4](#). Each of the following formulations produces an optimal p -cycle plan by choosing the number of unit-capacity copies of each elemental cycle, n_j , to be configured as a p -cycle.

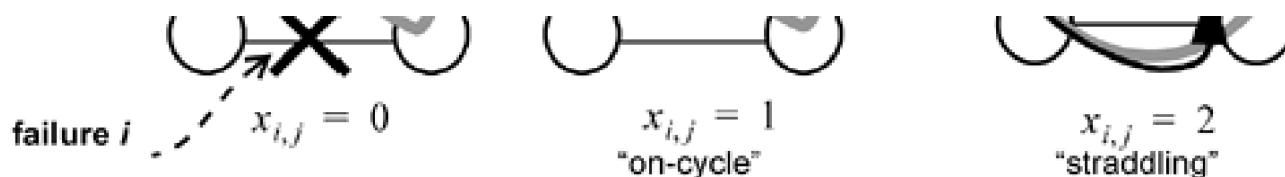
10.3.1 Concept of "Useful Paths" for p -Cycle Design

In preparing the two p -cycle design models below, and in p -cycle and preconnection design in general, a concept of "useful paths" characterizes the amount of protection that any given p -cycle (or preconnected pattern in general) can contribute to restoration of a specific span failure. A unit capacity p -cycle can provide either 0, 1, or 2 paths for use in restoring working capacity in any given span failure scenario. The notation $x_{i,j}$ refers to the number of useful paths p -cycle j can provide if span i fails. The three basic possibilities are illustrated in [Figure 10-11](#).

1. $x_{i,j} = 0$ if one or both end nodes of span i are not nodes of cycle j .
2. $x_{i,j} = 1$ if both end nodes of span i are nodes of cycle j and span i is *on* the cycle.
3. $x_{i,j} = 2$ if both end nodes of span i are nodes of cycle j and span i is *not on* the cycle.

Figure 10-11. The number of "useful paths" a p -cycle provides depends on its relationship to the failure.





It is important to keep in mind that the number of useful paths is the most the p -cycle could provide to help in a given restoration problem, but they will not be used, nor should various formulations give "credit" for $x_{i,j}$ quantities that cannot actually be used by the presence of a corresponding w_j quantity.

10.3.2 Maximum p -Cycle Restorability with a Given Spare Capacity

The first formulation chooses a set of p -cycles that provides the highest level of restorability (against all single-span failures) that it is possible to achieve within a given set of existing spare capacity quantities, s_j . The following generates a selection of p -cycles that maximizes restorability within the existing spares. The other parameters and variables are:

- S is the set of network spans.
- P is the set of elementary cycles of the graph (or later—a reduced set of eligible candidate cycles)
- $d_{k,j}$ is 1 if cycle j includes span k , 0 otherwise.
- $x_{i,j} \in (0, 1, 2)$ is the number of useful paths cycle j provides for restoration of span i .
- w_j, s_j are the number of working and spare channels, respectively, on span j (integer).
- r_j is the number of available protection paths in the design in excess of those required for span j (a slack variable).
- u_i is the number of unrestorable working channels on span i .
- n_j is the number of unit-capacity copies of cycle j in the design (integer).

Before proceeding, a note about indexing will be timely. While it is mathematically sufficient to re-use one general index variable over within any set (they are local not global variables in effect), it aids understanding and checking of models in practice if a convention is adopted about the context that different range variables imply. Here, we try to reserve i to indicate that a span is being considered in the context of a failure span, and j to enumerate p -cycles or candidate p -cycles. At times this means that a third, general index, is required to enumerate spans in their ordinary non-failed context. We use k for this context and as a general index in other contexts.

The objective function is:

PC-1

Minimize

Equation 10.1

$$\sum_{i \in S} w_i$$

Subject to:

1. The spare capacity on each span can support the p -cycles that cross it:

Equation 10.2

$$s_k \geq \sum_{j \in P} \delta_{k,j} \cdot n_j \quad \forall k \in S$$

2. Every span accesses enough protection coverage (ideally) for all its working capacity:

Equation 10.3

$$u_i + \sum_{j \in P} x_{i,j} \cdot n_j = w_i + r_i \quad \forall i \in S$$

3. The unrestorable portion of any span cannot be more than its total working capacity:

Equation 10.4

$$0 \leq u_i \leq w_i \quad \forall i \in S$$

4. Numbers of each p -cycle copy and slack variable (excess restorability) are non-negative:

Equation 10.5

$$n_j \geq 0 \quad \forall j \in S \quad r_i \geq 0 \quad \forall i \in S$$

The coefficients $x_{i,j}$ and $d_{k,j}$ are evaluated in advance for each cycle $i \in P$. The w_i quantities are also given in advance for this model, as integers, arising from shortest-path routing or any other manner of routing the working paths. Note that [Equation 10.3](#) could alternatively be written as an inequality but instead uses the slack variable r_i . A side effect of fully protecting all spans is that some may have more protection built into the design than they individually require. The slack variable r_i allows this data to be observed directly in the output. Note that one of u_i or r_i is implicitly zero on each span. The design output is the n_j variables specifying how many copies of each cycle j from P to deploy—in effect, how much capacity to lay down on each p -cycle that it is decided to use. $n_j = 0$ indicates the p -cycle is not used by assigning it zero capacity.

10.3.3 Minimum Spare Capacity for 100% p -Cycle Restorability

Now we look at the converse problem where we *generate* the spare capacity and a of capacitated p -cycle specifications to ensure 100% restorability by design. This is the model for p -cycles that corresponds to the basic SCA model for span-restorable networks. The only new parameter relative those for PC-1 is c_j , the cost or length of span j . The *min spare* / $Rp=1$ problem is then:

Minimize

Equation 10.6

$$\sum_{k \in S} c_k s_k$$

Subject to:

1. Spare capacity on span j is sufficient to support all the p -cycles that cross it.

Equation 10.7

$$s_k = \sum_{j \in P} \delta_{k,j} \cdot n_j \quad \forall k \in S$$

2. The number of useful paths provided for each span supports 100% restorability.

Equation 10.8

$$w_i \leq \sum_{j \in P} x_{i,j} \cdot n_j \quad \forall i \in S$$

Equation 10.9

$$n_j \geq 0 \quad \forall j \in P \quad s_k \geq 0 \quad \forall k \in S$$

10.3.4 Adding a Span Capacity Constraint

As so far stated, this model minimizes spare capacity needed to support required p -cycles, without any explicit limit on the capacity of any span. In the DWDM optical networking context, it may typically be that one wishes to minimize the total use of spare wavelength channels to support the needed p -cycles, but also keep in mind that it is desirable to implement the whole network design using only a specific or a maximum number of whole fibers (or bidirectional fiber pairs) on each span. In fact when DWDM technology starts supporting 256 to ultimately perhaps 1000 ls per fiber, it may commonly be that we are interested in planning networks that use only one or two fibers per span. This consideration is easily added to the PC-2 model by addition of the following:

- K = set of wavelengths available on each fiber.
- F_k = number of (bidirectional) fiber pairs available on span k .

PC-2A

Minimize

Equation 10.10

$$\sum_{k \in S} c_k s_k$$

subject to: [Equation 10.7](#) though [Equation 10.9](#) plus:

Equation 10.11

$$w_k + s_k \leq |K| \cdot F_k \quad \forall k \in S$$

10.3.5 Results with Basic Capacity Formulations

Results obtained with the PC-1 (*max Rp | spare*) and PC-2 (*min spare | Rp=1*) design models are summarized below for five test networks. In [Table 10-3](#) the PC-1 (*max Rp | spare*) model for *p*-cycle design is being tested within the spare capacity distribution of a conventional span-restorable mesh network. In these results *P* included all distinct cycles with no hop limit for test networks 1, 2 and 3. For computational reasons, the cycle length was limited to 25 and 12 hops respectively, in networks 4 and 5. The restorability performance within the existing spare capacity of a span-restorable mesh, given in [Table 10-3](#), ranged from 87% to 97% in the test networks. The low of 87% in Net 5 is due to the relatively restricted cycle set (hop limit of 12) in the presence of the high average nodal degree of this network. (When the span-restorable reference design which define the spare capacity distribution for these test networks uses a hop limit that is the same as the cycle-circumference limit used for the *p*-cycle designs, we are inherently handicapping the *p*-cycle designs.) The remaining networks had preconfigured restorability in the range of 94% to 97% under the existing spare capacity of the optimal span-restorable SCA designs.

Table 10-3. Test Results for maximum *p*-cycle restorability with minimum mesh spare capacity

Net	Total Working	# Nodes	# Spans	Number of <i>p</i> -cycles ^[a]	Rp Restorability (%)
Net1	142	10	22	5	93.66
Net2	1404	15	28	88	96.58
Net3	4369	20	31	250	96.86
Net4	27522	30	59	2237	> 94.93
Net5	2191	53	79	161	> 87.26

^[a] These are the counts of individual unit-capacity *p*-cycles employed. The number of distinct cycles these *p*-cycles are formed upon is fewer.

Results with PC-2 are shown in [Table 10-4](#). Here each network was, by design, 100% restorable through preconfigured cyclic paths alone and the efficiency of optimal *p*-cycle spare capacity designs for these networks is compared to corresponding *optimal* spare capacity designs for span-restorable mesh networks using the basic SCA design method ([Section 5.3.4](#)). It was validated by a separate restoration test program that these designs were indeed fully restorable through their *p*-cycles only. Thus for each of these designs, 100% restorability

is achieved *solely* through breaking into cycles to perform traffic substitution at the two end-nodes of the failure. In [Table 10-4](#), "Excess Sparing" is the percentage of total spare capacity that the p -cycle design required above the mesh spare capacity design reference. The remarkable results are that the extra spare capacity to support 100% restorability through preconfigured cycles ranged from zero to 9% above a span-restorable mesh. In other words, with an optimally selected set of cycles, we are seeing completely preconfigured restoration, where only two failure nodes do any real-time switching—which has heretofore only ever been a property obtained with rings—but with little more sparing than in a conventional span-restorable network. This drives home the point that the p -cycle principle is essentially mesh-like in its efficiency. The last two columns of [Table 10-4](#) record the number of individual unit capacity p -cycles employed

(in effect these values are $\sum n_j$ from the solution) and the number of distinct cycles employed. The latter indicates the "system count" of different p -cycle structures that would be needed. Each would have its corresponding n_j value as its capacity.

Table 10-4. Results for Minimum p -Cycle Spare Capacity for Full Restorability

Net	Total Working	# Nodes	# Spans	Excess Spare Relative to SCA	Number of Unit p -Cycles	# Distinct Cycles
Net1	142	10	22	9.09 %	5	5
Net2	1404	15	28	3.07 %	88	10
Net3	4369	20	31	0.0 %	250	10
Net4	27522	30	59	2.38 %	2237	27
Net5	2191	53	79	0.0 %	161	39

10.3.6 Joint Optimization of Working Path Routing and p -Cycle Placement

In the models above, we take the working capacity values of spans to be protected, the w_j values, as inputs. In other words, the routing of demands is done ahead of the placement of p -cycles for protection of those demands. In principle, a reduction in total capacity should be obtained by coordinating the routing of working paths with the placement of p -cycles. In other words, this will be a joint formulation in the same sense of the prior formulations for ordinary span and path-restorable mesh networks in [Chapters 3](#) and [4](#). The corresponding *joint* p -cycle formulation is put together by combining elements of prior formulations. The additional parameters and variables associated with the working path routing are:

- D = Set of O-D pairs or relations with non-zero demands, values d^r (integer), index r .
- Q^r = A Set of eligible working routes for r^{th} O-D pair, index q .
- $g^{r,q}$ = Demand flow assigned to the q^{th} eligible route to serve relation r (integer).
- $\xi_k^{r,q} = 1$ if the q^{th} working route for the r^{th} O-D pair uses span k , 0 otherwise.
- w_j = Working capacity needed on span j ; to be protected by p -cycles.

The relationships between these elements establishes the w_j quantities, based on the routing decisions $g^{r,q}$. Once w_j quantities are defined, the rest of the p -cycle placement model for minimum capacity is really the same as **irPC-2**. So the complete model becomes:

Minimize:

Equation 10.12

$$\sum_{k \in S} (w_k + s_k) \cdot c_k$$

Subject to:

1. The demands are fully routed:

Equation 10.13

$$\sum_{q \in Q} g^{r,q} = d \quad \forall r \in D$$

2. Span working capacities supports the routing of all demands:

Equation 10.14

$$\sum_{r \in D} \sum_{q \in Q} g^{r,q} \cdot \zeta_k^{r,q} = w_k \quad \forall k \in S$$

3. Spare capacity on span k is sufficient to support all the p -cycles that cross it:

Equation 10.15

$$s_k \geq \sum_{j \in P} \delta_{k,j} \cdot n_j \quad \forall k \in S$$

4. The number of useful paths provided by p -cycles is enough for 100% restorability:

Equation 10.16

$$w_i \leq \sum_{j \in P} x_{i,j} \cdot n_j \quad \forall i \in S$$

Equation 10.17

$$n_j \geq 0 \quad \forall j \in P \quad s_k \geq 0 \quad w_k \geq 0 \quad \forall k \in S$$

The model as stated routes the total demand for each O-D pair over possibly several distinct routes, although keeping the flow on each route to discrete quantities if $g^{r,q}$ is made integer. This may be technically reasonable if the d are lightpath quantities. If d represents a packet flow over a label switched path, however, the flow should not be split. This can be reflected, while still allowing for a choice of the

single route from many possible working path routes, by changing [Equation 10.14](#) to:

Equation 10.18

$$\sum_{r \in D} \sum_{q \in Q} \gamma^{r,q} \cdot \zeta_j^{r,q} \cdot d^r = w_j \quad \forall j \in S$$

Equation 10.19

$$\sum_{q \in Q} \gamma^{r,q} = 1 \quad \forall r \in D$$

where $\gamma^{r,q} \in \{0, 1\}$ is a binary decision variable for the one route to which all flow for relation j is assigned. Some results comparing joint to non-joint p -cycle designs using the PC-3 formulation follow in [Section 10.6](#). They show that joint optimization has quite a significant beneficial effect for p -cycles, in the range of a 20 to 30% improvement in (standard) redundancy.

Modularity

Another change that can be made to better suit detailed network planning is to assert modular capacity. (Generally in comparative studies or research a unit-capacity model is more useful, but in production planning tools, modular system capacities should be reflected.) To add modularity, the following changes are made.

- M = A set of modular capacity sizes, indexed m .
- Z^m = Number of capacity units for the m^{th} module size.
- C_k^m = Cost of a module of the m^{th} capacity size if installed on span k .
- t_k^m = Number of modules of the m^{th} size added on span k (integer variable).

The objective function changes to:

PC-3(mod)

Minimize:

Equation 10.20

$$\sum_k \sum_m t_k^m \cdot C_k^m$$

$$\overline{k \in S} \quad \overline{m \in M}$$

and the following modular capacity constraint is added to the model:

Equation 10.21

$$w_k + s_k \leq \sum_{m \in M} t_k^m \cdot Z^m \quad \forall k \in S$$

Note that in the modular capacity case, both w_k and s_k become intermediate logical-capacity variables that can be kept for additional information about the design, or eliminated from appearing explicitly anywhere by substituting [Equation 10.14](#) and [Equation 10.15](#) into [Equation 10.21](#) for w_k and s_k , respectively.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

10.4 Application of p -Cycles to DWDM Networks

As treated above, the only constraint or consideration on the use of a span for routing an additional working path or creating part of an additional p -cycle is the total capacity to do so is provided or present. In WDM optical networking, this is called the Virtual Wavelength Path (VWP) paradigm because an end-to-end service path (that a user application perceives as a lightpath) can be constructed by switching to whatever wavelength as might be needed to access the required capacity (a l -channel) at each hop en route. This is conceptually analogous to SONET DCS-based mesh networking where the cross-connects can change timeslot assignment at any node. In this section we will look at the design models and issues involved in planning DWDM p -cycle-based networks where there is either full wavelength conversion at every node (the VWP case), or no wavelength conversion anywhere (the WP case—refer back to [Section 2.3](#) if needed). Later we will briefly look at an idea for strategically placing a relatively small number of wavelength converters in a DWDM p -cycle network so that the capacity penalty from any remaining wavelength mismatch blocking is insignificant.

10.4.1 Wavelength Continuity—the WP case

In Wavelength Path (WP) networking, however, the end-to-end path between O-D nodes must be assigned a single l and no wavelength changes are assumed possible en route. In other words, the initial l assignment has continuity over the whole logical path. This is the analogous case to routing in a SONET ring. Virtually all SONET ring ADMs had a timeslot assignment capability at the entry and/or egress add/drop locations, but which on the ring a feasible routing had to be found without involving any change of timeslot assignment.

For mesh networks in general, the non-protected version of the WP networking problem is known as the (routing and wavelength assignment) RWA problem (discussed in [Chapter 2](#)). In the RWA problem one has a certain number of distinct l s to work with on each fiber. l s can be reused on separate fibers. WP cross-connects en route can only change a signal's appearance in space, not the signal can go out on any bidirectional fiber system other than the one it came in on, but it has to do so on the same l . If that l is not available on a given fiber, then that fiber is not available for that path. Thus, the RWA problem focuses on assigning both a route and a wavelength to each demand in a way that minimizes the total fiber capacity required.

10.4.2 Dark-Fiber p -Cycles: Protection without any Wavelength Conversion

One approach to WP p -cycle planning would therefore be to take existing RWA solutions as starting points, equivalent in the VWP context to generating the initial w_i quantities, prior to solving a separate optimization problem for the placement of protection resources only. This would be the WP equivalent to the basic non-joint p -cycle problem PC-2. The result of the RWA stage would be aw_i -like capacitation of each span. Inside each span there would be a particular mapping of all the l s used to individual demands between O-D pairs on the network, but an interesting and simplifying observation is possible at this point: to protect against fiber failures or span cuts, *the l -assignments within the failed fibers are irrelevant as long as the p -cycle fibers support the same waveband*. This means that despite the general recognition that requiring l -continuity greatly complicates the basic service routing problem, there are no further complications due to protection considerations if p -cycles are used at the fiber-switching level to protect WP networks. This is one enormous advantage of span restoration techniques in general (of which p -cycles are one) as opposed to path restoration or protection techniques. This application of p -cycles for fiber-level protection of WP networks could effectively be called *dark-fiber p -cycles*. Its minimum cost design solution uses only PC-2 where the w_i quantities are given by the number of working fiber pairs needed from a solution to RWA (or fiber requirement forecasts directly). Implicitly every l -assignment retains continuity under protection rerouting because the corresponding l is by definition free for use on the dark fiber if not already in use protecting another failure. No new results or theory are needed for this strategy for fully protecting a DWDM network *without l -conversion*.

10.4.3 p -Cycle Wavelength Path (WP) Design Problems

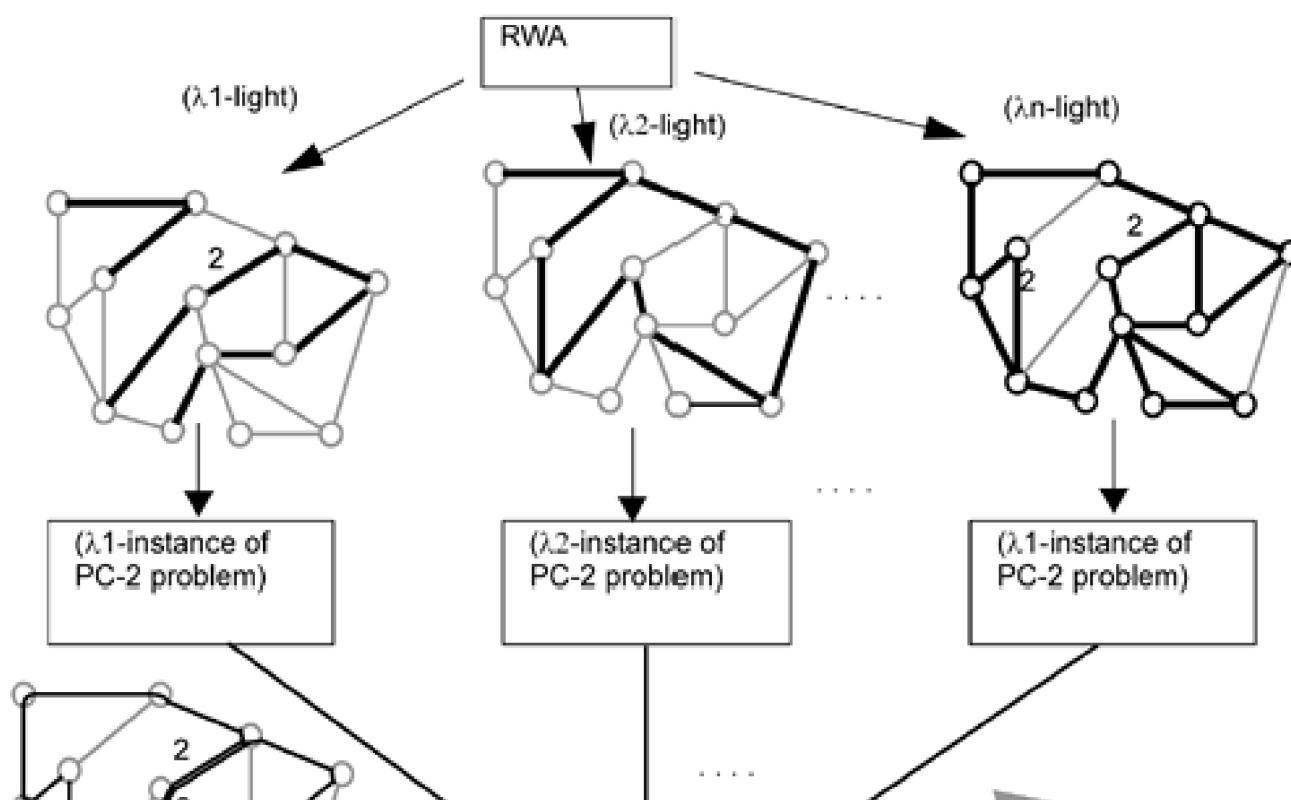
But another approach to the problem is to view p -cycles as being constituted at the l level, not the fiber level. Then, in the WP context, if a p -cycle is accessed by a specific l -channel that has failed, the specific logical unit p -cycle must be on the same l . This adds much computational complexity to the problem because now there may be a separate p -cycle on every l as well as repeated use of that l in different p -cycles not using any of the same fibers. In addition there is the equivalent to joint and non-joint planning contexts, which we look at in this section. In summary these two cases—which are considered in this section—consist of:

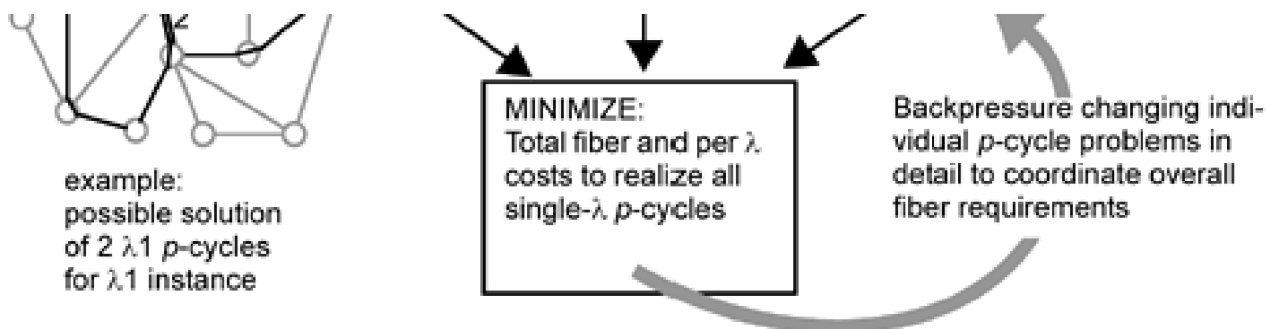
1. *Non-joint p -cycle WP context:* Here the problem is establishment of single l p -cycles to protect a base of already routing paths with WP wavelength assignments already made.
2. *Joint p -cycle RWA context:* Here, essentially everything is decided together—the route of each working demand, the WP wavelength assignment for each working path, and the layout and wavelength assignment of p -cycles. These are all jointly optimized.

Case 1. Non-Joint WP Design

In this design context some prior RWA process has already routed all demands, assigned wavelengths to them, and generated fiber requirements on each span. Since each p -cycle must also as a whole be assigned a single l and can only protect working l -channels of the same color, there is a rather strong decomposition structure that we can recognize in this problem. We call this a l -light capacitated decomposition. The basic idea is to view the entire network after RWA in glasses that only let us see a certain color or l . Each l -light view provides an individual capacitated subnetwork within which a same l p -cycle planning problem occurs. In fact for the significant special case of networks supporting enough l s on each fiber to require only one bidirectional fiber per span, this decomposition may be a suitable way to solve the whole l -level p -cycle planning problem. More generally, the subproblems are (still rather loosely, however) coupled together under the objective of minimizing total fiber counts. [Figure 10-12](#) illustrates this view of the problem structure. In each l -light subnetwork, the w_j quantities are assumed all to be unity except where annotated otherwise. (As sketched the example assumes an RWA stage that has enough l s to work with so that having to instantiate a second fiber pair on a span is relatively rare. But in general each l -light subnetwork can have its own arbitrary w_j capacitation.

Figure 10-12. A view of the (non-joint) WP p -cycle planning problem as multiple instances of PC-2, one in each l , coupled to minimize total fiber and l costs.





The corresponding problem formulation uses other parameters and variables as defined above with the new definition of: [\[6\]](#)

^[6] In this model index k is re-assigned to index wavelengths. i is still strongly reserved for a span in its failure context, but j is henceforth used to index either a span in its non-failed context or to index candidate p -cycles. A new index p is required in some cases to avoid using the same index twice in one expression.

- K = set of wavelengths available on each fiber, index k .
- nf_j = the number of fibers available on span j (each bearing $|K|$ wavelength channels).
- w_j^k = number of instances of wavelength k required on span j (given as an input from the RWA problem).
- cf_j = cost of a fiber (unidirectional demands) or a fiber pair (bidirectional demands) on span j (Common infrastructure cost to have the fiber and support up to $|K|$ wavelengths)
- c_j^λ = cost per incremental wavelength actually used ("turned up") on a fiber.
- s_j^k = the number of instances of wavelength k on span j to form p -cycles in wavelength k .
- n_p^k = number of copies of a p -cycle instantiated on cycle p in wavelength k .
- $d_{j,p}$ = one if cycle p crosses span j , zero otherwise.

WP-PC-1

Minimize

Equation 10.22

$$\sum_{j \in S} cf_j \cdot nf_j + \sum_{k \in K} \sum_{j \in S} c_j^\lambda \cdot s_j^k$$

Subject to:

1. Spare capacity on span j can support all the p -cycles that cross it in wavelength k :

Equation 10.23

$$s_j^k = \sum_{p \in P} \delta_{j,p} \cdot n_p^k \quad \forall j \in S \quad \forall k \in K$$

2. The total number of useful paths provided by p -cycles for each span is enough for 100% restorability in each wavelength subproblem:

Equation 10.24

$$w_i^k \leq \sum_{p \in P} x_{i,p} \cdot n_p^k \quad \forall i \in S \quad \forall k \in K$$

3. The number of fibers on each span must meet or exceed the maximum usage of any wavelength on that span:

Equation 10.25

$$nf_j \cdot |K| \geq w_j^k + s_j^k \quad \forall j \in S \quad \forall k \in K$$

plus all variables are bounded to be non-negative and integer.

Something to note is that to minimize the total cost of both fiber and wavelength-channels turned up, we have to put the initial working wavelength requirements on each span into [Equation 10.25](#). Even though by themselves they are not changeable quantities, their inclusion lets the spare capacity planning of the wavelength assignments take their presence into account to avoid triggering additional whole-fiber quantities that may not otherwise be needed.

Constraints From Already-Placed Fiber System Capacities

A closely related model does not try to minimize the number of fiber systems placed as is part of the objective function in WP-PC-1, but rather accepts as an input that a certain limited number of fiber systems are in place on each span, analogous to PC-2A. In this case we still want to minimize the total cost of individual channels used in the design, but the fiber counts on each span become the basis of a total use constraint per span on each span, not part of the cost function. While models like WP-PC-1 try to minimize total design cost to build the network to a forecast demand, this variation reflects a more operational point-of-view where the fiber systems are already built and we wish to do updates to find a p -cycle structure set that most economically protects a new demand pattern, routed within the as-built capacity of the network spans. The model becomes:

WP-PC-1A

Minimize

Equation 10.26

$$\sum_{k \in K} \sum_{j \in S} c_j^k \cdot s_j^k$$

subject to: [Equation 10.23](#) though Equation plus:

Equation 10.27

$$w_j^k + s_j^k \leq F_j \quad \forall j \in S \quad \forall k \in K$$

where, as above, F_j is the number of already-placed fiber systems on span j . The significance of the number of fiber systems is that one can only use each l once per fiber. In contrast to [Equation 10.11](#) that expresses $|S|$ simple constraints on total capacity for each span in the VWP (or capacity-only) model of PC-2A, [Equation 10.27](#) expresses $|S||K|$ constraints saying that each wavelength individually cannot be used more than once on each fiber on each span.

Case 2. Jointly Optimized WP ρ -Cycle Design

The final model is for the joint-WP case. There is little doubt that this stands as the most computationally complex problem we have so far identified and we do not mean to suggest that it might easily be solved at medium or large scale in planning tools. We present it for completeness nonetheless and because its solution on suitably small test cases can still be valuable in comparative studies seeking heuristics to approach the problem. Parameters and variables have already been introduced at various stages above, except that we must replace the previous $g^{r,q}$ variables (which assign whole number units of flow to working routes in the multi-fiber case) with 1/0 decision variables to decide the single route taken by each individual lightpath requirement. In this model the way to represent multiple lightpath demands between the same end nodes is by defining more than one relation between those entities:

- $b^{r,q} = 1$ if the WP for relation r is assigned to its q^{th} eligible route, 0 otherwise.
- $a^{r,k} = 1$ if the WP for relation r is assigned wavelength k in K

and as before we use the indicator parameters that show which spans are in each route:

- $c_j^{r,q} = 1$ if the q^{th} working route for the r^{th} O-D pair uses span j , 0 otherwise.

The objective function is the same as [Equation 10.22](#) but with the addition of costs per channel used in working path routing as a variable cost component:

WP-PC-3

Minimize

Equation 10.28

$$\sum_{j \in S} c f_j \cdot n f_j + \sum_{k \in K} \sum_{j \in S} c_j^k \cdot (s_j^k + w_j^k)$$

Subject to:

1. Each lightpath demand is assigned a route:

Equation 10.29

$$\sum_{q \in Q^r} \beta^{r,q} = d^r \quad \forall r \in D$$

2. The routed demands generate the working capacity requirements per wavelength on each span:

Equation 10.30

$$\sum_{r \in D} \sum_{q \in Q^r} \beta^{r,q} \cdot \alpha^{r,k} \cdot \zeta_j^{r,q} = w_j^k \quad \forall j \in S \quad \forall k \in K$$

3. Spare capacity on span j can support all the p -cycles that cross it on wavelength k :

Equation 10.31

$$s_j^k = \sum_{p \in P} \delta_{j,p} \cdot n_p^k \quad \forall j \in S \quad \forall k \in K$$

4. The total number of useful paths provided by p -cycles for each span is enough for 100% restorability in each wavelength:

Equation 10.32

$$w_i^k \leq \sum_{p \in P} x_{i,p} \cdot n_p^k \quad \forall i \in S \quad \forall k \in K$$

5. The number of fibers on each span must meet or exceed the maximum usage of any wavelength on that span:

Equation 10.33

$$w_j^k + s_j^k \leq n f_j \cdot |K| \quad \forall j \in S \quad \forall k \in K$$

Equation 10.34

$$\beta^{r,q} \in \{1, 0\} \forall (r, q) \in D \times Q^r, \quad \alpha^{r,k} \in \{1, 0\} \forall (r, k) \in D \times K$$

plus non-negativity and integral declarations for other variables.

The extension to the case where fiber system quantities are already placed is omitted as it follows rather directly the manner of conversion of WP-PC-1 into WP-PC-1A. We refer to such model nonetheless as WP-PC-3A. Some comments on WP-PC-3 are warranted here,

however, in that as written [Equation 10.30](#) states the problem relationships correctly, but it involves the product of two 1/0 variables. As such it is not linear and cannot be handled by software such as AMPL/CPLEX. This arises because in the *joint WP* case we have to make 1/0 decisions about the choice of working route, $b^{r,q}$ and about the choice of wavelength to assign, $a^{r,k}$. In [Equation 10.30](#) it is necessary to combine these decisions to generate working capacity in each wavelength on each span—but the expression becomes non-linear. One way (albeit brute force) around this is to create a single even higher-dimensional 1/0 decision variable, $g^{r,q,k}$ which is one if the q^{th} eligible route for relation r in wavelength k is chosen. A related radio button constraint assures that only one combination in the λ - k space for each r can be chosen. In practice, however, the joint WP case is a most difficult problem to pose for direct ILP solution. An effective heuristic alternative is to solve the joint VWP problem (PC-3) by an ILP and then use an RWA algorithm to make the wavelength assignments. This decoupling is efficient because once the joint VWP problem is solved, exact routes are defined so any feasible RWA solution on these routes is nearly optimal. Moreover, if warranted the RWA solution can be refined or iterated as needed to eliminate any cases where a whole extra fiber was triggered for a few extra needed channels, arising from wavelength mis-matches.

10.4.4 Summary and Discussion of p -Cycle Design Models

Let us now pause to summarize, as a sort of user guide, the context and use of each of the design models so far introduced. This is the role of [Table 10-5](#).

Table 10-5. Summary of p -Cycle Design Methods Applicable to DWDM Networking

	Demands routed prior to spare capacity optimization (non-joint problems)		Jointly optimized working routes and spare capacity	
	Generate spare capacities	With fiber capacity limits	Generate total capacities	With fiber capacity limits
Capacity-only planning (VWP)	PC-2 Section 10.3.3	PC-2A Section 10.3.4	PC-3 Section 10.3.6	Add constraints of PC-2A
Planning with I-continuity (WP)	WP-PC-1 Section 10.4.3	WP-PC-1A Section 10.4.3	WP-PC-3 Section	Add constraints of WP-PC-1A

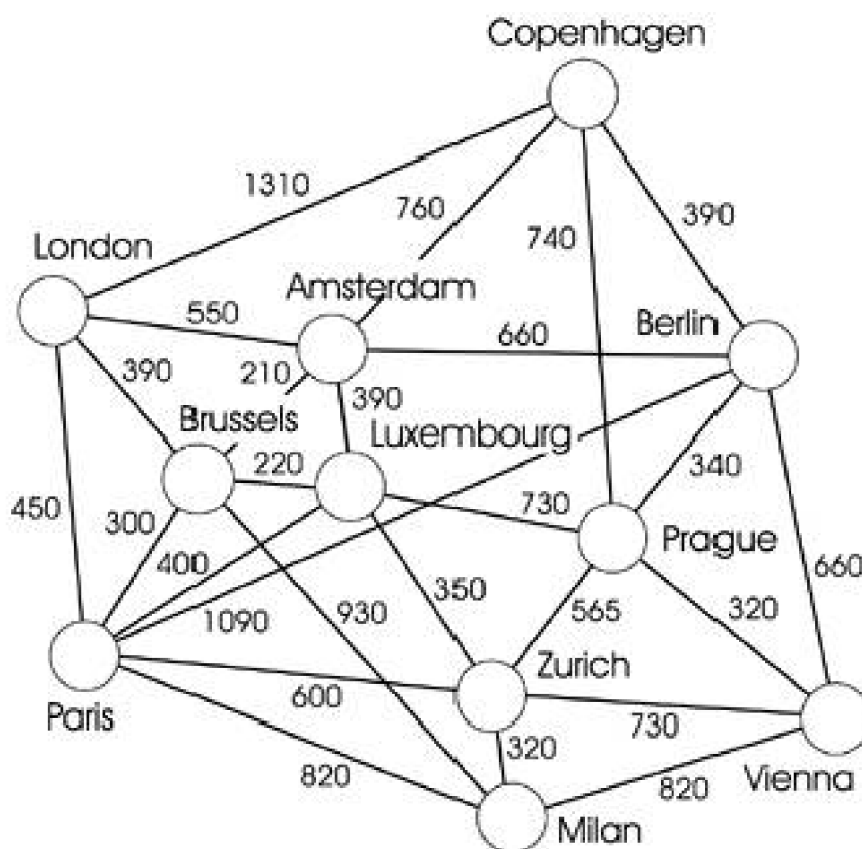
One clarification is that PC-1 is also a formulation that uses already placed capacities, but it had a special role only as an early experimental model to see how close to 100% restorability a set of p -cycles could come specifically under the spare capacities only of a span-restorable mesh SCA solution. The models that have span capacity limits added reflect circumstances where limited number of fibers are equipped for DWDM operation, each supporting up to $|K|$ wavelength channels. In those models the goal is to minimize additional cost of channels turned-up for working or spare on these fibers (assuming the common cost to equip the fibers with DWDM infrastructure is common to all alternatives). The aim, unlike PC-1 in those models is not to maximize restorability. The other models will generate any number of channels on any span as consistent with 100% restorability at globally minimum channel cost, with no per-span capacity limits. Note in that regard that nothing in the models with per-span capacity limits guarantees the feasibility of 100% restorability. These models apply if DWDM-equipped fiber-system counts are limited and it is assumed that enough channel capacity is potentially available if needed on the fibers to fully support restorability plus routability of working demands in the joint models. If this is not the case, it will be observed in the form of an infeasibility report from AMPL/CPLEX.

The WP-PC-1 and WP-PC-3 models have an explicit cost per additional whole fiber system required on each span in addition to per-channel costs. This is because when wavelength assignment is involved, with no fixed capacities as inputs, it is necessary that the optimization consider per-fiber costs as well. Otherwise, if only channel costs mattered, the wavelength assignment aspect of the solution could opt to having one fiber per required wavelength on each span. The per-fiber cost in the objective forces the solution to assign wavelengths as intelligently as possible from the fewest number of fibers per span.

10.5 Schupke et al. — Case Study for DWDM p -Cycles

Schupke, Gruber, and Autenrieth conducted a study on WDM p -cycle network design [ScGr02] that was also the first independent validation of the original claims [GrSt98] [GrSt00] about p -cycle efficiency. Their study network was the pan-European COST 239 network which is reproduced in Figure 10-13 with 11 nodes and 26 spans. The network has a high average nodal degree $\bar{d} \approx 4.7$ and every node has at least degree 4. Based on the lower bound of $1/(\bar{d} - 1)$, we can predict that design efficiencies as low as ~30% redundancy may be approachable for this network. Two demand patterns were employed in test cases. `demand_1` is shown in Figure 10-13 and reflects a distributed demand pattern with traffic intensities given in Gb/s. These quantities were divided by 2.5 Gb/s and interpreted as lightpath demands yielding a total 348 of demands in the unscaled `demand_1` pattern. `demand_2` is a centralized pattern where each node has a demand to/from an arbitrarily nominated hub node rather centrally located in the topology (node 7).

Figure 10-13. COST-239 Study Network Model. Demand in Gb/s. Distances in km. (source [Grub01]).



		0	1	2	3	4	5	6	7	8	9	10
Paris	0		12.5	15	2.5	5	27.5	12.5	2.5	17.5	25	2.5
Milan	1	12.5		15	2.5	7.5	22.5	5	2.5	5	7.5	2.5
Zürich	2	15	15		2.5	7.5	27.5	7.5	2.5	7.5	7.5	2.5
Prague	3	2.5	2.5	2.5		2.5	5	2.5	2.5	2.5	2.5	2.5
Vienna	4	5	7.5	7.5	2.5		22.5	2.5	2.5	2.5	5	2.5
Berlin	5	27.5	22.5	27.5	5	22.5		20	5	15	20	7.5

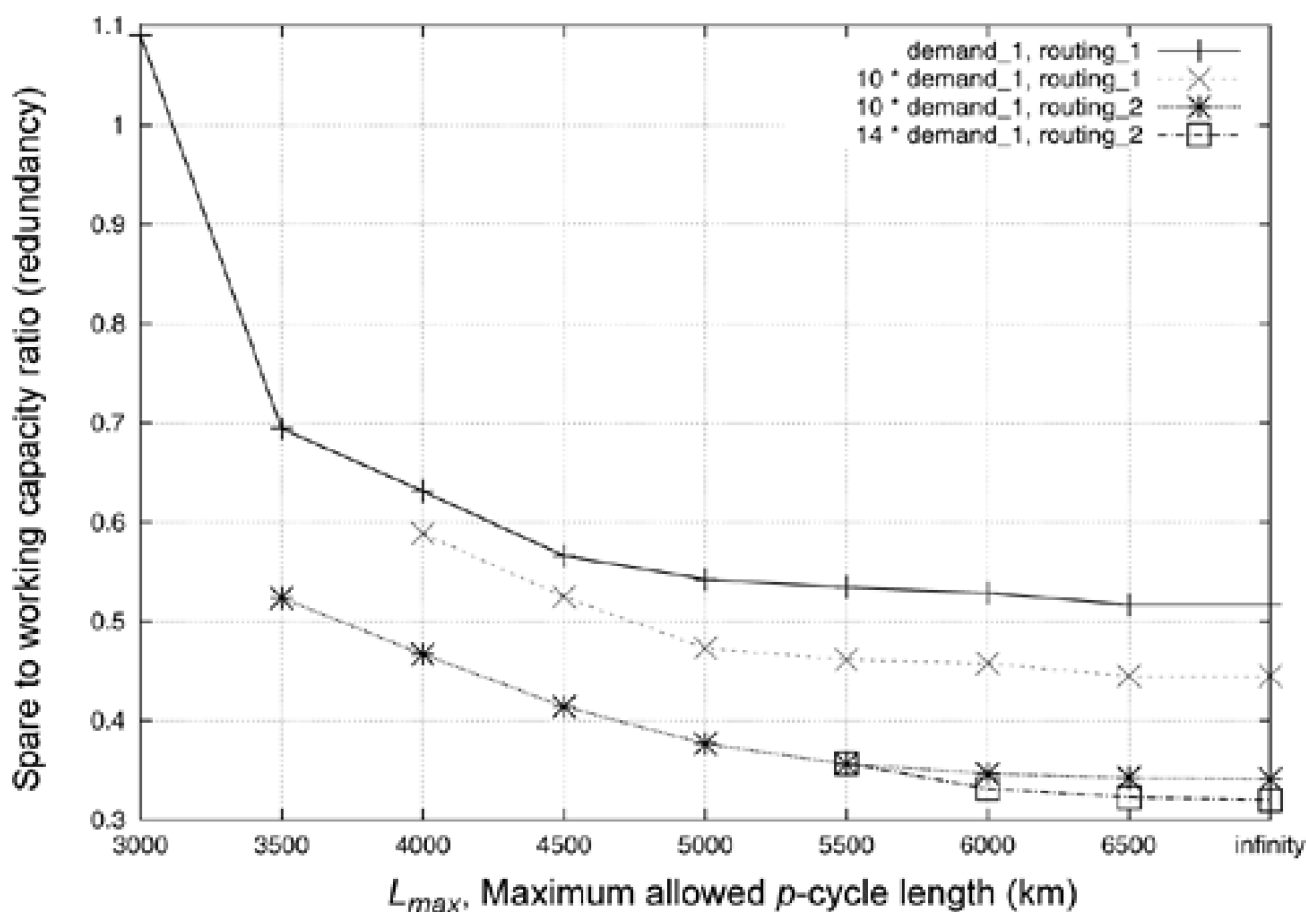
Amsterdam	6	12.5	5	7.5	2.5	2.5	20		2.5	10	12.5	2.5
Luxembourg	7	2.5	2.5	2.5	2.5	2.5	5	2.5		2.5	2.5	2.5
Brussels	8	15	5	15	2.5	2.5	15	10	2.5		10	2.5
London	9	25	7.5	7.5	2.5	5	20	12.5	2.5	10		2.5
Copenhagen	10	2.5	2.5	2.5	2.5	2.5	7.5	2.5	2.5	2.5	2.5	

In each case, two types of demand routing are considered. Under *routing_1* all demands follow shortest hop routes. Under *routing_2*, demands are shortest path routed with the hop metric being the reciprocal to the unused capacity of the link, which is updated each time a demand has been routed. The idea under *routing_2* is to try to balance load on the links, while still achieving a low overall use of link resources. The set P includes all distinct elementary cycles up to a circumferential length not longer than L_{max} .

10.5.1 VWP Results

The VWP case used formulation PC-2A (Section 10.3.4) where the additional span capacity constraint was used to assert that each span of the test network is considered to have two fiber links each supporting 128 wavelengths and that all required p -cycles must be formed within the I capacity remaining after working path routing over the two fiber links per span. Figure 10-14 shows the redundancy (spare to working ratio) for the VWP case versus the allowed maximum length of a p -cycle. At $L_{max} = 4500$ km or more, attractively low redundancy (< 60%) is achieved. The efficiency improves even further when more of the demand is uniformly scaled by 10 of the pattern. This is attributed to the greater number of working paths being able to use the same p -cycle infrastructure that was established for the lighter demand, i.e., $x_{ij} = 2$ potentials are fully realized and more on-cycle routing relationships are also picked up.

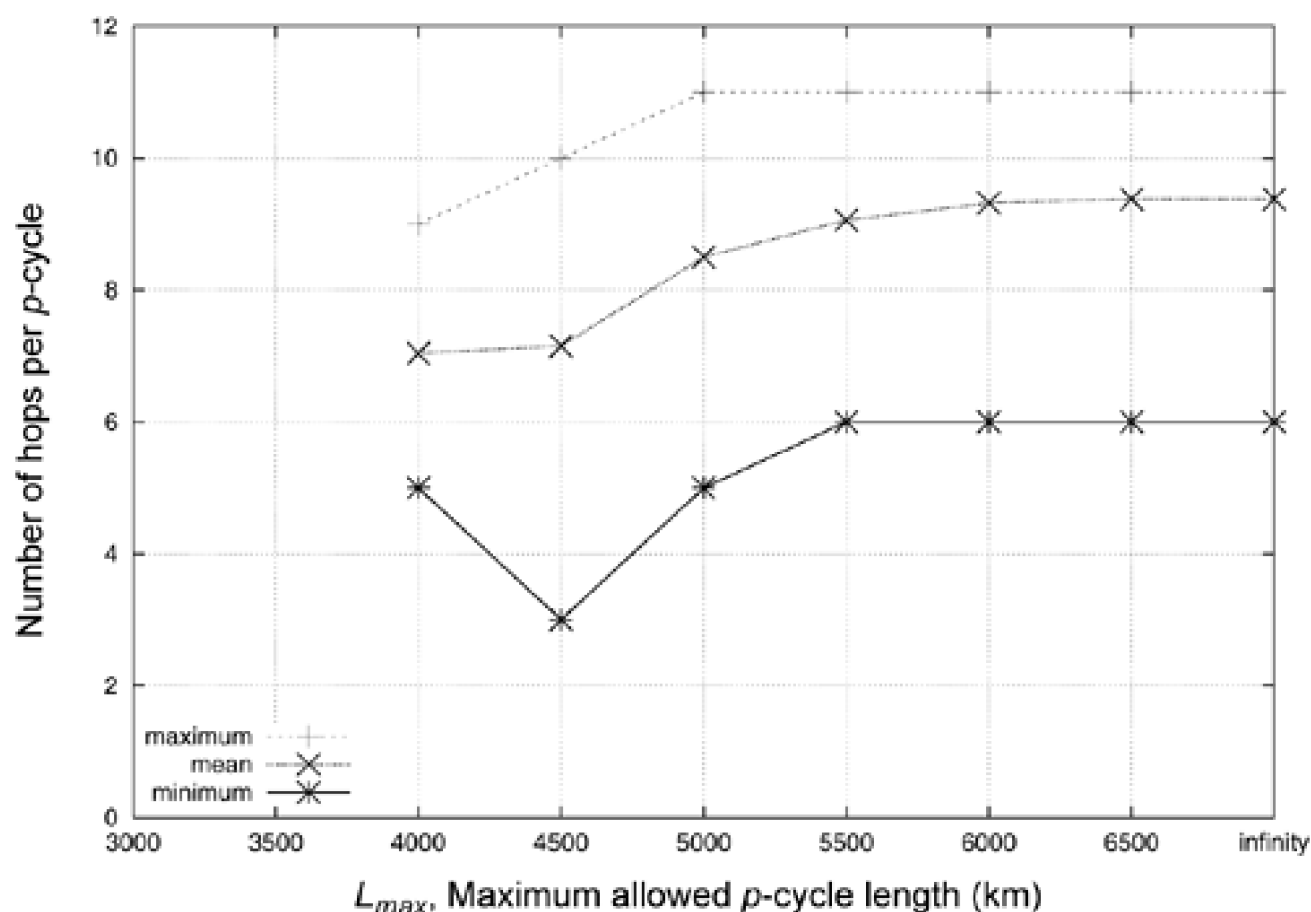
Figure 10-14. Redundancy of fully restorable VWP p -cycle designs versus the maximum allowable p -cycle length, L_{max} , in the COST 239 study network (from [ScGr02]).



It is interesting to examine some overall conditions when the limiting efficiency is asymptotically achieved for **demand_1** at a scaling of about 14 and $L_{max} \approx \infty$. The working demands are then utilizing 61% of the potential total network capacity (two bidirectional fibers/span @128 l) and the protection capacity takes only another 22% of the total potential capacity. And the ratio of spare to working capacity is indeed just slightly above the 27% lower limit predicted by $1/(\bar{d} - 1)$ for this network ($\bar{d} = 4.7$). This confirms the high efficiency of p -cycles as originally reported [GrSt98]. Also, regarding the $1/(\bar{d} - 1)$ bound, note that it is with **routing_2**, (which tends to level out the w_i quantities) that the bound is nearly reached, not with **routing_1** under the same demand pattern. This is consistent with the insights developed in Chapter 5 (Section 5.4.3) as to how capacity-balance affects the potential to approach the lower bound. Also understandable from the same standpoint is that we do not see as high efficiency under the hubbed demand pattern (**demand_2**). Even with $L_{max} \sim 4500$ km the redundancy remains 84%. Similar to line-switched rings, p -cycles are less able to support centralized traffic patterns at the same high efficiencies as with a more distributed demand pattern.

Also of interest is the number of logical hops and the corresponding number of p -cycle nodes used in the designs. Because span lengths vary, this is in some ways a better metric by which to judge how large are the p -cycles that are being formed. Figure 10-15 shows the minimum, mean and maximum number of p -cycle nodes (or logical hops) versus the geographical length limit L_{max} . The mean curve is noticeably nearer to the maximum curve, suggesting that there is a tendency to choose cycles with a higher number of nodes. This is consistent with the theoretical investigations of [StGr00b] and the intuition that as p -cycle increases in size it accumulates potential straddling relationships as $O(n^2)$. These observations set the stage for further consideration of these principles in Section 10.7.1.

Figure 10-15. Number of logical hops on VWP p -cycles versus L_{max} (from [ScGr02]).

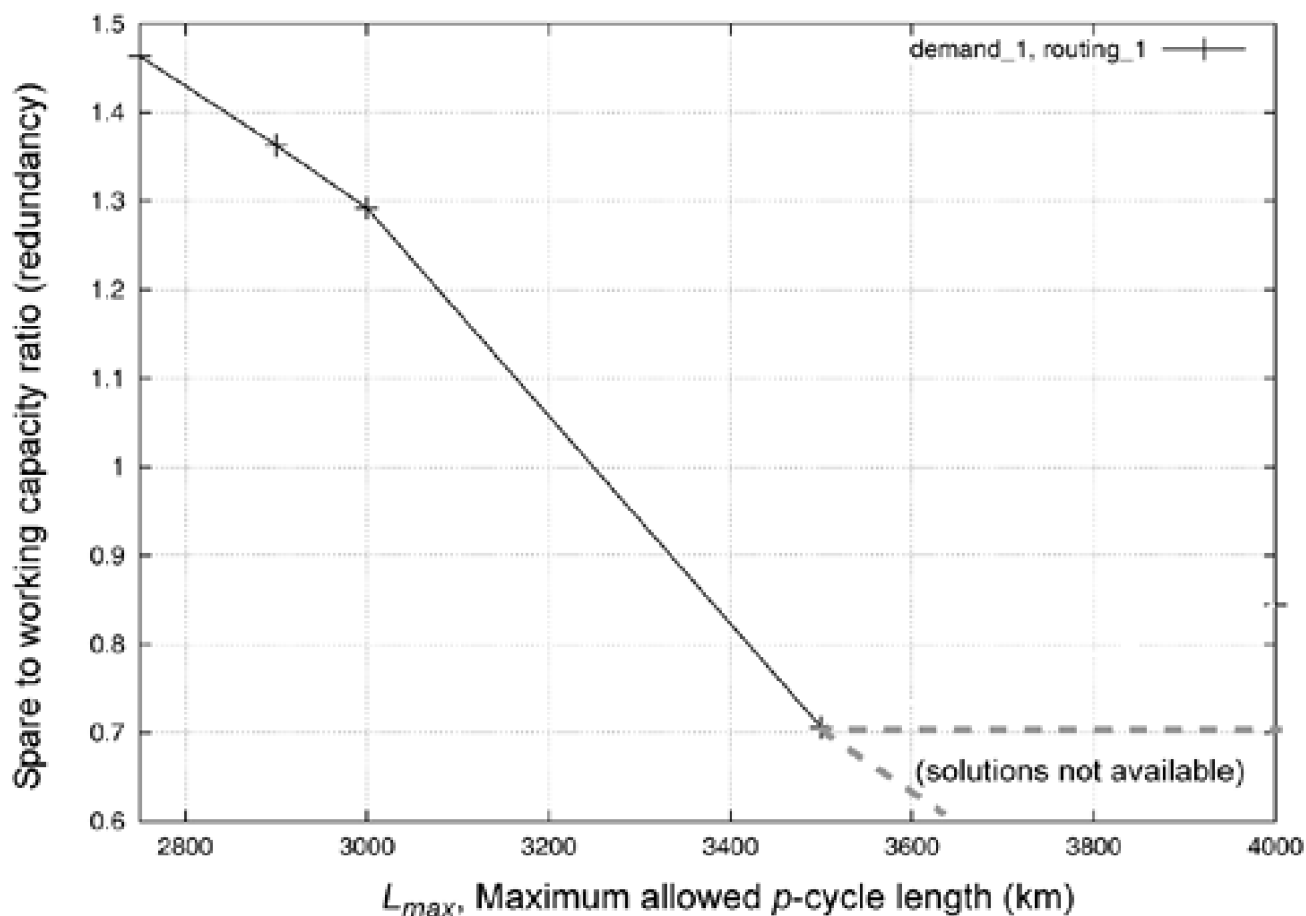


WP Results

Results for the WP case are understandably more difficult to obtain. The available results are therefore limited to **demand_1** under

routing_1. Figure 10-16 shows the redundancy versus L_{max} . These solutions are also limited by a memory consumption of 300 MB that was reached before optimal termination, so there is no guarantee of optimality of the solutions and the curve is not monotonously decreasing. Despite these limitations reasonable design results can be found for L_{max} up to about 3500 km.

Figure 10-16. p -Cycle redundancy versus L_{max} for WP designs (from [ScGr02]).

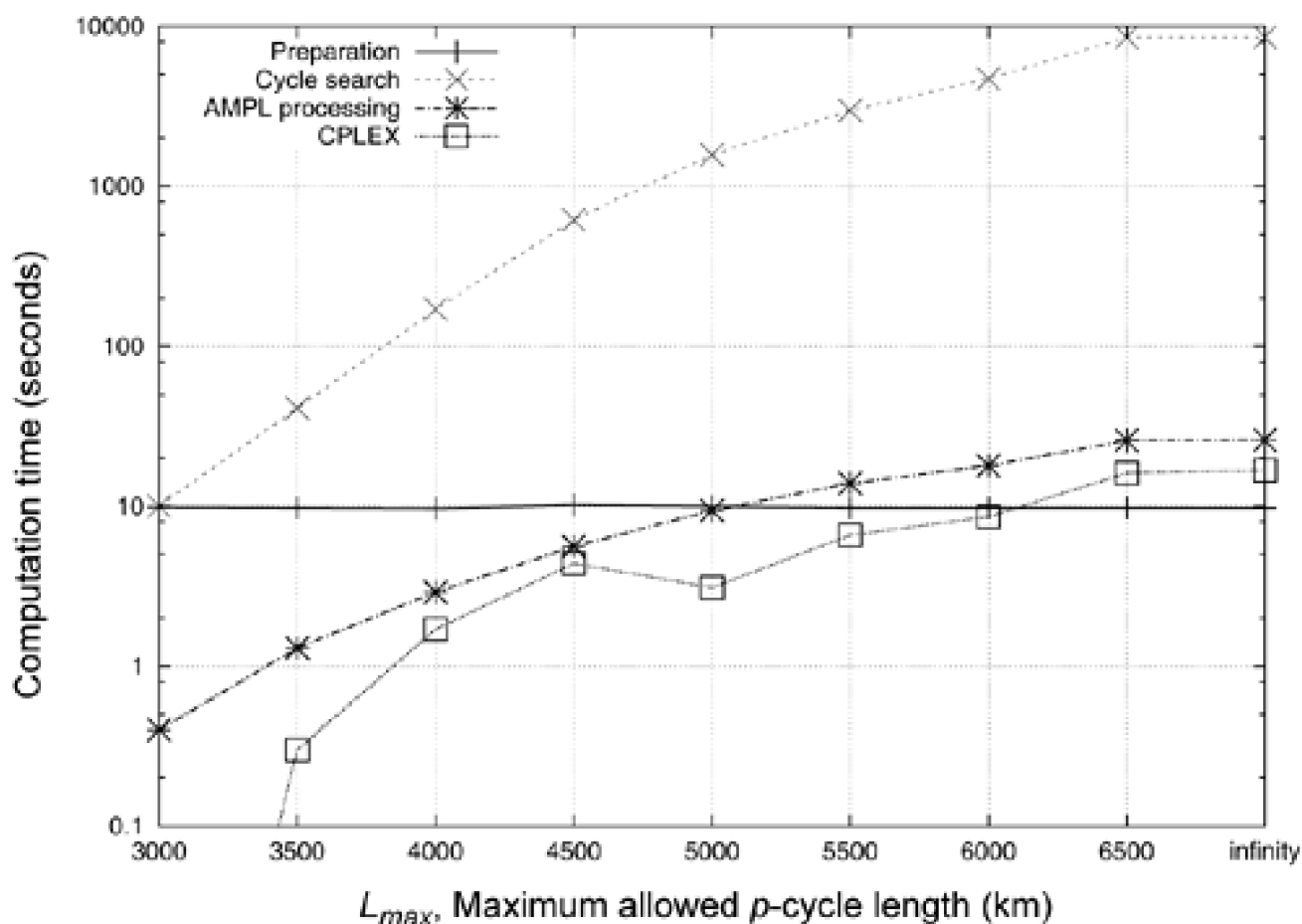


10.5.2 Computation Times

Schupke et al. [ScGr02] also studied the computational time breakdown of their series of VWP results. They found the cycle search to construct P to take the biggest portion of time, although this function only has to be performed once for any number of studies or design updates on a given topology and is also amenable to application of faster cycle-finding algorithms than used in that study. Figure 10-17 shows the computation times for the cycle search, for AMPL processing (data generation and processing), and for the solver CPLEX in the VWP case with 10* demand_1 under routing_1. The times for the other computations are negligible. When using routing_2 instead of routing_1 the routing computation time becomes slightly longer. In either case, CPLEX ILP computation time is surprisingly short. The total computation times are thus acceptable (mostly much lower than three hours) for the VWP case. In the WP case the total calculation time lasted approximately two hours and stopped if CPLEX reached a memory maximum of 300 MB. Therefore the solver time was always the dominant part. This can become a critical factor for the configuration of WP networks, although a number of avenues are being developed to attack this problem. Primarily, tactics of waveband bundling or other logical scaling-down of the number of individual lightpaths on separate Is may ease the computational difficulty. Here in the 10* scaled WP design problem, each demand was being handled individually as was every p -cycle being solved for at the individual-assignment level. Various relaxations of internal flow variables and, in particular, preprocessing steps can be devised that reduce $|P|$ while populating it with only "high-promise" p -cycle candidates, based on the ideas for algorithmic generation of p -cycle designs in later sections.

Figure 10-17. Computation times for cycle set enumeration, AMPL processing and CPLEX

solution for VWP designs (from [ShGr02]).



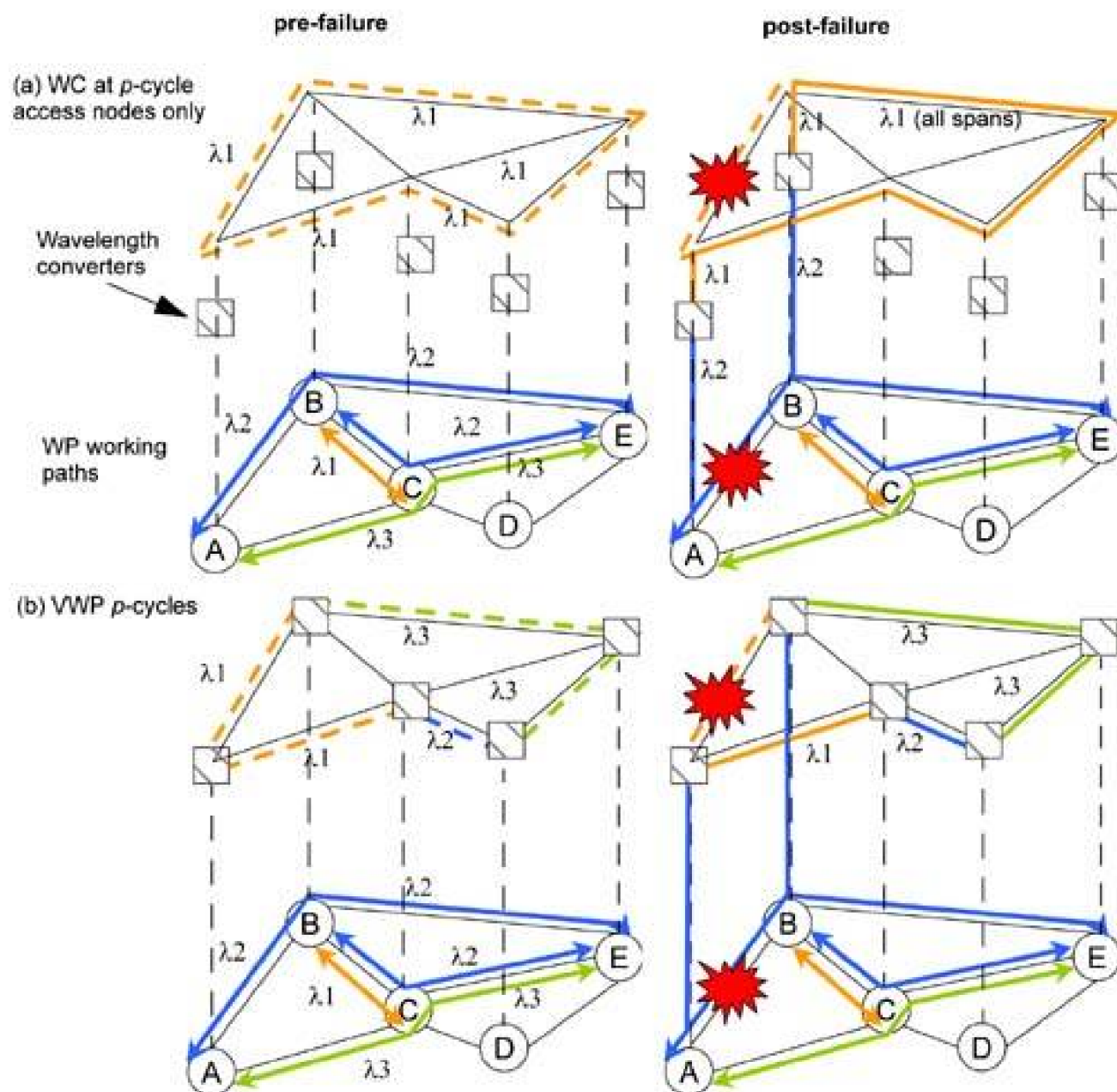
10.5.3 Placing Wavelength Converters at the p -Cycle Access Points Only

Schupke's study, and the design models above, both consider only the extremes of pure VWP and pure WP DWDM networks. We have said several times in earlier chapters, however, that the most economic and practical type of optical network may be one where there is "partial" or limited wavelength conversion capability. In such a network most paths are WP in nature, based on transparent optical OXC switch fabrics (avoiding the cost of o-e-o conversion) but there are "just enough" opportunities for wavelength conversion at each node so that I-blocking can be made negligible. Such networks reap most of the cost savings of optical transparency but from a design and operations standpoint are essentially as simple as VWP networks. In this regard, an interesting idea to arise in Schupke's study was that—in planning DWDM p -cycle networks—one might let the entire working layer be WP in nature and associate wavelength converters *only* with the p -cycles that protect these WP paths—in other words a combination of VWP- p -cycles to protect WP working paths. Several variants of this basic idea—called the "WP-working, VWP- p -cycle concept"—were recently investigated in [ScSc03].

The OXC node architecture for consideration of this strategy is that of Figure 2-8. WP working paths are provisioned through the local access and transparent optical switch core. The converter pool is used if needed either to avoid I-blocking on the working paths, or for pre-planned access to p -cycles upon a failure. There are two architectural options to consider, as illustrated in Figure 10-18. In (a) the p -cycles themselves are actually WP structures (as well the working layer) and wavelength conversion is used only at the p -cycle access points to map the failed working path I to the p -cycle I for rerouting. Working paths on the same p -cycle can also directly access it. In the second approach, (b), the p -cycles are themselves completely VWP structures. Here the converters are notionally "in series" with the p -cycle at every node and can interface any working path into p -cycle for protection and provide wavelength-agility along the p -cycle to permit p -cycle formation out of any available I-channels on a span. In (b) (the VWP- p -cycles), we may require more converters in total if the same number of p -cycles are used, but by designing the VWP- p -cycles we tend in fact to require fewer p -cycles in total, as well as less spare capacity. The exact cost trade-off between these two depends in detail on the relative costs of I-conversion compared to I-channels themselves, and on the graph topology and demand pattern, i.e., as to whether efficiently shared WP p -cycles can be formed on demands present, or if VWP is needed to enhance the achievable sharing of protection capacity. However, both of these "partial I-conversion"

strategies are much more cost-effective than an entire corresponding VWP network when I-conversion is a significant cost. They are also attractive in that they provide a systematic way of planning the I-conversion for a network.

Figure 10-18. Two strategies for efficiently associating wavelength converters with p -cycles—protection an all-optical working path layer with a minimum of wavelength conversion.



Another elegant aspect is that a general issue in optical networks is that protection re-routing may unpredictably affect transmission path quality and BER—regeneration may be needed in a protected path that was not needed in the normal working path. (This is especially of concern for "all-optical" SBPP.) But transmission quality is inherently safeguarded in either of these schemes as long as the I-conversion involves o-e-o. In the VWP case this is clear because each p -cycle node then employs regeneration. In the WP p -cycle case it is also addressed, however, as long as the WP path segment between each pair of access points on the p -cycle is properly characterized and designed in advance. If choosing between strategies, the VWP p -cycles option would have added practical benefit of being more easily reconfigurable. As demand patterns evolve the corresponding set of protecting p -cycles can be easily changed without any impact on services because these are only structures formed in the standby spare capacity. Space limits further discussion of this approach, but see the book's web site for a supplement.

10.6 Results with Jointly Optimized (VWP) p -Cycles

Schupke's study on the COST 239 network has shown us a lot about how p -cycle designs perform in the VWP and WP case, but something that is not known yet is how much *joint* optimization affects p -cycle designs. To consider this question we used the PC-3 model ([Section 10.3.6](#)) for the joint VWP case to run a number of comparisons. Results comparing the joint versus non-joint p -cycle designs are summarized in [Table 10-6](#) for three random test networks of 32 nodes and the COST 239 topology. One of the 32 node networks has 51 spans (denoted 32n51s) and is used as a master network from which sparser 32-node test cases are obtained by random elimination of nonessential spans to create lower average nodal degree. These are referred to as the "32n51s family" of networks. The demand matrix for the 32-node networks was generated from the product of end-nodes' degrees (giving a distance independent mutual-attraction demand pattern characteristic—see [Section 1.5.5](#)). The span lengths are assigned based on the Euclidean distances between nodes. For the non-joint designs 10,000 eligible p -cycles were provided based on the *preselection heuristic* that follows in [Section 10.7.4](#) and the PC-2 design model was used. These problems were solved to within 0.2% of optimal. For the jointly optimized designs, the PC-3 model was used with the same set of 10,000 eligible p -cycle candidates and the five shortest routes per end-node pair were provided as the set of eligible routes for working path assignment. These results were solved to within 0.5% of optimal.

For COST-239 the non-joint designs employed 500 eligible p -cycles chosen by the same preselection heuristic and solved within 0.1% of optimal. The demands in the COST-239 case are a uniform random number of wavelength path requirements between 1 and 10 (inclusive) on each O-D pair. For the joint designs the 10 shortest (by distance) eligible working routes between each end-node pair were provided. In all of the non-joint problems, demands were routed via shortest (distance) paths prior to solving for the p -cycle set that gives 100% span protection with minimum spare capacity. There was no length limit on the candidate p -cycles that were allowed. All these problems were solved in a few minutes.

[Table 10-6](#) identifies each network and shows its average nodal degree. Then the non-joint designs are summarized in terms of the total working and spare capacity (in channel x distance units), the corresponding redundancy ("red.") and the number of distinct cycles on which p -cycles were formed. The joint design columns are the same except that the *standard* redundancy measure is used (because the working path lengths may increase in the joint designs). Recall the standard redundancy is the ratio of the spare capacity plus any excess working capacity arising from routing that is above shortest paths in length, to the working capacity total from shortest path routing (See [Section 1.5.3](#)). The total working capacity required for shortest path routing is available from the non-joint designs.

Table 10-6. Results Comparing Joint and Non-Joint p -Cycle Designs

Net	\bar{d}	Non-joint				Joint			
		total work.	total spare	red.	# cyc.	total work.	total spare	(std.) red.	# cyc
32n 51s	3.18	394,877	352,559	89.3%	16	405,539	246,943	65%	28
32n 47s	2.94	414,984	347,285	90.2%	36	424,267	300,400	74.6%	33
32n 44s	2.75	423,596	403,982	95.4%	30	431,853	341,269	82.5%	23
COST-239	4.72	137,170	81,790	59.6%	7	143,745	46,910	39%	4

The significant finding is that joint optimization leads to improvements of 20 to 25% in capacity efficiency. The redundancy of 39% in COST-239 indicates extremely high efficiency, in the order of the best one could expect from path restoration on many other networks, albeit of typically somewhat lower degree. The number of distinct cycles employed in the COST-239 designs is about the same number of rings as would be expected in a ring-based design for that network, if not fewer. The other numbers of distinct cycles also do not imply an unmanageable number of logical p -cycles. Also quite evident in the results is the effect of nodal degree. In the sparsest of the 32 node trials the non-joint redundancy was approaching 100% but improved to a much more reasonable 82.5% under joint design, which is not unreasonable for a \bar{d} of only 2.75. Each design had an efficiency that ranked directly as would be predicted by the \bar{d} sequence. And

again, at the high degree of 4.7, COST-239 shows the superbly low redundancy of 39% under joint optimization.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

10.7 Heuristic and Algorithmic Approaches to p -Cycle Design

So far we have relied on MIP optimization models to formally introduce p -cycle concepts and to design actual p -cycle based networks of various types. Such models are surprisingly often quite manageable with commercial solvers. In other cases, they can be virtually intractable even at relatively small sizes when large numbers of pure binary decision variables are involved, such as the choice of a working path route in joint formulations, or, in assigning a specific wavelength to each demand or p -cycle. This section is therefore devoted to looking at algorithmic or other heuristic approaches for solving p -cycle design problems.

10.7.1 p -Cycle Efficiency Metrics

It turns out that for p -cycles, there is a highly relevant figure of merit that can be the basis for several constructive algorithms and a MIP-based preselection heuristic. The metric is strongly diagnostic of how efficient a particular cycle might be in a p -cycle design, if the opportunity exists in terms of demand flows to use it to its full potential. This is analogous to somehow knowing ahead of time how useful a given unit of spare capacity might be in an SCA design depending on which eligible route it was allocated to. But there is no such effective *a priori* principle for other mesh architectures as there is for p -cycles. There are actually two closely related metrics we want to define:

The basic *topological score* (TS) of a candidate p -cycle is defined as:

Equation 10.35

$$TS(j) \equiv \sum_{i \in S} x_{ij}$$

where j denotes the a candidate p -cycle in the set of all cycles P , and as described before, $x_{ij} = 0, 1$, or 2 depending on the topological relationship of the cycle j to the network. Thus, $TS(j)$ simply measures the total number of protection relationships that the cycle j could provide. The topological score takes no account of the resources that may be required to build the corresponding p -cycle on cycle j , nor (at this stage) whether the demand flows are available to fully realize the maximum potential for protection relationships that $TS(j)$ indicates. Obviously bigger cycles tend to have higher TS values. In fact it is easily shown that if a Hamiltonian cycle exists in a network, it maximizes TS at a score of $TS = N + 2(S - N)$ in a network of N nodes and S spans.

The potential efficiency of using a particular cycle as a p -cycle is measured by the *a priori efficiency metric* ($AE(j)$):

Equation 10.36

$$AE(j) \equiv \frac{\sum_{i \in S} x_{ij}}{\sum_{k \in S} \delta_{k,j} \cdot c_k} = \frac{TS(j)}{\sum_{k \in S} \delta_{k,j} \cdot c_k}$$

where $d_{k,j}=1$ if span k is part of cycle j (zero otherwise) and c_k is the cost of building the part of a unit capacity p -cycle that traverses span k . More generally the denominator is any measure of the cost of building the p -cycle. Thus, the basic idea is that a high AE metric indicates a candidate p -cycle with many potential on-cycle and/or straddling spans relative to its own cost (or length), that is, one that offers high protection capacity efficiency. Note, importantly that in TS and AE metrics, straddlers have double the weight of on-cycle spans in contributing to the topological score. In prior literature, and informally here, the AE metric may at times be referred to simply as the score of a p -cycle candidate.

As stated, both TS and AE are purely topological assessment of *potential* protection merit. Whether the potential is fully realized or not depends on whether the corresponding w_i quantities are present or not. Depending on the context for use of the demand-weighted efficiency, the relevant w_i quantities may be either the total span working capacities to be covered or, in iterative constructive algorithmic contexts such as will follow, w_i may be only a remaining uncovered amount of working capacity. Thus the *demand weighted* efficiency of p -cycle candidate j is defined as $DWE(j)$:

Equation 10.37

$$DWE(j) = \frac{\sum_{i \in S} \min(x_{ij}, w_i)}{\sum_{k \in S} \delta_{k,j} \cdot c_k}$$

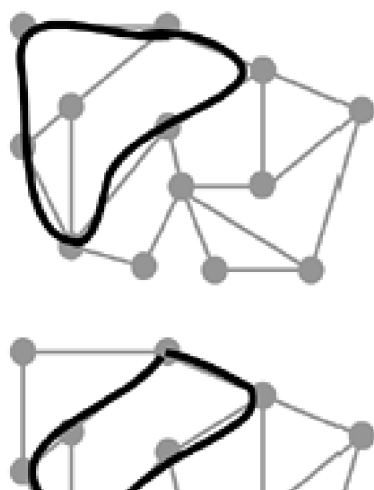
and the *distance - demand weighted* efficiency of a p -cycle is correspondingly defined as:

Equation 10.38

$$DDWE(j) = \frac{\sum_{i \in S} \min(x_{ij}, w_i) \cdot l_i}{\sum_{k \in S} \delta_{k,j} \cdot c_k}$$

where l_i is the length or cost of span i that is deriving protection from candidate p -cycle j . While TS and AE are *a priori* measures of potential protection scope and potential efficiency (which do not depend on the actual association with working capacity in a given design), DWE and $DDWE$ measure the *actual* efficiency of a p -cycle in the context of the straddling and on-cycle working channels that are actually available for it to protect. In this regard they are not measures of *potential* efficiency as TS and AE are, but actual *realized* efficiencies that can only be assessed based on the working channels actually assigned to each p -cycle within a solver or other planning process. [Figure 10-19](#) gives three examples of some would-be p -cycles and how their AE metrics would be computed using just a hop count total as the measure of the p -cycle cost.

Figure 10-19. Identifying good p -cycle candidates with the *A Priori Efficiency (AE)*.

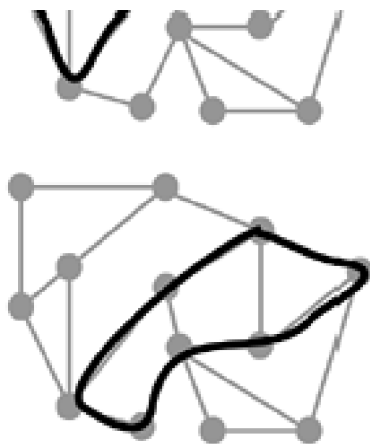


(a) AE metric =

$$\frac{6 \text{ on-cycle} \cdot (1) + 0 \text{ straddlers} \cdot (2)}{6 \text{ spans traversed}} = 1.0$$

(b) AE metric =

$$\frac{6 \text{ on-cycle} \cdot (1) + 1 \text{ straddlers} \cdot (2)}{6 \text{ spans traversed}} = 1.33$$



(c) *AE* metric =

$$\frac{7 \text{ on-cycle} \cdot (1) + 2 \text{ straddlers} \cdot (2)}{7 \text{ spans traversed}} = 1.57$$

The examples make several general observations apparent:

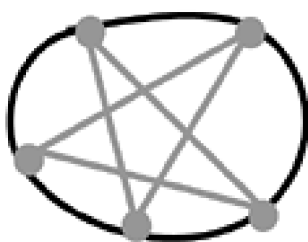
- Picking up an additional straddling span relationship is twice as good as picking up an extra on-cycle relationship.
- Sometimes straddling relationships are present in a sense, but involve more than a single span on the "straddling subnetwork." These are not being counted in the present assessment, but we show below how they could be taken advantage of.
- Non-simple cycles could pick up more straddling relationships than the simple cycles we are restricting ourselves to at present.
- The maximum score p -cycle would have a full mesh of spans between its N nodes. (N nodes of the p -cycle, not all nodes of the network).
- The *AE* score of a conventional ring of any type is 1.0

10.7.2 The "Perfect Score" for a p -Cycle

We have already observed in passing that the topological score for a ring is always unity. Any ring consisting of N hops protects against N on-cycle failures. p -Cycles do better by virtue of adding straddling span relationships. So what is the limiting efficiency for p -cycle? It is the score corresponding to a p -cycle of N hops having a full mesh of connectivity among all its nodes. The perfect score for p -cycle can thus be worked out accordingly, to be $(N-2)$ protection relationships per unit of its own capacity, as shown in [Figure 10-20](#).

Figure 10-20. The perfect p -cycle has $N-2$ protection relationships per span in the cycle.

p -cycle with a full mesh of straddling spans



- number of edges in a full mesh graph on N nodes = $(N \cdot (N - 1)) / 2$
- number of edges on-cycle in N -node p -cycle = N
- score =

$$\frac{N \text{ on-cycle} \cdot (1) + \left(\frac{N \cdot (N - 1)}{2} - N \right) \text{ straddlers} \cdot (2)}{N} = N - 2$$

Moreover, by the same kind of analysis, it is fairly easily shown that for a network with N nodes and S spans, if there exists a Hamiltonian cycle on the network then the *AE* of this cycle is the highest possible and is $\bar{d} - 1$, where \bar{d} is the average nodal degree of the network, i.e., $\bar{d} = 2S/N$. Of course an efficiency of $(\bar{d} - 1)$ corresponds exactly to the $1/(\bar{d} - 1)$ redundancy lower bound. The specific relationship

between p -cycles and Hamiltonians is developed further in [Section 10.10.2](#).

Let us now discuss how the concept of p -cycle score can be used in algorithms and heuristics for p -cycle designs. In this context we initially assume that demands are already routed, generating span w_j quantities to be protected, and that our aim is to place a corresponding set of p -cycles that give 100% restorability against any single span cut with minimum total spare capacity required to form p -cycles.

10.7.3 A Score-Based Design Assembly Algorithm

The first idea is an extremely simple algorithm based on two observations:

- It may take some time to generate the set of all distinct cycles up to a given circumference limit, but this only has to be done once for any number of p -cycle planning studies on a given topology. Moreover the topological k_{ij} scores can be worked out and stored for each cycle at the same time.
- Reevaluation of the *demand weighted* score can be an $O(n)$ operation on a suitable datastructure.

To convey the idea for this class of algorithms, let us define the linked-list data structure shown in [Figure 10-21](#). Other data structures could function equivalently but we use the linkedlist approach to facilitate discussion of the algorithmic ideas below. For each p -cycle in the master set P , the first variable-length field following the cycle number is the list of on-cycle spans of the cycle and their initial or current uncovered w_j values. This is followed by a variable length field listing all the straddling spans on this cycle and their initial or current uncovered w_j values. In practice, only the span names need be listed as pointers into a single current set of all the network w_j values. The basic design algorithm would use this data structure as follows:

1. Enumerate all cycles up to a circumference limit and build the data structure shown, assigning initial w_j values.
2. (Optionally): If P is too large, immediately throw out all cycles with AE scores $< (1-\epsilon)$ where ϵ is a threshold, such as 0.1 or 0.2, set either to reduce the total size of P to a target working size or simply to get rid of all cycles with p -cycle efficiencies hardly better than rings.
3. p -cycle design assembly:

For every current entry in the data structure: {

- compute current demand-weighted efficiency =

$$\left\{ \sum_{i \in \text{on-cycle list}} \min(1, w_i) + \sum_{i \in \text{straddler list}} \min(2, w_i) \right\} / |\text{on-cycle list}|$$

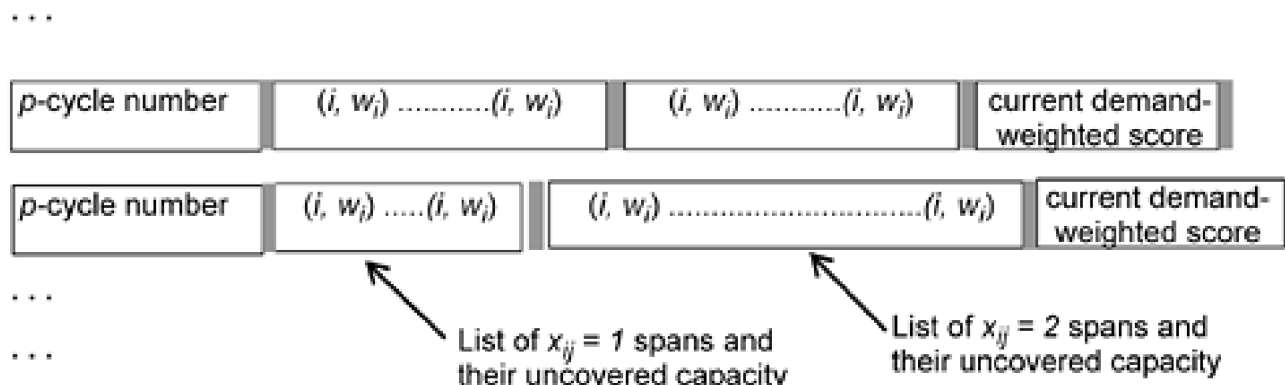
- if demand-weighted efficiency (DWE) $< (1+\epsilon)$, delete the record. }

4. Sort all remaining records (or simply scan the surviving list) to find the p -cycle number with the highest score.
5. Place the p -cycle found at (4.) into the design.
6. Update the uncovered w_j values:

$$\begin{aligned} & - k \leftarrow \text{winning cycle number} \\ & \forall i \in (\text{on-cycle list } k): (w_i \leftarrow (w_i - 1)) \\ & \forall i \in (\text{straddling list } k): (w_i \leftarrow (w_i - 2)) \end{aligned}$$

7. Until all $w_j = 0$

Figure 10-21. A linked-list data structure for set P on which fast p -cycle design assembly algorithms can operate.



Some initial results suggest that this procedure, as simple as it is, can be within 5% of optimal. An alternate stopping condition (other than all $w_i = 0$) may further improve its efficiency. The idea is to stop iterations at somewhat less than 100% restorability requirement or whenever the demand weighted score of all remaining candidates is below a level of unacceptable efficiency. The latter exit strategy would produce an almost complete design within which small number of remaining uncovered channels would be rerouted or otherwise dealt with by some secondary touch-up method. This avoids the last placement of a possibly quite poor p -cycle, strictly driven by the need to complete to 100% protection coverage. It draws on experience from ring-based network design with iterative ring placements. Rather than place a last ring in the design, if the last ring would be poorly loaded it is better to exit and adjust or pack the working demand routes to complete the design under the set of already placed systems. One drawback of this algorithm remains, however—the set of all distinct eligible cycles still needs to be generated as a pre-processing step. A later set of ideas, involving the combination and growth of "primary p -cycles," will be aimed at removing this requirement.

10.7.4 Preselection of Candidate p -Cycles: a Heuristic for MIP Solutions

The above iterative placement of a succession of individually good p -cycles is of course still not optimal because it is greedy. At each step a currently best-looking p -cycle is chosen. It does consider how the next p -cycle relates collectively to all those placed before it, by virtue of the evolution state of the uncovered w_j values, but there is no global exploration of how efficient the set of p -cycles is as a single collective combination. In this section we propose and test the idea of simply using AE ranking to preselect a limited number of high-merit p -cycles to be used in conjunction with the various MIP models of this chapter. The basic idea is that the MIP solution is attempted but it is given a significantly reduced set of "elite" p -cycle candidates. There may be for instance 100,000 distinct simple cycles in a network. The idea is to create a heuristic using the MIP solver but with only (say) 5,000 of the "most promising" p -cycle candidates to consider. The selection can be based on ranked TS or AE scores.

The basic framework is one within which many specific heuristic ideas can be tried, all having to do with defining the reduced set of elite cycles to consider. First, some experience with memory and run times may show, for example, that a budget of 10,000 cycles is realistic to work with. Then regardless of network size each problem is formulated with the budgeted number of cycles. The budget can be used up representing any number of mixed strategies for populating the elite P set. An example could be:

- Admit the 5,000 best cycles ranked by AE score.
- Add the 2,000 cycles with most absolute number of straddlers.
- Add the 2,000 of the longest cycles.
- Add 1,000 random cycles.
- Add all the cycles getting chosen from the iterative greedy algorithm above.

Not all the above subsets may be exclusive of each other but the duplicates can be either deleted, in which case less than the budgeted

number will actually be presented to the solver, or duplicates can be simply ignored in the problem file. (Although it is somewhat brute-force, the kind of redundant constraints that would arise from doing so are typically eliminated by AMPL and/or the CPLEX pre-solver in any case.) Some other things to note are that by itself the first set of cycles may not necessarily ensure feasibility. When choosing only individually elite cycles, there is no strict guarantee that a cycle will be represented that would cover, for example, a long degree-2 chain connected to an otherwise highly connected mesh. However, cycles in batch three above definitely cover that eventuality. Finally, although the example is for a total budget of 10,000 candidate cycles—which is accounted for in the first four batches—the number of cycles added by the last criteria is typified by the numbers of p -cycles seen in results such as in [Table 10-6](#)—that is to say a few tens of extra cycles at most.

Although certainly not as fast as the iterative best p -cycle choosing algorithm, this form of MIP-based heuristic should generally do better than the iterative technique, while still solving in a reasonable time. The key difference is that the MIP will still be choosing the p -cycle set as a collective whole, and it is reasonable that a collectively good combination of a handful of p -cycles should be likely to exist within the feedstock set of some thousands of individually meritorious candidates. In fact it is often found in combinatorial optimization problems that certain key building blocks appear frequently in the solutions. As long as the process through which we populate the elite set catches many of these key building blocks, the MIP solver will find them and assemble a set of p -cycles that collectively form a design of high quality.

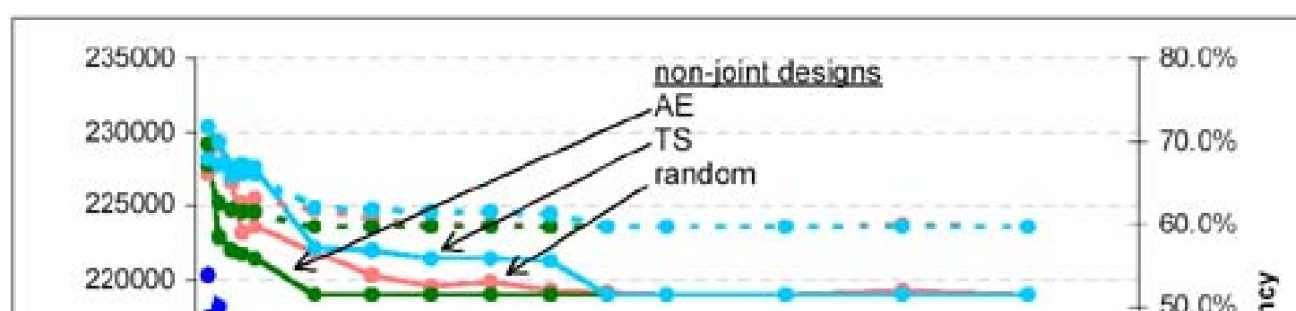
10.7.5 Results with Preselection to Solve the Joint p -Cycle Design Problem

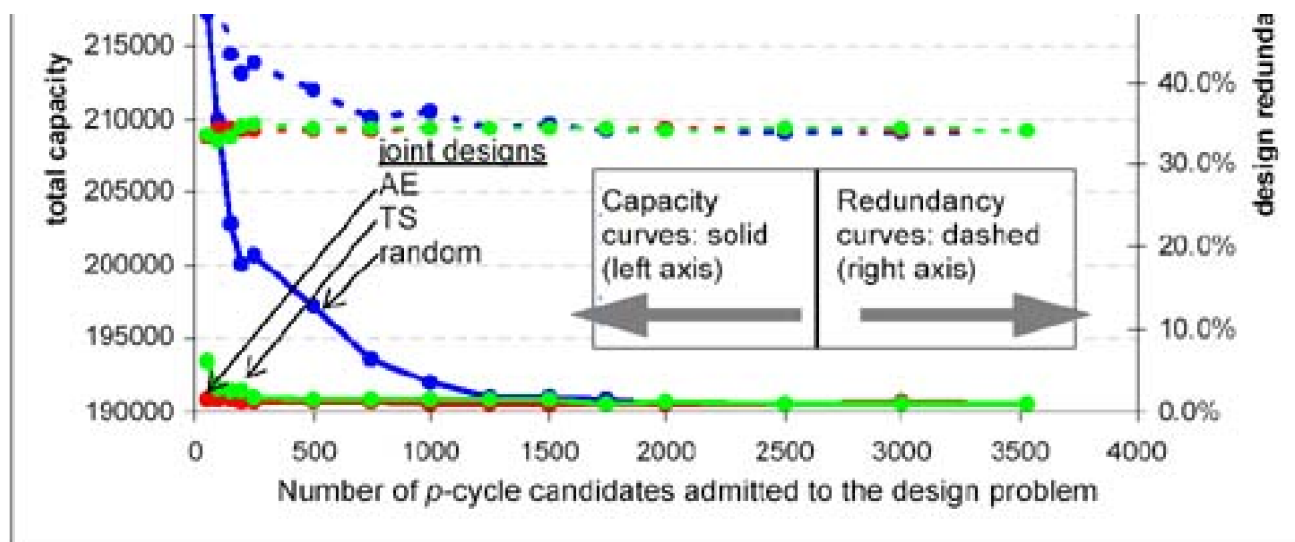
This section presents some test results showing the effectiveness of the idea of preselecting a reduced number of "high merit" p -cycle candidates to provide to an otherwise unchanged optimal solution model. In addition we present some interesting research data on the statistics of cycle AE measures themselves. The test case employed is the COST-239 network model with topology and demand data as given above. The preselection strategy was used to obtain the first reported results for *jointly* optimized p -cycle network designs, where the technique made a major difference to the ability to obtain essentially optimal joint designs [[GrDo02b](#)].

Selected results are summarized in [Figure 10-22](#) in terms of the total design capacity versus the number of eligible p -cycles admitted to the non-joint ("SCA") and joint ("JCA") design problems. There is a set of three curves for joint and non-joint designs respectively, indicating the capacity (and on the right hand axis, the corresponding redundancy) of designs produced with the given number of candidate p -cycles preselected by AE and TS measures. The same number of randomly chosen candidates is also used as a control. In the SCA p -cycle designs, demands are shortest path routed and the sum of the shortest-path working routes is used as the normalizer in all designs to measure redundancy. In JCA designs a minimum of 10 distinct eligible route choices are provided for each node pair. The unrestricted optimal design problem for this network involves the set of all simple cycles, which in this case is 3531 cycles, for both JCA or SCA. When given only the 250 top-ranked cycles to consider by the AE metric, the non-joint problem was solved to within 1.14% of optimal 857 times faster than when all 3531 cycles are represented. The JCA problem was solved to within 0.16% of optimal in 1/17th of the time if provided with only 50 cycles preselected by the AE criterion. In addition to showing how highly effective preselection by AE is, the results in [Figure 10-22](#) also show that:

- Joint optimization is of significant benefit, reducing redundancy by 25% over optimal non-joint designs. This is essentially the same as in span-restorable networks.
- The overall efficiency of joint p -cycle designs is extremely high (~34% redundancy).
- The AE preselection heuristic is superior to either TS or a randomly selected set of p -cycle candidates.

Figure 10-22. Solutions to joint and non-joint p -cycle designs using preselection of candidate cycles by TS and AE metrics.





An interesting effect to note, however, in the non-joint designs is that preselecting candidate p -cycles by the TS metric alone can apparently be worse than random. The reason is that TS has no concern at all about cycle size. The highest TS scores are therefore highly correlated with large cycles. Therefore when the preselection process removes all but the top-ranked cycles by this measure, only large cycles are left for the design. But a mixture of both large and small cycles is needed for a good design. Random selection avoids the bias toward exclusively large cycles and therefore does better. Notably, however, the joint designs are able to make better use of TS-selected candidate cycles than a random selection. This is attributable to the freedom to alter working routes to better exploit the bias toward large p -cycles under TS preselection. In both joint and non-joint cases, however, AE preselection performs the best.

The programs to identify and rank all p -cycle candidates by their TS and AE scores also allow us a first glimpse at some of the statistical characteristics of the set of all cycles in a graph when viewed as potential p -cycles. Figure 10-23 shows a histogram of the AE metric values from all cycles in COST 239 with an arbitrary example of a preselection threshold overlaid on the distribution showing how the vast majority of cycle candidates can be excluded from the design problem, keeping a relatively smaller number of high promise candidates. The units of AE values are the number of protection relationships per 1000 km of p -cycle spare capacity-distance in the COST 239 network. The histogram is well approximated by a scaled normal distribution with a mean value at 5.0 and a standard deviation of 1.2. Note the wide range of AE values exhibited by the 3531 various p -cycles within the network. This is why a limited but random selection of eligible p -cycles does not result in efficient or fast solutions. Preselection is effectively finding only a small number of high quality p -cycles at the far right of the curve. Note, however, that in networks with millions of cycles, it is not actually necessary to evaluate the entire population of cycles for ranking exactly as Figure 10-23 would suggest. Rather, a branch and bound type of depth-first type of search can dynamically reject low-merit cycle candidates without explicitly enumerating them all and simply build up an unbiased sample of high-merit candidates of the desired practical size for the design problem.

Figure 10-23. Histogram of the *a priori* efficiency of all simple cycles in the COST-239 graph.

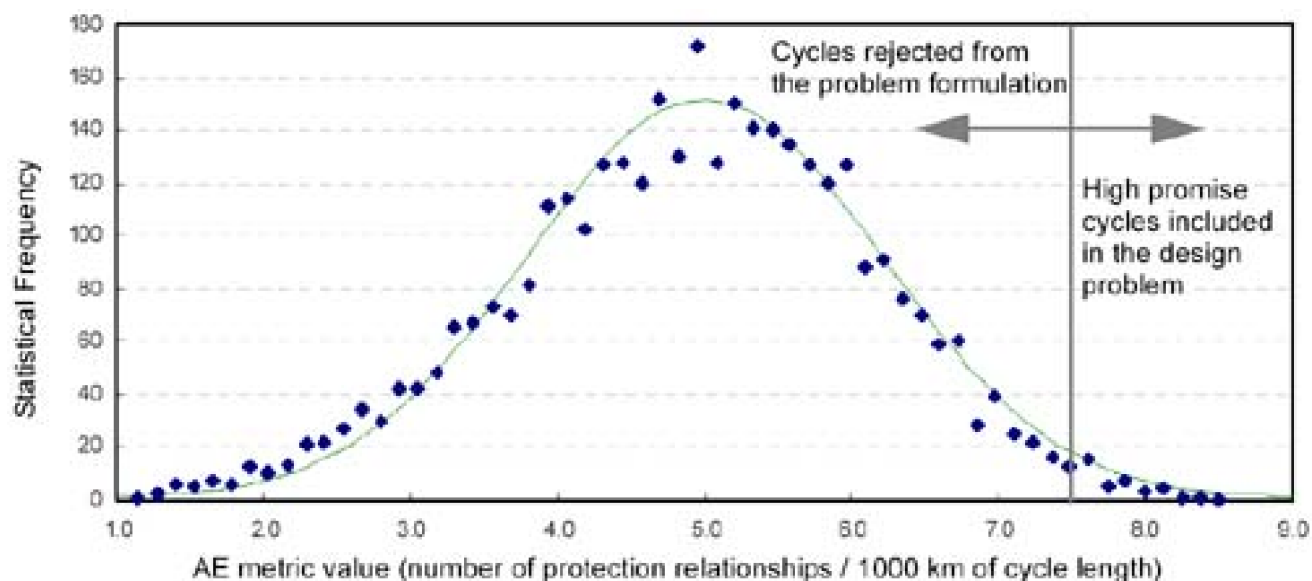
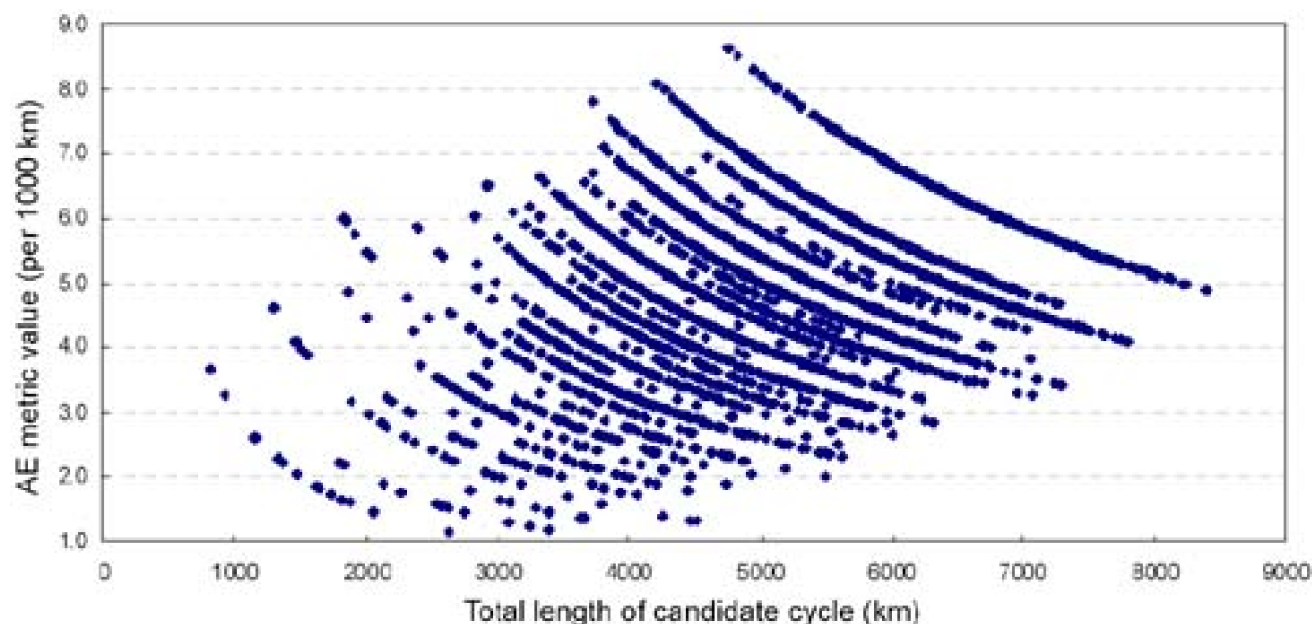


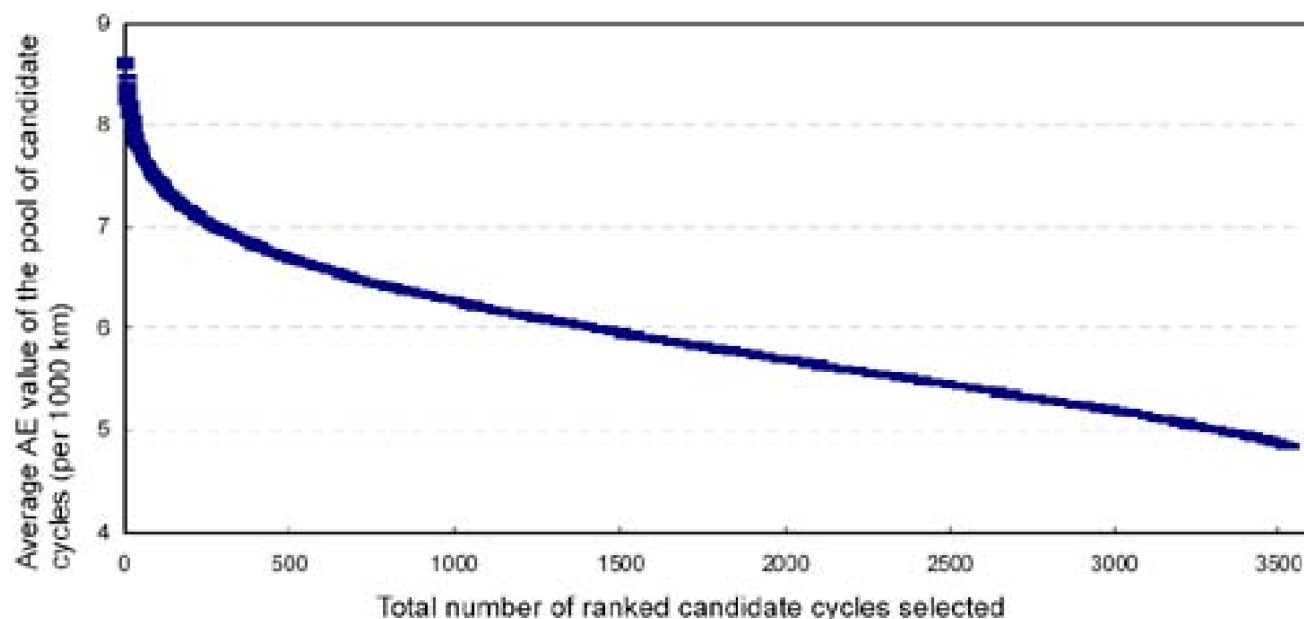
Figure 10-24 is a scatterplot of the individual AE metrics versus p -cycle length. The interesting structure of the scatter is explained by the fact that p -cycle TS values can take on discrete integer values only, but the total length of a cycle is more nearly a continuous variable. For any given TS value, the AE metric therefore follows a $1/x$ type of relationship with cycle length. This causes all p -cycles with the same TS value to follow the same gradient curve, producing a scatterplot with gradients or steps as observed in Figure 10-24.

Figure 10-24. Scatterplot of COST-239 p -cycle AE metric versus p -cycle length.



Finally Figure 10-25 provides a plot of the average AE score as a function of the number of ranked eligible p -cycles selected. To understand this data, the right-most point is the grand average AE value over all 3531 cycles (~4.8 protection relationships / 1000 km of p -cycle unit spare capacity). Conversely the left-most single value is the AE value of the single best p -cycle candidate in the entire COST 239 graph. Undoubtedly it is a Hamiltonian, which does exist on COST 239. All other values answer the question: If I ask for the N top-ranked cycles, what will the average AE metric be over those N cycles? The steep initial drop suggests where the most computational benefit will be obtained. If we say that the average AE value appears to be decreasing linearly after about 400 cycles, and that the first 200 cycles seem to be the elite ones as a group, we can expect that the greatest "bang for the buck" in terms of high solution quality for fastest run-time will probably arise from selecting N below about 400 on this network.

Figure 10-25. Average p -cycle AE metric versus number of preselected p -cycles in COST-239.



10.7.6 Zhang and Yang's Straddling Span Algorithm

In contrast to both MIP-based design and MIP-based design with the preselection heuristic, Zhang and Yang [ZhYa02] have explored an approach which has the main aim of avoiding the need for any cycle enumeration stage at all. The resulting algorithm is an extremely simple and fast procedure to produce a set of what can be considered "primary" or elemental p -cycles, having one straddling span each. In this section we look at the "straddling span algorithm" (SSA). In the following section we consider how the set of such primary cycles from SSA can be improved upon by merging operation(s) that evolve to a much more efficient design. The significance of the SSA to obtain primary p -cycles followed by evolution into a set of fewer efficient p -cycles is that the overall algorithm is highly scalable in that it is of low polynomial order complexity in network size and completely avoids any cycle enumeration stage. It also has the feature that the number of p -cycles considered never exceeds the number of spans in the network. With the SSA by itself 100% coverage of all spans is not strictly assured, although a simple inspection and repair phase can be applied to add any further needed p -cycles. Otherwise the algorithm is intended to quickly produce high-coverage initial designs for further consideration, such as in the merging operations that follow here.

The SSA is based on the observation that a p -cycle could be looked on as the combination of two node-disjoint paths between the two end nodes of a cycle-straddling span. This can only happen if the end nodes of a straddling span have a node degree of 3 or more. The algorithm assumes a biconnected graph, as is normal for protection feasibility. A fundamental part of the algorithm is to classify spans by their potential to obtain protection as straddling spans. In effect, an on-cycle protection relationship is only established when unavoidable due to topological considerations. [Figure 10-26](#) illustrates the SSA in three parts. In each case, the orientation is to consider the conditions of span A-B as a potential for a straddling span relationship. In [Figure 10-26\(a\)](#) both end nodes of the span are degree 3 or higher and two disjoint paths exist between nodes A-B excluding span A-B. This identifies span A-B as a kind of natural straddler for a future p -cycle. In (b) the two end nodes are of high enough degree to support A-B as a straddler, but the condition of two disjoint paths between nodes A-B excluding span A-B does not exist. Finally, in (c) one of the end nodes of the span is of degree 2. As in ring-based networks, a span next to a degree 2 node can only be covered by an on-cycle protection relationship. Thus, the basic idea is to accept that in cases (b) and (c) ring-like on-cycle coverage relationships are unavoidable, but for efficiency we arrange for every span with Condition (a) to be covered as a straddler relative to a p -cycle. To do this the steps of the algorithm are:

[\[View full width\]](#)

For each span: {

Find the shortest path between its end nodes excluding the span itself (ndpath=1) and the next shortest node-(or span-) disjoint path, if the latter exists, in which case ndpath=2).(Use $O(n \log n)$ Dijkstra's alg).

Case of:}

Condition (a): form a p -cycle on the two disjoint paths that exclude the span

Condition (b): form a p -cycle on the span itself plus the one disjoint path between node A and node B.

Condition (c): no action; rely on coverage being provided as a side-effect of p -cycles placed for spans above.)

end for}

Test all spans for restorability. If a span of condition (c) remains unprotected, add a

➡ p -cycle of type (b).

Figure 10-26. Basic cases considered by the straddling span algorithm (from [ZhYa02]).

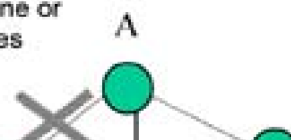
Condition (a):
• both nodes $d > 2$
• ndpath = 2

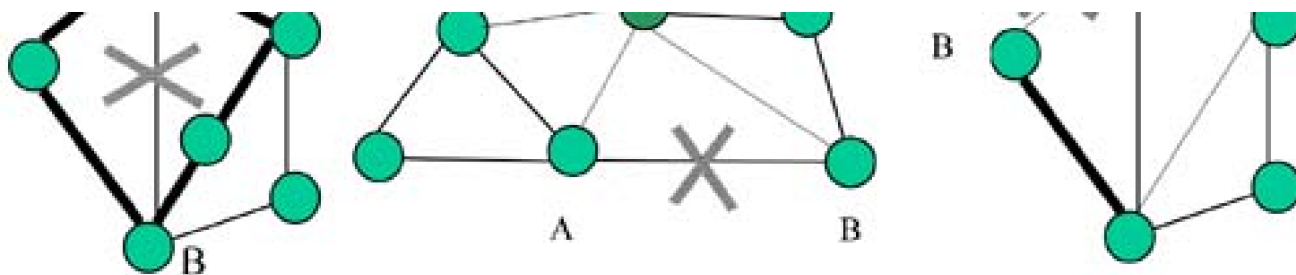


Condition (b):
• both nodes $d > 2$
• ndpath = 1



Condition (c):
• $d = 2$ at one or both nodes





In [ZhYa02] results for the heuristic were reported on two long haul networks both having a sparse mesh topology and average nodal degrees around 3. The results are summarized in [Table 10-7](#). Under the number of p -cycles, the two values given in X / Y format indicate the outcome if the disjoint path criteria is node disjoint (X) or only span disjoint (Y). All run times were under a second.

Table 10-7. Results from Straddling Span Algorithm [ZhYa02]

	USA Long Haul	France Telecom
Number of Nodes	28	44
Number of Spans	44	69
Average nodal degree	3.14	3.14
No. of p -cycles	27/24	41/34
Mean p -cycle length	6.25	6.29
No. of uncovered spans	0/0	0/0

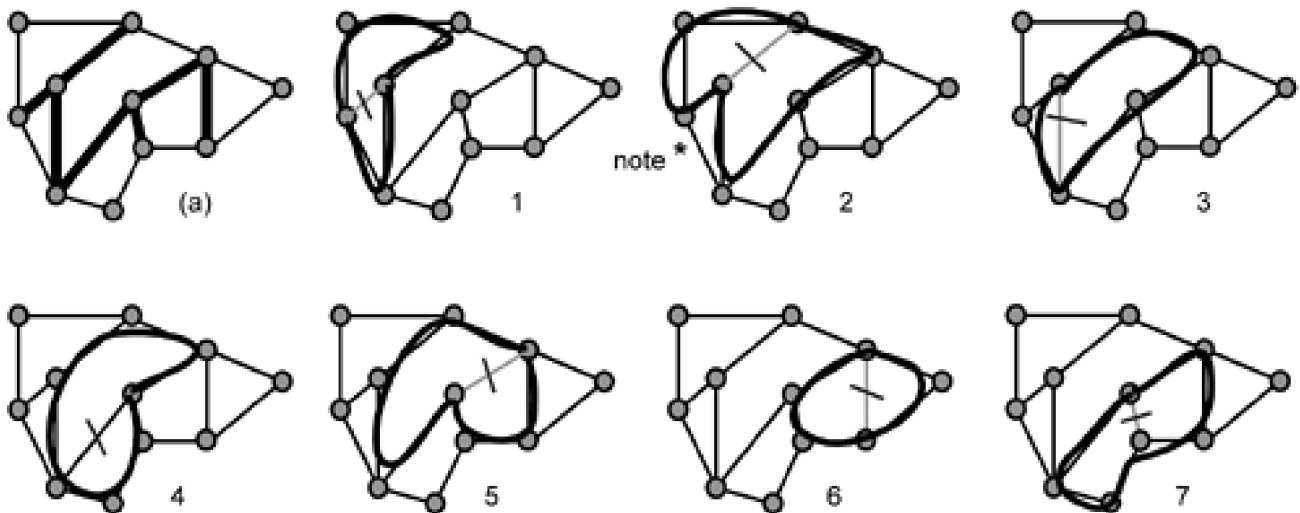
In summary, SSA is an extremely simple and fast procedure to produce high-coverage initial designs for further use in network planning. Note that the algorithm exclusively forms a particular type of p -cycle for each straddling span (i.e., spans of Condition (a)). These are what we will call "primary" p -cycles. Each primary p -cycle is in effect dedicated by its construction to only one straddling span, and the cycle it consists of is formed of the $k=2$ shortest disjoint paths excluding the span itself. (This could easily be changed to the shortest cycle containing the two end nodes but not the straddling span). Thus, it follows that the AE score for any primary p -cycles (that indeed has only the one intended straddler) is $AE_{primary} = (2+|C|)/c(C)$ where $|C|$ is the number of spans of the cycle C that is used and $c(C)$ is the measure of cost for cycle C . The other form of p -cycle formed by the algorithm (for spans of Condition (b)) are really non- p -cycles but rings in the sense that they have no straddlers and have $AE=1.0$ by definition (if we measure $c(C)$ by hop count).

As described in [ZhYa02] the algorithm takes a graph protection coverage view. This would be applicable to planning fiber-level protection assuming exactly two fibers on each span (with many wavelengths per fiber). If there are different working capacities to be protected on each span, the understanding would be that as many copies of each primary p -cycle are made as needed to match the working capacity of the corresponding straddling span.

10.7.7 Add and Join Operations on Primary p -Cycles

The family of primary p -cycles generated by the straddling span algorithm can be the starting point for heuristics that operate on the primary p -cycles to develop more efficient sets of p -cycles. The main concern with primary p -cycles is the amount of duplication or overlap that they imply. Put another way, primary p -cycles fail to share p -cycle spare capacity to cover more than one straddling span each. To appreciate the unnecessary duplication of coverage that a set of primary p -cycles may involve, [Figure 10-27](#) uses a small network example identifying all the spans of Condition (a) in the straddling span algorithm and the corresponding primary p -cycles. Note that although defined solely for the purposes of a single straddling span, primary p -cycle 2 actually has, by chance, a second straddling span topological relationship at the (*) note in [Figure 10-27](#). To avoid confusion below we consider only its primary intended straddler span.

Figure 10-27. The set of primary p -cycles generated by the straddling span algorithm.



We know from the preselection considerations just covered that high efficiency involves using p -cycles that provide multiple straddling span protection relations within the p -cycle. *Add* and *Join* operations on primary p -cycles can produce non-primary p -cycles with more straddling span relationships and hence network capacity efficiency.

Add Operation

An *Add* operation is possible on any two primary p -cycles $\{A\}$ and $\{B\}$ with straddling spans $i_s\{A\}$ and $i_s\{B\}$ when:

Equation 10.39

$$i_s\{A\} \in \{B\} \cap i_s\{B\} \in \{A\} \cap (i_s\{A\} \text{ not contra-adjacent to } i_s\{B\})$$

in which case the new p -cycle becomes:

Equation 10.40

$$\{C\} = \{A\} + \{B\} \equiv \{(\{A\} - i_s\{B\}) \cup (\{B\} - i_s\{A\})\}.$$

The resultant p -cycle $\{C\}$ is then of total length (in hop counts):

Equation 10.41

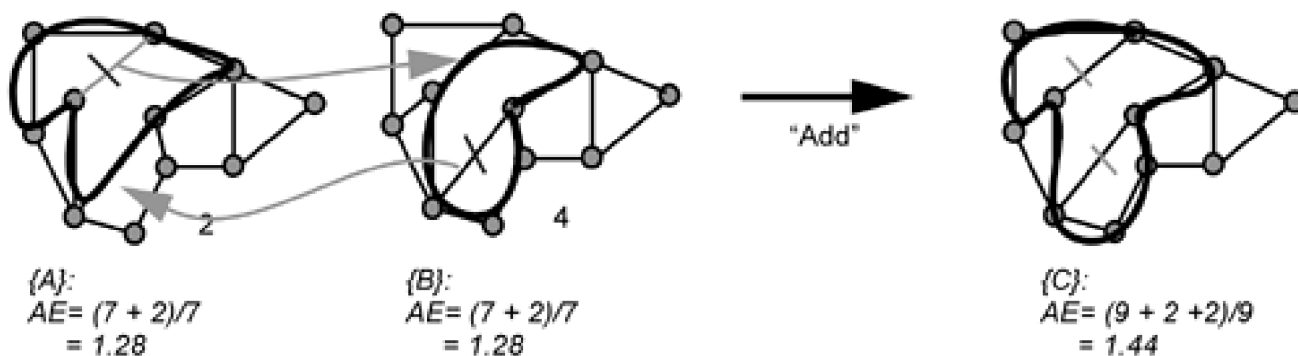
$$|\{C\}| = |\{A\}| + |\{B\}| - 2$$

and has both previous straddling spans as its new straddlers. The "contra-adjacent" condition remains to be defined below.

Less formally, an *Add* operation is possible when a pair of primary p -cycles each have the other's straddler as one of their on-cycle spans, and (for reasons to be explained) the two straddler spans involved are not adjacent in the graph. As an example, consider primary p -cycles 2 and 4 in [Figure 10-27](#). The qualifying condition and the result are illustrated in [Figure 10-28](#). The key is that each cycle contains

the other's straddler.

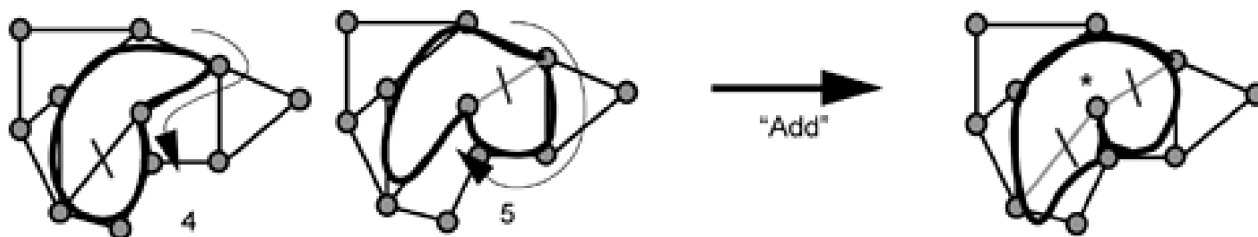
Figure 10-28. Example of the *Add* condition and operation on primary *p*-cycles.



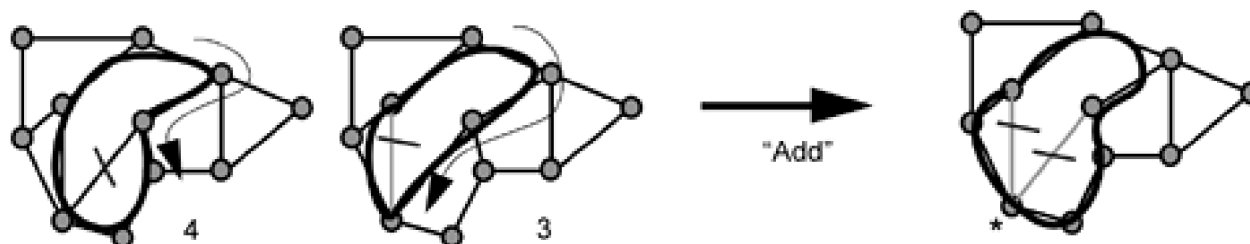
The exclusion condition of "contra-adjacency" means that the two straddlers cannot be both adjacent to each other and have their common node arrived at in opposite directions if a clockwise (or CCW) "tour" of each of the two primary *p*-cycles is taken. The resulting *p*-cycle may have a straddling relationship to the two spans as a concatenated subnetwork [Figure 10-29](#) illustrates using primary *p*-cycles (4 and 5) and (3 and 4) from [Figure 10-27](#). The result of the set union is not strictly a cycle, (unless it is interpreted as running both down and back on the visible "stub" section) nor does the identifiable outer cycle component by itself protect either previous straddler span.

Figure 10-29. Why *Add* operation is not allowed if the straddling spans are "contra-adjacent".

(a) straddlers adjacent and contra-directional at the common node (*): *Add* results in a looped *p*-cycle



(b) straddlers adjacent but co-directional at the common node (*): *Add* results in a normal *p*-cycle



This can be acceptable under the later flow-protecting *p*-cycle concept but it fails to meet the basic straddling span conditions for ordinary *p*-cycles. Note that technically the straddlers can still be protected if it was acceptable to run the *p*-cycle down and back on the "stub" span, but this implies twice the spare capacity for the *p*-cycle on that span as well as more complicated operation in general. However, we can invariably find sets of non-looped *p*-cycles that are just as efficient or better.

Join Operation

A *Join* operation is possible on any two primary *p*-cycles {A} and {B} with straddling spans $i_s\{A\}$ and $i_s\{B\}$ when:

Equation 10.42

$$|\{B\} \cap \{A\}| = 1$$

in which case the new p -cycle becomes:

Equation 10.43

$$\{C\} = \{A\} + \{B\} - \{\{A\} \cap \{B\}\}.$$

The resultant p -cycle $\{C\}$ is then of total length (in hop counts):

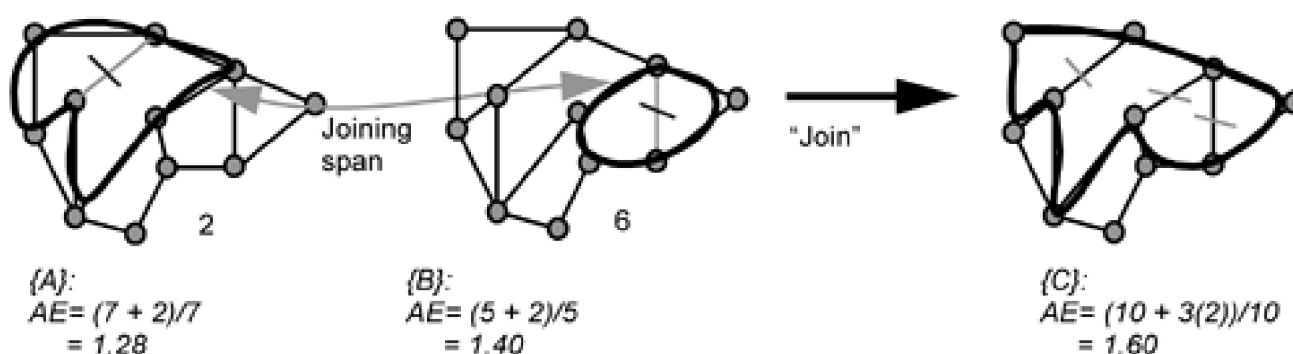
Equation 10.44

$$|\{C\}| = |\{A\}| + |\{B\}| - 1$$

and has the *three* straddling spans $\{s\{A\}, s\{B\}, s\{C\}\}$ where $s\{C\} = \{A\} \cap \{B\}$ is a new straddler formed out of the single span where both previous primary p -cycles have now been joined.

Less formally, the *Join* operation is possible when a pair of primary p -cycles have exactly one (non-straddler) span in common (i.e., on their cycles). As an example, consider primary p -cycles 2 and 6 in [Figure 10-27](#). The qualifying condition and the result are illustrated in [Figure 10-30](#). *Join* is generally more advantageous than *Add* because it adds one straddler (+2 on *TS* score) and deletes one span (-1 on cost), whereas *Add* keeps the same number of straddlers (+0 on *TS*) but deletes two spans from the total spans (-2 on cost).

Figure 10-30. Example of the *Join* condition and operation on primary p -cycles.



10.7.8 Application of *Add/Join* Operations to Design Improvement

Let us now use the network of [Figure 10-27](#) as a working example to show how an all-pairs examination of the primary p -cycles can produce a more efficient set of network p -cycles. In the example there are seven primary p -cycles so we need to inspect $7(6)/2 = 21$ combinations for possible application of *Add* or *Join* improvements. [Table 10-8](#) summarizes the feasible combinations of primary p -cycles

for Add / Join operations.

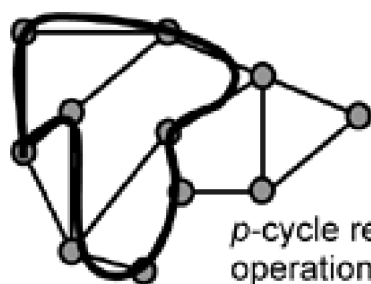
Table 10-8. Summary of Opportunities for Add or Join Operations on the Primary p -Cycles of Figure 10-27.

{A}	{B}	Add/Join feasibility?	{A}	{B}	Add/Join feasibility?
1	2	no	3	4	Add
1	3	no	3	5	Add
1	4	no	3	6	Join (*)
1	5	no	3	7	no
1	6	no (disjoint)	4	5	no
1	7	no (disjoint)	4	6	no
2	3	no	4	7	no
2	4	Add (*)	5	6	Add
2	5	Add	5	7	no
2	6	Join	6	7	Add
2	7	no			

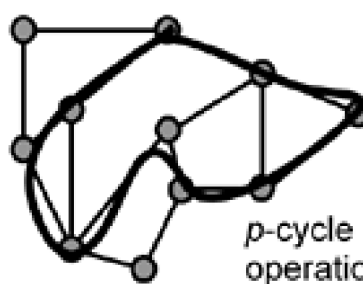
Not all of these operations can necessarily be applied in combination because a primary p -cycle ceases to exist as such once an operation has been applied to it. We can therefore consider which operations to employ on a cycle by cycle basis. Cycle 2 has three operations available to it. A *Join* is generally most advantageous but both of the available *Joins* are to cycle 6. We should therefore first decide the best *Join* operation to commit to for cycle 6. It turns out that *Join*(6,3) is preferable to *Join*(6,2) resulting in a new p -cycle with $AE=1.75$.

Let us commit to the choice of *Join*(6,3) as this also leaves cycle 2 open for a separate choice of *Add* operations. Let us choose *Add*(2,4). (*Add*(2,5) is equally good on AE terms.) Thus, we have two mutually compatible improvement operations to apply as transformations on the initial set of primary p -cycles. These are: *Join*(6,3), *Add*(2,4). The result of applying these two operations is a pair of p -cycles, both of 9 hops, shown in Figure 10-31. They have in one case two straddling spans, the other with three straddlers and they alone provide complete coverage of all spans in the graph. This should be compared to the seven primary p -cycles consuming a total of 43 spare channels from the initial straddling span algorithm.

Figure 10-31. A complete protection design using two p -cycles and 18 spare channels found by Add / Join operations on the seven primary p -cycles for the corresponding network.



p -cycle resulting from operation *Add*(4,2):
 $AE = (9 + 2(2))/9 = 1.44$



p -cycle resulting from operation *Join*(6,3)
 $AE = (9 + 2(3))/9 = 1.66$

10.7.9 General p -Cycle Merge Operation

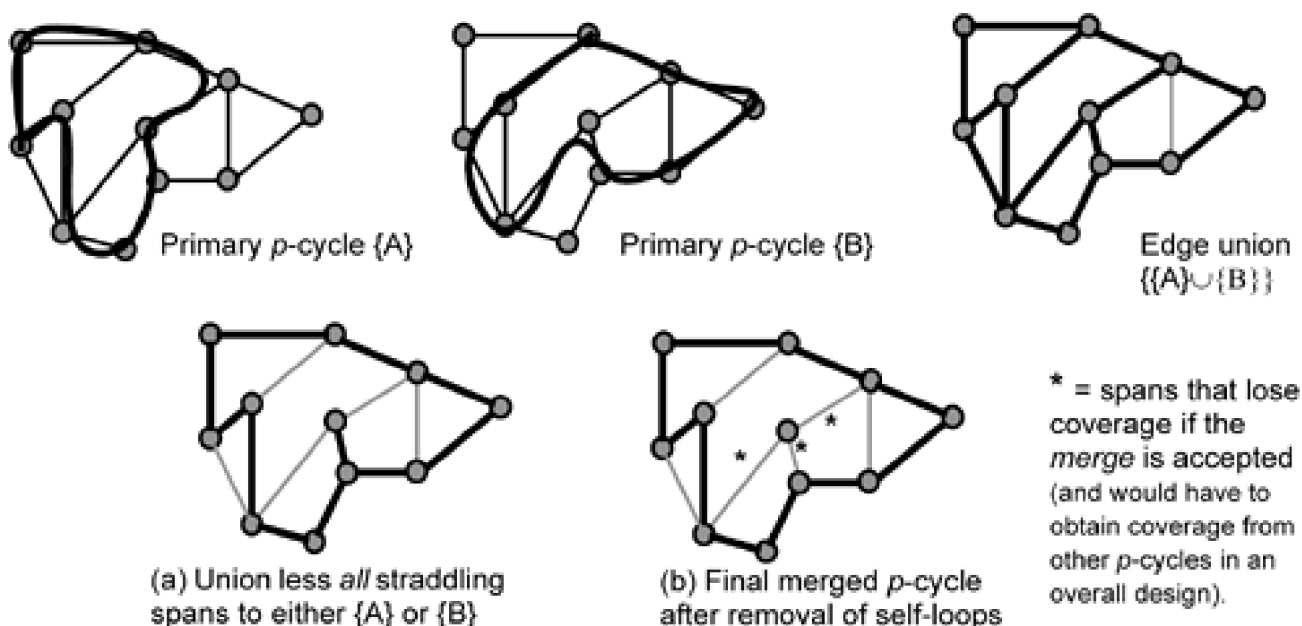
Add and *Join* were defined above as fairly simple and convenient operations that are specifically intended only for application to pairs of primary p -cycles. More generally, however, a variety of meta-heuristics (GA, SA, Tabu search for example) could all be formulated to employ a basic *merge* operation as a search step looking for a reduced number of more efficient p -cycles. The general merge operation has to accept any two existing p -cycles and see if a single p -cycle of higher efficiency results. One possible definition of a general *merge* operation is:

Equation 10.45

$$\text{merge}(\{A\}, \{B\}) \equiv \{A\} \cup \{B\} - (\{\text{straddling spans } A\} \cup \{\text{straddling spans } B\})$$

the result of which is illustrated in [Figure 10-32](#) for the two p -cycles of [Figure 10-31](#). The result of a merge operation has thus to be tested for suitable properties. The example shows an outcome where, depending on viewpoint, we either have a non-simple p -cycle with a self-loop on one span (a), or, after elimination of any implied self-loops (b) a normal p -cycle. In general use within a design enhancement search this pairing of p -cycles under the *merge* operation can then be evaluated in terms of total cost and total protection relationships provided and either adopted or rejected relative to keeping the two initial p -cycles. Note that if this particular *merge* operation was accepted (in the form of [Figure 10-32\(b\)](#)) that the coverage level of network spans goes down as a result and the ongoing search would have to find some other p -cycles to protect the spans marked (*) in [Figure 10-32\(b\)](#).

Figure 10-32. Example of generalized p -cycle Merge operator applied to p -cycles {A} and {B}.



10.7.10 Simulated Allocation Approach for Joint Design Algorithms

One of the shortcomings of the preceding purely algorithmic approaches (i.e., not the preselection heuristic for MIP-based solutions) for p -cycle placement is that they really only address the non-joint design problem. We assume a routing of working demands then look for an efficient set of p -cycles with respect to that pattern of w_j accumulations on each span. One way to approach joint optimization in a heuristic would be to couple the latter p -cycle placement processes with a Simulated Allocation approach to explore global effect arising from

changes in the working route assignments. Simulated Allocation is like Simulated Annealing to make random rearrangements to the details of a solution with a time-decreasing probability of accepting a step backward in terms of objective function increase. A general description and application of Simulated Allocation to path-restorable design heuristics was given in [Section 6.16](#). The application to joint p -cycle design is similar.

As applied to p -cycle design problems the basic random perturbation would be to select a demand and change its route, followed by a rerunning or simply an efficient update procedure to the p -cycle placement generated by the fast iterative algorithm above. Also key to exploring the combined design space of routing and p -cycles would be to not always place the locally best p -cycle at each iteration but with a random decreasing probability, deliberately accept the second or third best p -cycle candidate at each iteration.

[\[Team LiB \]](#)

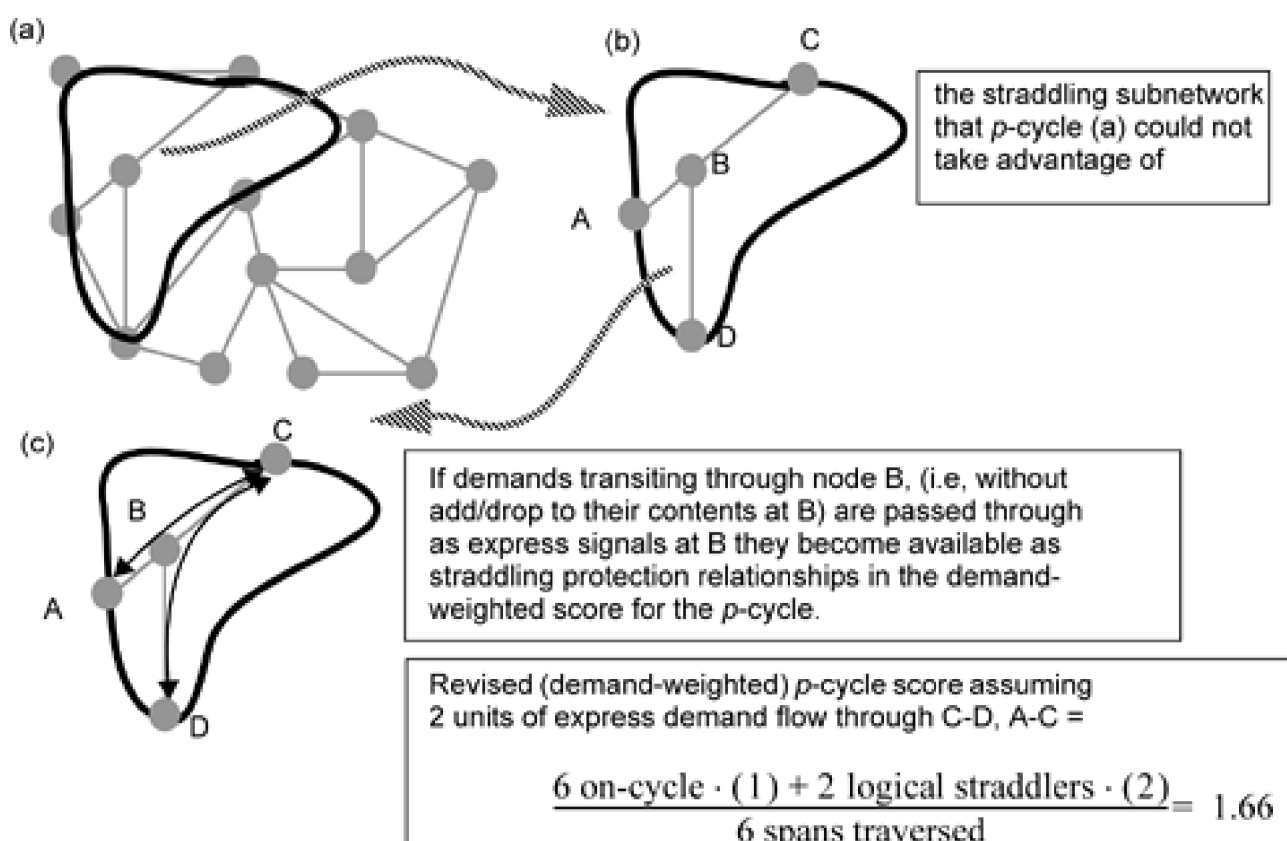
◀ PREVIOUS NEXT ▶

10.8 Concept of a Straddling Subnetwork and Domain Perimeter p -Cycles

Note that the spans marked (*) in [Figure 10-32\(b\)](#) cannot be protected by the p -cycle shown because, as so far defined, these spans have neither an on-cycle nor a straddling relationship to the cycle shown. Although our aim is not to fully treat the topic here, we want to give an overview of a significant extension to the p -cycle concept to cover *straddling flows*, not just straddling spans per se. Straddling flows can transit any subnetwork that is in a straddling relationship to a p -cycle. The complete design model and theory is available in [\\$hGr03b](#). Here our emphasis is only on the concept involved and some particular applications it enables.

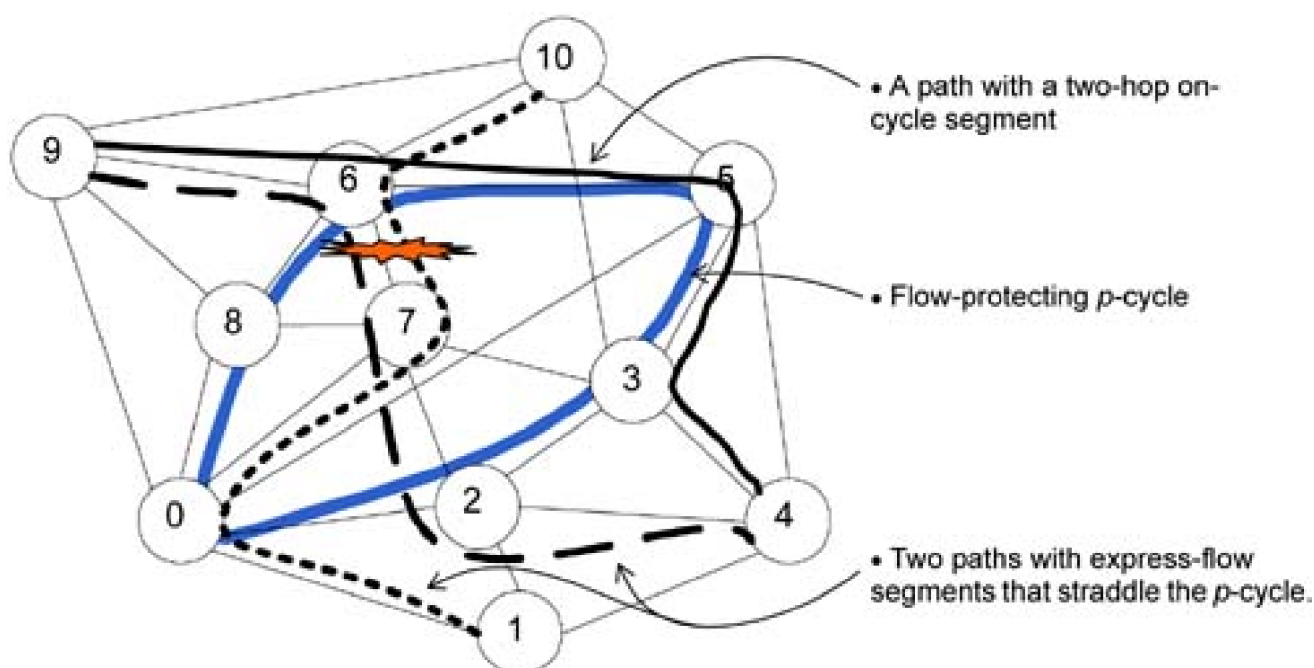
For discussion [Figure 10-33\(a\)](#) shows a p -cycle that as so far considered we would say only has on-cycle spans. But we can see that there is a kind of "straddling subnetwork" present in the form of the tree formed on nodes A,B,C,D. None of these spans individually has a straddling span relationship per se, but [Figure 10-33\(c\)](#) shows, however, how the p -cycle could still be used to protect up to two units of demand that flows entirely through the subnetwork between each pair of nodes of the subnetwork that are nodes of the p -cycle. This is only possible if there are express flow demands between the respective boundary node pairs of the subnetwork. By express demands we mean that, with reference to the example, they are lightpaths or other carrier signals that are unaltered in their demand composition at node B. In addition such demands also physically bypass the equipment at node B, so that a failure state on physical spans CB, BD or AB will propagate to the outer nodes of the subnetwork, which are on the p -cycle boundary. Alternately, node B must be just arranged to generate AIS on the surviving path segments if spans CB BD AB are terminated at the line level at node B. The principle of bypass through a subnetwork that is involved here is really the same one as seen to be advantageous for restoration efficiency in the case of degree 2 chain subnetworks in [Chapter 5 \(Section 5.8.1\)](#). It is true that this does not allow the p -cycle in question to completely protect the A-D-C subnetwork, but by adding up to four additional protection relationships to the p -cycle in the example, we are reducing the remaining amount of capacity on spans CB BD AB (in this case by a total of eight 1 - channels) that still need to be protected by some other p -cycle(s).

Figure 10-33. Example of how nodal bypass of express demands through "straddling subnetworks" can improve the demand-weighted score above the topological score.



The full generalization of this point leads to the concept of "flow protecting p -cycles" [ShGr03b]. It was natural, even at the time of the work in [Stam97] and [GrSt98], to ask if there was a *path protection equivalent* to basic span protecting p -cycles. As simple as basic p -cycles are, the latter question turned out to be difficult to address. Indeed we volunteer that flow protecting p -cycles may be more important as a theoretical advance than for direct application. Ultimately the difficulty is how to handle the aspect of "mutual capacity" contention (which is intrinsic to any path-oriented or multi-commodity flow type of recovery scheme) in formulating the design model under a paradigm of cyclic spare capacity structures. The corresponding operational complexity lies in coordinating which paths can access which p -cycles because the use of each p -cycle now becomes failure-specific. Figure 10-34 shows an example for discussion of the added efficiencies and the complicating issues. In Figure 10-34, spans (6-7) and (7-2), and several others, of the p -cycle are close to being straddling spans but cannot actually be span-protected by the cycle shown. However, a *path* that crosses both spans (6-7) and (7-2) as shown can be considered to straddle the cycle when taking a *path-level* view. More specifically the path segment (6-7-2) straddles the cycle shown between the nodes 6 and 7.

Figure 10-34. Extension of the straddling span concept to p -cycle protection of a straddling (express) flow through a subnetwork that straddles the p -cycle.

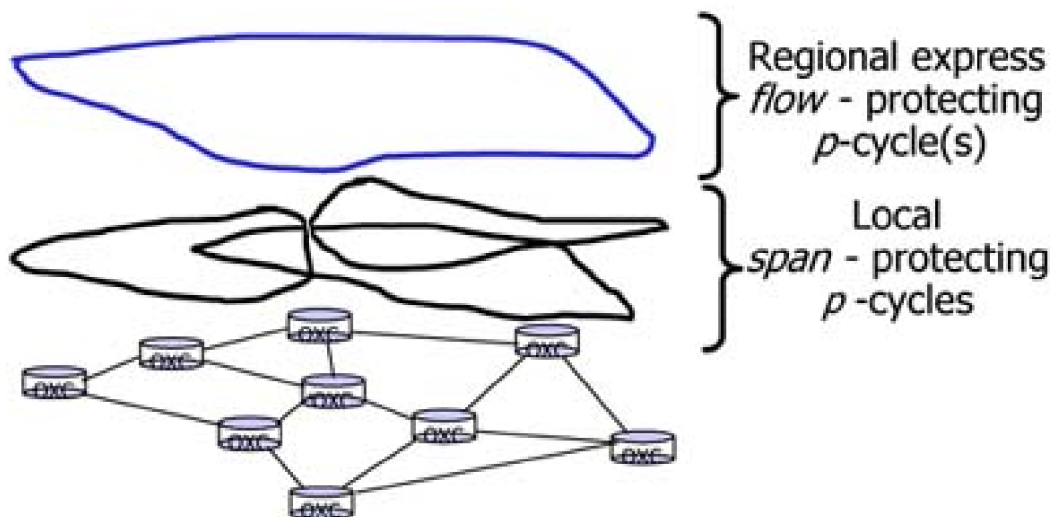


Clearly flow p -cycle designs can access more opportunities for spare capacity sharing than the span p -cycle method. Any contiguous flow segment that intersects a p -cycle can be protected, not simply spans directly on or straddling the cycle. Any demands being added or dropped at node 7 cannot be handled by the same flow p -cycle. The flow must be unchanged in its composition between the nodes where it intersects the flow p -cycle. Theoretically, flow p -cycles can be categorized as the equivalent to path restoration with stub reuse. The main complication that arises with admission of straddling flows is, however, that the ability of the cycle to provide the assessed number of protection paths can depend on which other paths fail at the same time, or in other words, *where* in the straddling subnetwork the span failure occurs. Unlike span protecting p -cycles, when a span fails in the flow-oriented context, path segments with more than one set of end nodes are affected simultaneously. Because the affected paths (more specifically here, the affected path segments) all have different intersecting nodes on the flow p -cycle, we get into the problem of "mutual capacity." This is the fundamentally complicating aspect of efficient rerouting problems involving multiple node-pairs simultaneously. Here we see the issue manifest itself in Figure 10-34: if span (7,2) fails, then only the flow segment for path (4-9) needs protection and it can obtain two protection paths from the cycle shown. But if span (6,7) fails, then access to the same flow p -cycle must be coordinated between the segments for paths (9-4) and (10-1), which fail simultaneously. Nonetheless if the complexities are managed in the design, results in [ShGr03b] show that networks based on flow p -cycle designs can typically require 10 to 20% less spare capacity than span p -cycle designs.

Although generally more complex as mentioned, an appreciation of the basic idea suggests some simpler adaptations of flow p -cycles that may be quite practical. One is to use flow p -cycles only for the important express flows through a network region. Conceptually one can picture an overall network design comprised in part of a set of simple fast-acting "local" span protecting p -cycles. Logically overlaid on this is a select set of "express flow" protecting flow p -cycles. An economic advantage of this selective use of a few flow p -cycles is that the express flows may take advantage of long-reach optics for optical bypass of all intermediate nodes on the protected flow segment. This allows the express flow signals to remain in the optical domain through the entire region, but remaining protected, saving considerably

over the alternative of terminating on each OXC en route. Closely related to this is the concept of an *area-boundary flow-protecting p-cycle* in the context of multi-domain optical networks. The concepts are summarized in [Figure 10-35](#).

Figure 10-35. Using a mixture of span-protecting p -cycles and a regional boundary flow p -cycle to protect all flows that completely transit the domain.



[[Team LiB](#)]

◀ PREVIOUS NEXT ▶

10.9 Extra Straddling Relationships with Non-Simple p -Cycles

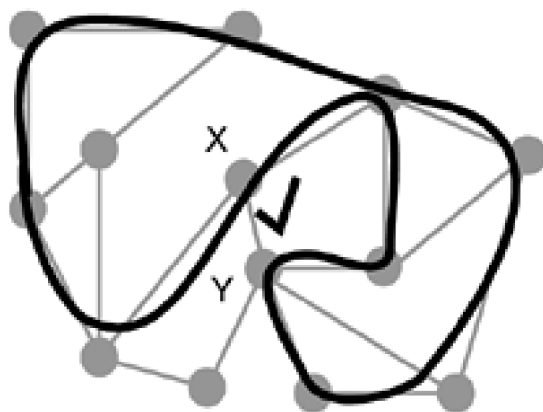
So far in developing the applications and theory of p -cycles, we have taken it as simply a practical assumption that we will work with simple cycles only. If we admit non-simple cycles as well as simple cycles, there are vastly more candidate cycles to populate the set P and we risk creating a perception of unacceptable complexity when picking p -cycles to practical network operators. p -Cycles deployed as a networking concept on simple cycles would be eminently practical, manageable and efficient. But p -cycles proposed on non-simple cycles is about as likely as being adopted in practice as would figure-eight rings have been in the SONET era. Setting aside practical concerns of acceptability of the engineering solutions based on p -cycles, however, there certainly is no theoretical barrier that would stop us from admitting non-simple cycles as candidates for p -cycles. Our aim now is just to show that higher efficiency scores can be achieved in general with non-simple cycles than when restricting ourselves to simple cycles.^[7]

^[7] This effect was first pointed out to the author by D. Schupke and C. Gruber.

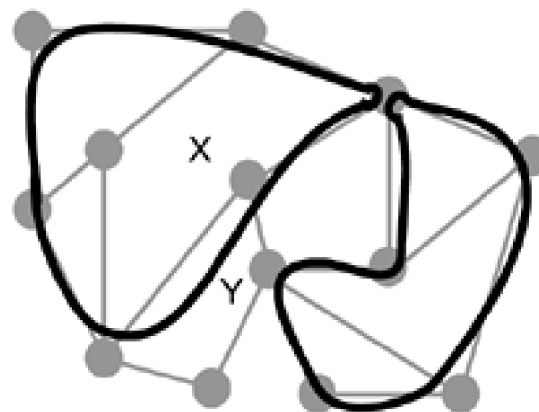
To illustrate this, consider again the network of [Figure 10-19](#). [Figure 10-36](#) shows a construction of a non-simple cycle (it passes through one node twice) that provides normal p -cycle type straddling span protection to span X-Y. In contrast, the two component simple cycles that correspond to the same non-simple cycle cover all the same other on-cycle and straddling spans, except for span X-Y.

Figure 10-36. Example of how a non-simple p -cycle (a) can pick up an extra straddling protection relationship that is not available to either of the two corresponding simple cycles (b).

(a) Span X-Y is covered



(b) Span X-Y is not covered



10.10 Hamiltonian p -Cycles and Homogeneous Networks

It is easy to observe that if the network is Hamiltonian, a single p -cycle can protect all spans of the network. This simple observation was first made in [StGr00b] and pursued further in [HuCo02]. Often this would be far from the most capacity-efficient design, however, because with different amounts of working capacity on each span, a specific set of several p -cycles—some of which may individually be Hamiltonians, but in general including smaller p -cycles as well—provide a solution that uses less total spare capacity. Nonetheless there is a context in which Hamiltonian p -cycles could be of special practical interest. This is the possibility of protecting entire networks at the whole-fiber level of protection switching, or more generally protecting what is called a "homogeneous network" in which all spans are assumed to have identical amounts of installed channel capacity. The consideration of p -cycles in homogeneous Hamiltonian networks also leads to some interesting further insights about p -cycle-based networks in general and leads to the novel concept of a semi-homogeneous network that is specifically inspired by p -cycles and has redundancy that is exactly at the $1/(d-1)$ lower bound.

10.10.1 Concept of a Homogeneous Network

An outcome of DWDM technology may be that for some networks, especially small or moderate sized metro or regional networks, all required capacity can be supported on a single pair of fibers per span, one for each direction. An assumption of this type, and that costs are incurred to turn up an equal number of wavelength channels on all fibers, gives rise to the paradigm of "homogenous" networks, or perhaps more descriptively, "flat capacity" networks, where every span of the network is assumed to have equal (and essentially unlimited or always adequate) capacity. This is the paradigm in which Generalized Loopback [MeBa02] and Ellinas's Protection Cycles [EIHa02] are both proposed. Under the homogenous network assumption, network planning problems become *uncapacitated* versions of the more general capacitated design problems we have considered throughout the book. They collapse to a view of the network as a simple graph, not a capacitated multigraph. All edges have the same capacity and the capacity is assumed adequate to support any foreseeable amount of working flow on an edge. In this framework problems such as ring covering become purely logical graph covering problems.

It is hard to judge how practical or economically realistic such flat capacity networks might be because the implication is that, assuming a DWDM transmission system on every fiber, all systems implement the requirements of the single highest channel requirement of any link under the actual demand pattern. Even if the demand matrix is fully uniform, the load on each link is highly non-uniform. So the homogeneous view inherently assumes that, for example, in a network of S spans, S 2-fiber DWDM systems capable of supporting 512 channels per fiber would not cost significantly more than two fibers at 256 channels each for the most highly loaded link and mixtures of systems of 256, 64, 32, or 16 channels (as needed) on multiple fiber pairs on other links. To prove-in economically a homogenous network thus seems to assume the most advanced and cost-reduced future DWDM technology possible because at the optical path level a DWDM network is never homogenous. Even if the fibers support an infinite number of channels, there is still a cost for each DWDM channel turned up. It is reasonable to expect operators to differentially capacitate the number of channels on each span to suit actual requirements. Even assuming dynamic lightpath provisioning on demand, it would not necessarily be economic to turn up exactly the same number of operating channels on every link. Thus, applicability of the homogenous network view may be limited at the lightpath level.

It would, however, seem more relevant to the prospect of implementing fiber-level protection where a single fiber-pair fully suffices everywhere even if each fiber has a different number of turned-up channels (i.e., the concept of "dark fiber p -cycles" in [Section 10.4.2](#)). Fiber-switching cross-connects (or simpler fiber-switching devices made specifically as p -cycle nodes) can implement p -cycle protection at the whole-fiber level. Such fiber-level protection of a network could be especially cost-effective because there are no wavelength assignment or wavelength conversion costs when switching the whole fiber for protection. Implicitly every l -assignment retains continuity under protection re-routing because the corresponding l -channel is by definition free for use on the dark fiber (for any single failure). And at the whole fiber level, a network may indeed look homogenous as long as the entire demand flows in the network remain accommodated by the number of DWDM channels that can be supported on a single fiber. In any case the supposition of homogeneous networks poses an interesting "what if" question regarding the role of p -cycles, and especially Hamiltonian p -cycles, in such networks.

10.10.2 The Role of Hamiltonian p -Cycles in Ordinary Capacitated Designs

As a preliminary, let us first develop a few aspects of ordinary capacitated p -cycle designs. The notions of p -cycle score ([Section 10.7.1](#)) already make it apparent that a Hamiltonian cycle will always have a high AE—it will cost N hops to construct and provide a TS of $N + 2(S-N)$ for an AE score of $2(S-N)/N = 2S/N - 1 = (\bar{d} - 1)$. So in a network of degree 3.5, any Hamiltonian will have an attractively high AE score of 2.5. Note that although a Hamiltonian has every span of the network either as a straddler or on-cycle span, this is not the same as saying it has the "perfect score" of $(N-2)$ from [Section 10.7.2](#), because that applies to a Hamiltonian in a network that is a full mesh of N nodes, (i.e., N nodes and $N(N-1)/2$ spans).^[8] Nonetheless, it is reasonable to expect that Hamiltonian p -cycles will play a generally important role in optimal p -cycle network design. It may even be tempting to think that optimal designs could be based only on Hamiltonian p -cycle. But this would be mistaken. One of the things we want to illustrate here is that while Hamiltonians are important building blocks of good overall p -cycle designs, using only Hamiltonians, or using a single Hamiltonian, would be far from optimal in general.

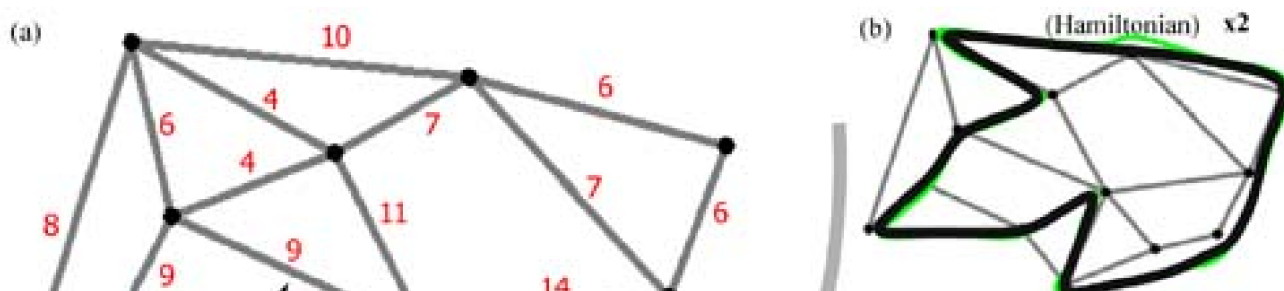
^[8] Parts of this section were excerpted during preparation for publication in *IEEE Network Magazine* [\[SaGr03\]](#).

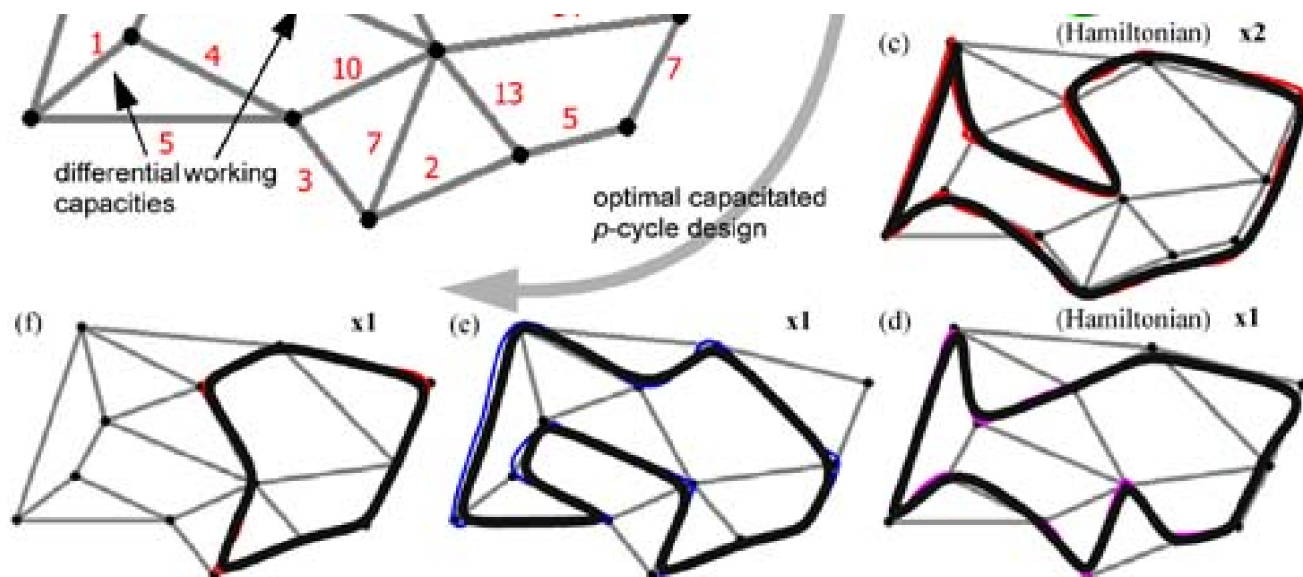
To do so, Anthony Sack has provided optimal solutions to a small test case for use here. Let us inspect some properties of a strictly optimal p -cycle design obtained with the basic PC-2 design model ([Section 10.3.3](#)). This is the basic model for non-joint design of a minimum capacity set of p -cycles for 100% span protection. [Figure 10-37\(a\)](#) shows a test network of 23 links and 13 nodes that is Hamiltonian and has the working capacity counts (w_j) that arise from least-hop routing of one unit of demand between each node-pair. Note immediately how variable the required number of working channels on each span is, even when the demand matrix is completely uniform. This reinforces the point made earlier that homogeneous networks—and related designs based on a single Hamiltonian p -cycle—are of questionable practical importance. Even if every span of [Figure 10-37\(a\)](#) consisted of one bidirectional fiber pair using DWDM technology capable of, say, 32 I per fiber, the network would be homogeneous at the fiber level, but quite inhomogeneous at the channel level. Significant cost in a DWDM system is related to the number of optical channels turned up.^[9]

^[9] It should also be observed that while the span with 14 required channels may be well-served with DWDM infrastructure for 32 I per fiber, spans with only 1 to 3 working channels would be more cost effectively implemented with DWDM that only supports (say) 8 I per fiber. The common infrastructure cost and per-channel cost of the latter both cost less than the former. It is thus even harder to see any economic justification for assumptions of homogeneous networks at the DWDM channel level. Homogeneity at the whole-fiber level is somewhat more plausible, but even this is illogical where fiber is not in short supply. It will then usually be more economic to use multiple fibers in parallel on some spans, but not others, instead of installing common equipment infrastructure for the highest DWDM channel capacity on every span, just to realize it over a single fiber-pair.

The network in [Figure 10-37](#) has 410 distinct simple cycles comprising P in its optimal design problem. Using the complete set of 410 candidate cycles, the PC-2 formulation is solved to optimality with parallel CPLEX 7.1 in 0.63 s. The result—detailed in [Figure 10-37\(b\)](#) through (f)—is optimal in terms of requiring the least number of spare channels (all $c_j=1$) to protect all w_j values. Seven unit-capacity p -cycles exist in the solution using five distinct cycles. Three of these are Hamiltonians. The two non-Hamiltonians arise—and are a key part of an economic solution—because of differential working capacity considerations. Smaller more precisely placed p -cycles are required to match the actual capacity distribution to be protected, complementing the Hamiltonians. The "x1" and "x2" notation denotes the number of unit-capacity p -cycles instantiated on each of the cycles shown. This simple result illustrates an important general point—that the optimal solution to a capacitated problem involves a number of individual p -cycles, many of which may be Hamiltonian, but no "single best" Hamiltonian suffices, and in addition other p -cycles arise that are not based on Hamiltonian cycles.

Figure 10-37. Optimal capacitated p -cycle design employs Hamiltonians and other cycles.





The design in [Figure 10-37](#) is optimal and has a logical redundancy of 53.8%. In the corresponding distance-weighted optimal design (where c_j is proportional to the length of edge j as drawn), thirteen unit capacity p -cycles are employed on seven distinct cycles but only two p -cycles are also Hamiltonians. The distance-weighted redundancy is 65.9%. It makes sense that fewer Hamiltonians also arise in the distance-weighted optimal solution. Under distance-weighted capacity, a Hamiltonian has no particular guarantee of efficiency that is higher than a non-Hamiltonian for which the ratio of total straddling span distances protected is high relative to the p -cycle circumference. In other words, the previously defined distance-demand weighted efficiency ($DDWE$) of p -cycles ([Equation 10.38](#)) is actually paramount and more general than a criterion that p -cycles be Hamiltonians. In summary, we can see overall that efficiency in the sense of $DDWE$ is ultimately what always matters to characterize a good p -cycle design. In the case where capacity is *not* distance weighted (i.e., logical capacity counts only) it turns out that Hamiltonians will maximize DWE if a Hamiltonian is feasible and if unprotected capacity remains to be covered by the Hamiltonian p -cycle on all spans of the network.

To compare the seven p -cycle solution in [Figure 10-37](#) to the best case if only one ("best") Hamiltonian p -cycle were employed, we can observe that with a single p -cycle the entire p -cycle would have to have a capacity of seven. This is because (in the best case) the link with $w_i = 14$ (the maximum present) would have a straddling relationship to the single assumed Hamiltonian p -cycle. In this case the single p -cycle would require $23(7) = 161$ units of protection capacity. There being a total of 158 units of working capacity, this represents a logical redundancy of 102%. In contrast, the seven-cycle optimal solution is constructed with only 85 units of protection capacity (for 53.8% redundancy). This illustrates why the tempting notion of using a single Hamiltonian p -cycle is far from optimal for a capacitated network design.

10.10.3 p -Cycle Design in Homogeneous Hamiltonian Networks

Let us now consider homogeneous networks where the graph is also Hamiltonian. Intuition suggests in this case that a single Hamiltonian p -cycle may be the optimal (i.e., minimum cost) solution. If costs are hop-based, then in an N -node network, any Hamiltonian cycle has N hops and every link of the network has either an on-cycle or a straddling relationship to the Hamiltonian cycle, so any one Hamiltonian cycle protects all spans at the theoretical limiting efficiency. If cost is distance-based then the shortest Hamiltonian will usually be a strong contender for the most capacity-efficient solution in a homogeneous network, but it may not always be the best. This is because with arbitrary edge costs, a counter-example such as that in [Figure 10-38](#) (provided by D. Schupke) can always be constructed where ∞ is an arbitrarily large constant.

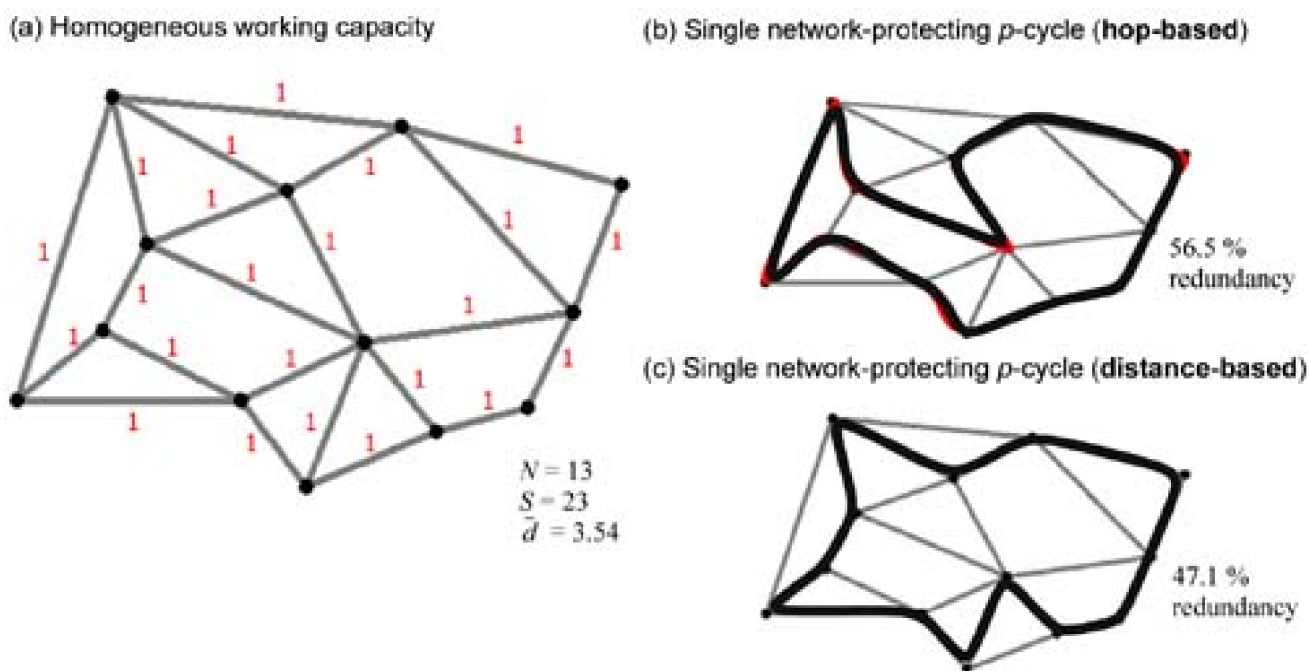
Figure 10-38. With arbitrary real-valued edge costs, Hamiltonians are not always most efficient.





It is interesting to look at how some actual solutions to PC-1 behave in this regard, in a homogeneous network where, in one case (i) edge costs are all constant (i.e., hop-based distance) and (ii), where edge costs are real-valued but reflect the "ordinary" (i.e., Euclidean) distances as seen on a diagram of the network. The expectation is that if the working capacity is homogeneous then any Hamiltonian will be the best solution in the hop-based case. And—notwithstanding the formal counter-example above—we still expect that in the real-valued case a single least-cost Hamiltonian will often still emerge as the optimal solution. We tested this using PC-2 in the homogeneous network shown in [Figure 10-39\(a\)](#) (and in several other cases not shown). In all of these trials, single Hamiltonian p -cycles—shown in [Figure 10-39\(b\)](#) and [\(c\)](#) for the network shown—constitute the optimal solutions for the *homogeneous* network case. Where distance costs are hop-based the resulting Hamiltonian p -cycle has a logical redundancy of $13/23 = 56.5\%$. Notably this is also exactly $2/\bar{d}$ for this network. This is a general result we predict below. Note that the particular Hamiltonian found (shown in [Figure 10-39\(b\)](#)) is not unique. Any Hamiltonian will yield the 56.5% hop-based redundancy in this network. When span costs are proportional to the Euclidean distance between end-points, however, a different (and generally unique) single Hamiltonian p -cycle arises (shown in [Figure 10-39\(c\)](#)) with an even better distance-weighted redundancy of 47.1%.

Figure 10-39. Optimal p -cycle solutions in a homogeneous Hamiltonian test network.



10.10.4 Lower Bounds for p -Cycles on a Hamiltonian Network

If we know that the optimal solution for a homogeneous Hamiltonian network is always a Hamiltonian p -cycle, then we can also derive some general bounds on efficiency. Consider a network of degree \bar{d} , with N nodes and S spans, and assume it is Hamiltonian. Taking note that $\bar{d} = 2S/N$, a simple initial view of the logical (i.e., hop count based) redundancy in the homogeneous case would be:

Equation 10.46

$$\sum_{w_i \in \mathcal{C}} s_j / \sum_{w_i \in \mathcal{C}} w_j = \frac{N}{S} = \frac{N}{N \cdot \bar{d} / 2} = \frac{2}{\bar{d}}$$

11 11

Thus, where applicable, a single Hamiltonian p -cycle would be extremely efficient. Note an implication of [Equation 10.46](#), that is quite in line with intuition in this case, is that in order to reach the minimum redundancy of $2/\bar{d}$, the solution must be based on a single p -cycle which therefore must also be Hamiltonian. If any more than one cycle was used, the number of spare links in the redundancy calculation would be higher than N .

This result ($2/\bar{d}$) would be the lower bound on redundancy for a strictly homogeneous network. However, under strict homogeneity we fail to reflect or otherwise exploit an important aspect of p -cycles: that a straddling span can actually have twice as much working capacity as the capacity of the p -cycle that protects it. This is a special sense in which p -cycles would support selected departures from strict homogeneity (discussed further below). If a span is a straddler to the network Hamiltonian p -cycle, then its particular fiber (or channel) count can in fact be doubled, without any change or addition required to retain full protection. We will return to look at how this could be implemented. For now, if we take this into account in the lower bound, the sum of working channels changes from being just S to

Equation 10.47

$$\sum_{\forall j \in S} w_j = N \cdot 1 + (S - N) \cdot 2 = N + 2 \cdot \left(N \cdot \frac{\bar{d}}{2} - N \right) = N \cdot (\bar{d} - 1)$$

and so the redundancy becomes

Equation 10.48

$$\sum_{\forall j \in S} s_j / \sum_{\forall j \in S} w_j = \frac{N}{N \cdot (\bar{d} - 1)} = \frac{1}{(\bar{d} - 1)}$$

which implies that the lower-bounding redundancy of a span-restorable mesh network is *exactly* realized! Interestingly, we can note with hindsight that when we first derived the $1/(\bar{d} - 1)$ lower bound (in [Section 5.4.2](#)) the assumption of evenly matched working capacity at each span at the isolated node under consideration is in effect the same as the assumption being made here of a homogeneous network. This also gives us a certain further insight about the general capacitated p -cycle network design problem. It tells us (as did the considerations of p -cycle "score" in [Section 10.7.1](#), but from a different viewpoint) that in a good p -cycle design many individual p -cycles *will* tend to be Hamiltonian cycles and in the limit where all p -cycles employed in a capacitated design are Hamiltonians (although there may be many different Hamiltonians), the limiting redundancy of $1/(\bar{d} - 1)$ would be achieved even in a capacitated network.

If we now revisit the capacitated p -cycle design above ([Figure 10-37](#)) we see that its logical redundancy of 53.8% is *below* the prediction of $2/\bar{d}$ which would apply for this network if it was homogeneous, but above $1/(\bar{d} - 1)$ (which is 39.4%) if it was *semi-homogeneous* in the way defined above. In fact this is consistent and explained by the inference just above — the set of p -cycles for the capacitated problem are evidently being chosen in a way that exploits the two-times protection factor of straddling links (which the homogenous model cannot reflect) as often as possible in the design, although it is not always possible to associate two working capacity units with every possible straddling relationship (which the pure semi-homogenous view assumes). Thus, there is clearly an aspect to the capacitated p -cycle design problem that is not taken advantage of in a strictly homogeneous network.

Does Maximum Score *Imply* a Hamiltonian Cycle?

It may be tempting at this point to want to generalize in the reverse direction that in a homogeneous Hamiltonian network, the p -cycle with

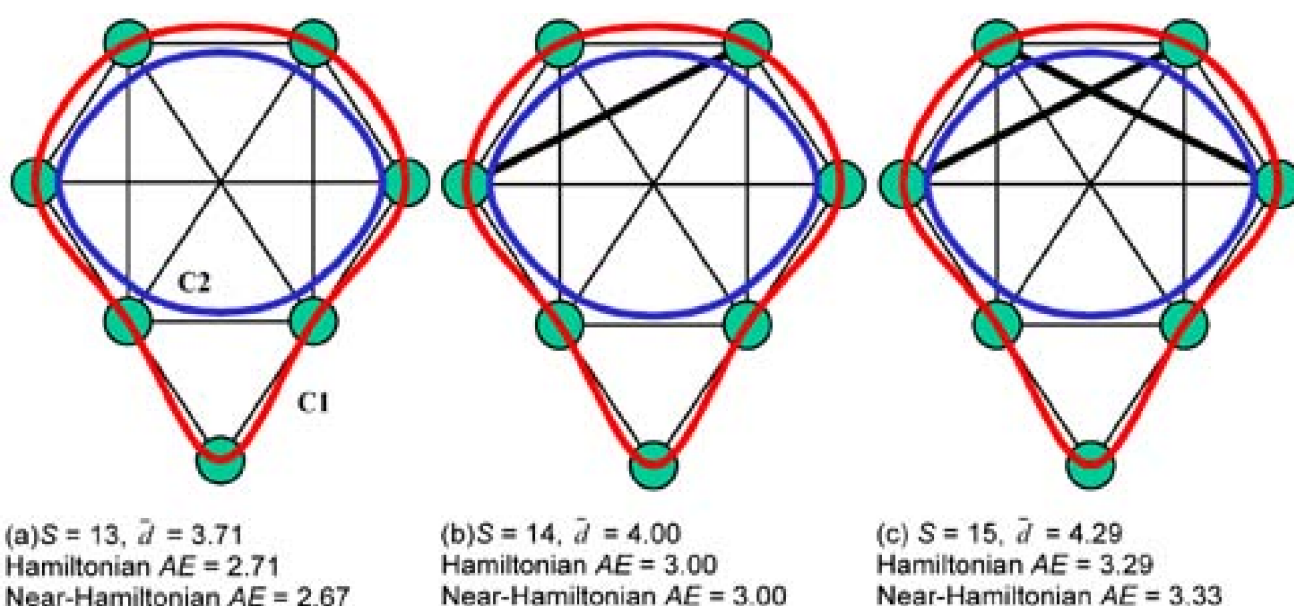
the highest *individual* efficiency (in the *AE* sense) is also always a Hamiltonian *p*-cycle. If true, this would lend itself to a simple search procedure to find a single minimum cost *p*-cycle to protect an entire network (or designated subnetwork). However, it can be demonstrated that this will not always be the case. Compare a Hamiltonian cycle C1 to the "best" non-Hamiltonian cycle C2 possible in the network, where *AE* is the figure of merit. We can let the latter cycle be just one hop shorter than C1 and so visits just one less node than the Hamiltonian. The best case for C2 is that the excluded node is a degree-2 node. We can then observe that C2 has one fewer on-cycle span (from being one hop shorter) and one fewer straddling span (because a total of two spans lose protection). The *AE* of the Hamiltonian (with *N* on-cycle spans and *S-N* straddlers) is thus $[N + 2(S - N)]/N = (2S - N)/N$. For the near-Hamiltonian possessing *N*-1 spans on the cycle and *S-N*-1 straddlers, *AE* is $(N - 1) + 2(S - N - 1)/(N - 1) = (2S - N - 3)/(N - 1)$. Thus, the Hamiltonian cycle has a higher individual efficiency when:

Equation 10.49

$$\frac{(2S - N)}{N} > \frac{(2S - N - 3)}{(N - 1)}$$

Substituting $S = (N\bar{d})/2$ and simplifying, we are left with $\bar{d} < 4$. This indicates that for $\bar{d} \geq 4$ there may exist a cycle with *AE* score greater than or equal to that of a Hamiltonian. To illustrate this, Figure 10-40 shows an example network and two modified versions. In Figure 10-40(a) we can see that the degree is less than 4 and the Hamiltonian has the highest *AE*. In Figure 10-40(b) where a span is added to make \bar{d} exactly 4, the *AE* scores of the two *p*-cycles are identical. When a second new span is added in (c) to increase the average nodal degree even further, the shorter, near-Hamiltonian *p*-cycle has the highest *AE* value.

Figure 10-40. A Hamiltonian cycle is not always the best *p*-cycle.



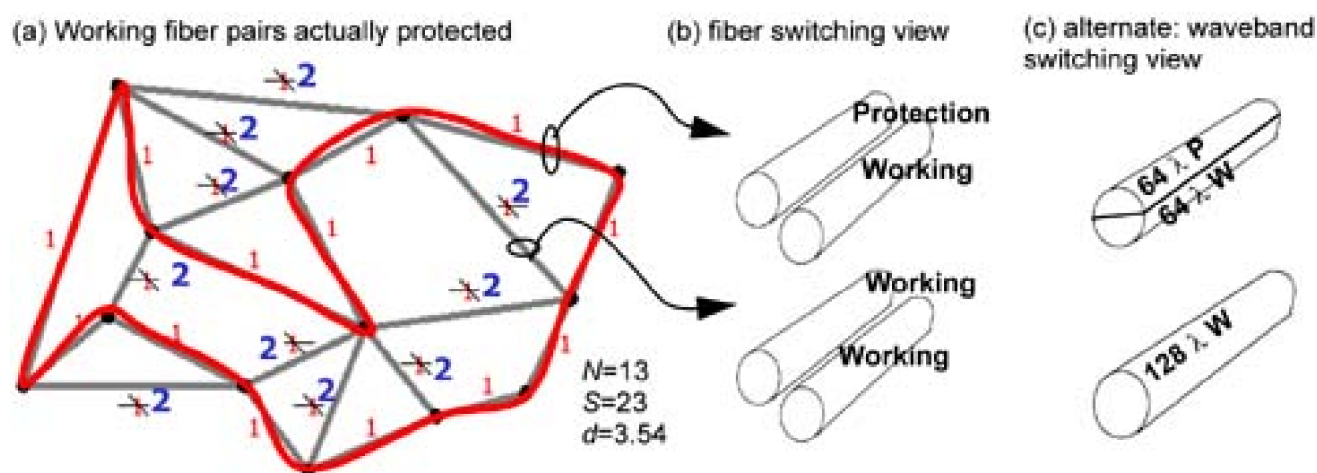
Note, however, that in all cases the single Hamiltonian *p*-cycle remains a better *global* design choice because the near-Hamiltonian C2 does not cover the network entirely, implying that a second *p*-cycle is still required for full protection which would indeed reduce the overall *network AE* score below the score with a single Hamiltonian. Thus, a strategy based solely on finding the cycle with maximum *AE* will not always result in a Hamiltonian. So, even in a homogeneous (and Hamiltonian) network, more than one *p*-cycle may be required for a minimum cost design.

10.10.5 Semi-Homogeneous Networks Inspired by *p*-Cycles

Let us now return to the observation that logical redundancy of 53.8% for the capacitated design problem is *below* the prediction of $2/\bar{d}$. The reason this happens is that the differentially capacitated situation allows some of the p -cycles to take advantage of the 2-to-1 ratio of working capacity on straddling spans, relative to the p -cycle capacity. As explained in deriving [Equation 10.48](#), this is an advantage that the strictly homogeneous network model fails to exploit. Indeed, it makes us realize that a special class of p -cycle based survivable networks is possible that are homogeneous but for one exception: anywhere needed, a straddling span can have two fibers (or channel sets).

The point is illustrated in [Figure 10-41](#) which is based on the Hamiltonian p -cycle of [Figure 10-39](#) for the strictly homogeneous case, but in which we revise the (permissible or actual required) working fiber counts from the homogeneous case to the maximums that can actually be handled by the single Hamiltonian p -cycle involved. This semi-homogeneous case is where the Hamiltonian p -cycle actually *realizes* $1/(\bar{d}-1)$ redundancy, which for this case is the impressively low value of 39.4%.

Figure 10-41. The semi-homogeneous network of two capacity levels actually supported by a Hamiltonian p -cycle.



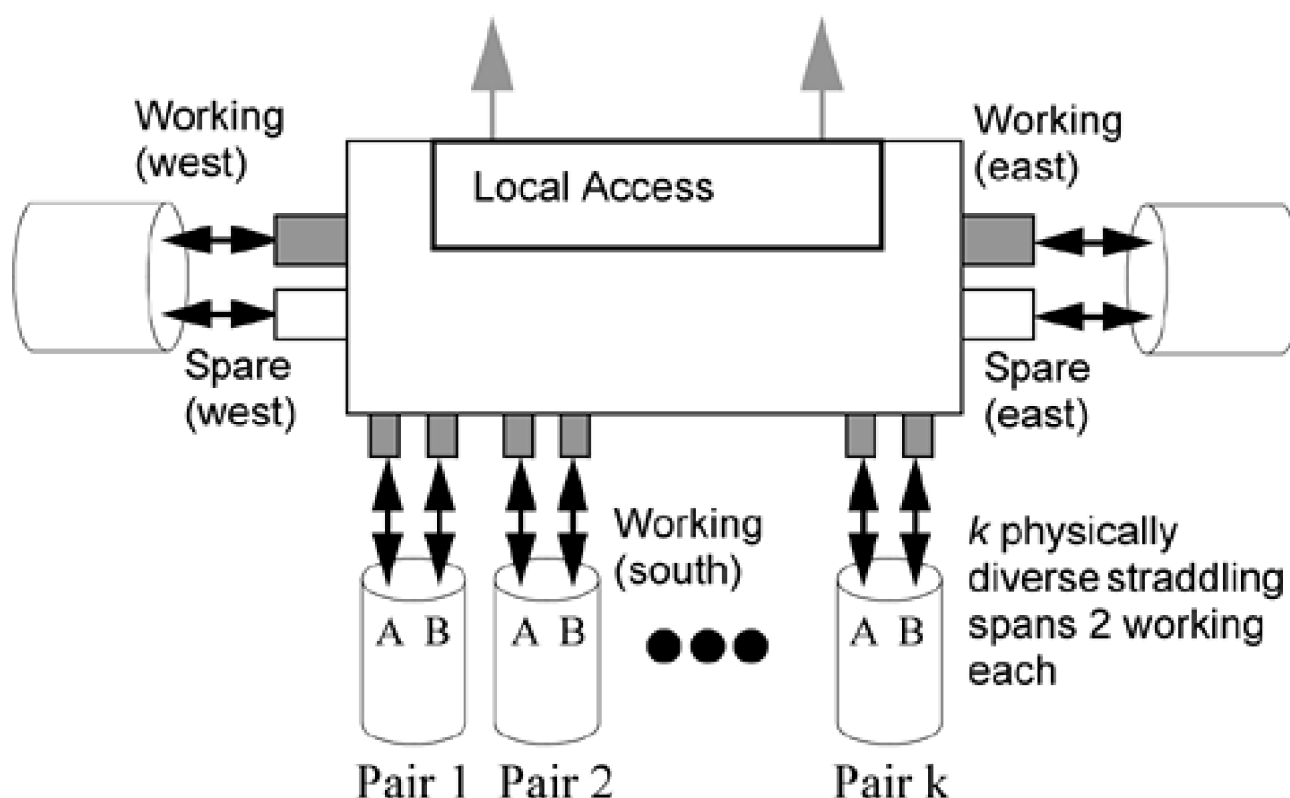
The Hamiltonian p -cycle shown has 13 spans but now protects a maximum of up to $13 + 2(10) = 33$ working fibers (giving 39.4% redundancy). [Figure 10-41\(b\)](#) shows a blow up view of what is actually implied if the p -cycle is implemented at the fiber-switching level. One fiber (pair) is devoted to the p -cycle and each straddling span can have up to two fibers devoted solely to bearing working channels. This could be called a *semi-homogeneous network* because spans either have only one or two fiber pairs for working capacity. It is in the working capacity that homogeneity is usually defined and in which this arrangement is semi-homogeneous. However, because of the two times efficiency of p -cycles with respect to straddlers, this actually makes the physical fiber counts constant everywhere at two fiber pairs per span. Note, however, that although this results in four fibers per span, it is not the same as prior "four-fiber networks" (e.g., Generalized Loopback Networks or cycle covers) because here, two out of two fiber pairs are *working* on straddling spans and 1 out of every 2 pairs on other spans are also working fibers. This is in contrast to one out of two pairs being for protection on every span. An alternate implementation technique in (c) would keep the fiber counts at one pair per span but use straddling span fibers wholly for working demand and on-cycle fiber pairs half for protection and half for working.

Thus, building from the concept of homogeneous networks, we see that the p -cycle technique lends itself in a defining way to the concept of *semi-homogeneous* networks that can have one or two units of capacity on their straddlers, but exactly one on each on-cycle link. This can be viewed as providing a future-proofed form of homogeneous network in that unexpected growth would have a higher probability of being accommodated through routing that exploits the double-capacity option of the straddling spans. It is also of practical and theoretical significance that such a network has a capacity efficiency that exactly *realizes* the $1/(\bar{d}-1)$ lower bound of redundancy for any span-restorable mesh network.

10.11 An ADM-like Nodal Device for p -Cycles

As so far described the p -cycle concept is amenable to realization in a DCS-based network environment. It is interesting, however, to look at the p -cycle equivalent to an ADM node. The basic case follows from [Figure 10-42](#) that it would be ADM-like in terms of having two optical line interfaces (each 50/50 working/spare) and a local access tributary add/drop interface. But unlike a ring ADM it would also provide *two* additional line-rate working interfaces on its "south" or straddling-span face. In fact, it may have 2, 4, 6 or more working ports on the straddling-failure side if these correspond to 1, 2, 3 or more physically diverse straddling spans (so they fail independently). The resulting generalized nodal device is portrayed in [Figure 10-42](#).

Figure 10-42. ADM-like nodal device for p -cycle-based networking.



Failure may be sustained on any two such "straddling side" interfaces by switching their payload signals into the respective east and west direction spares. The "straddling" face is comprised of *pairs* of working line rate signals because the device provides shared protection access to the two halves of the respective p -cycle on which it resides. Typically the expected failures would be associated with each other on the same physical span which has undergone a failure, but strictly any two single working line rate failures, over the set of straddling spans, can be supported simultaneously.

This combination of properties puts the device in a unique middle ground in terms of a networking element architecture between ring ADMs and full-blown digital cross connect (DCS) systems. It is like an ADM in that it is a buy-as-you-need capacity-slice building block. This is one of the benefits of rings over DCSs which are large complete switching systems interfacing all the transmission capacity arriving at a node. But unlike an ADM, they have a nodal redundancy that reflects their network context. In a nodal site of degree $d=2$, a conventional ADM is the only choice. In higher degree sites, however, devices like this may support up to $(d-2)$ "straddling side" interfaces. Thus whereas an ADM has redundancy $R = 100\%$, a p -cycle node device would have $R = 2/(2 + 2k) = 1/(k + 1)$ where k is the number of straddling physical spans interfaced at the site. A $d=5$ site, for instance, could then have an individual nodal redundancy as low as 25% (i.e., $k=3$).

Figure 10-43 and Table 10-9 explain the internal switching arrangements that the device needs to support. For simplicity and generality the "S" circles represent the internal connection to the source/sink of the line-rate signal that is being protected. For instance S_3 identifies the source/sink of whatever working signal path was traversing straddling span W_3 before any failure. The signal loading W_3 may for instance have originated internally from the add/drop interface subsystem of the particular node under view, or it may have originated elsewhere in the network and in normal circumstances have been connected to W_3 from W_4 on some overall route through the network. The example also portrays the case for only a single straddling span but equipped with both possible line-rate working interfaces. $Alarm_N$ indicates the failure state for working line system N. It is high if there is an alarm condition; low if operating normally. Selectors Sel_1 , Sel_2 are actually bidirectional signal selectors/distributors in the bidirectional context shown. In the normal (no failure) condition a connection path is maintained between the two protection ports. This is so that when a failure occurs adjacent to some other node on the p -cycle, not the node shown, the protected signals flow transparently through the unalarmed nodes between the ends of a span failure. Basically the arrangements shown are set up to effect the following switching actions in response to any single span failure (in which case W_3 W_4 fail together) or any single fiber-system failure (in which case W_3 W_4 may fail individually).

Figure 10-43. Internal switching functions of ADM-like nodal device for p -cycle-based networking. (Illustrated for one straddling span).

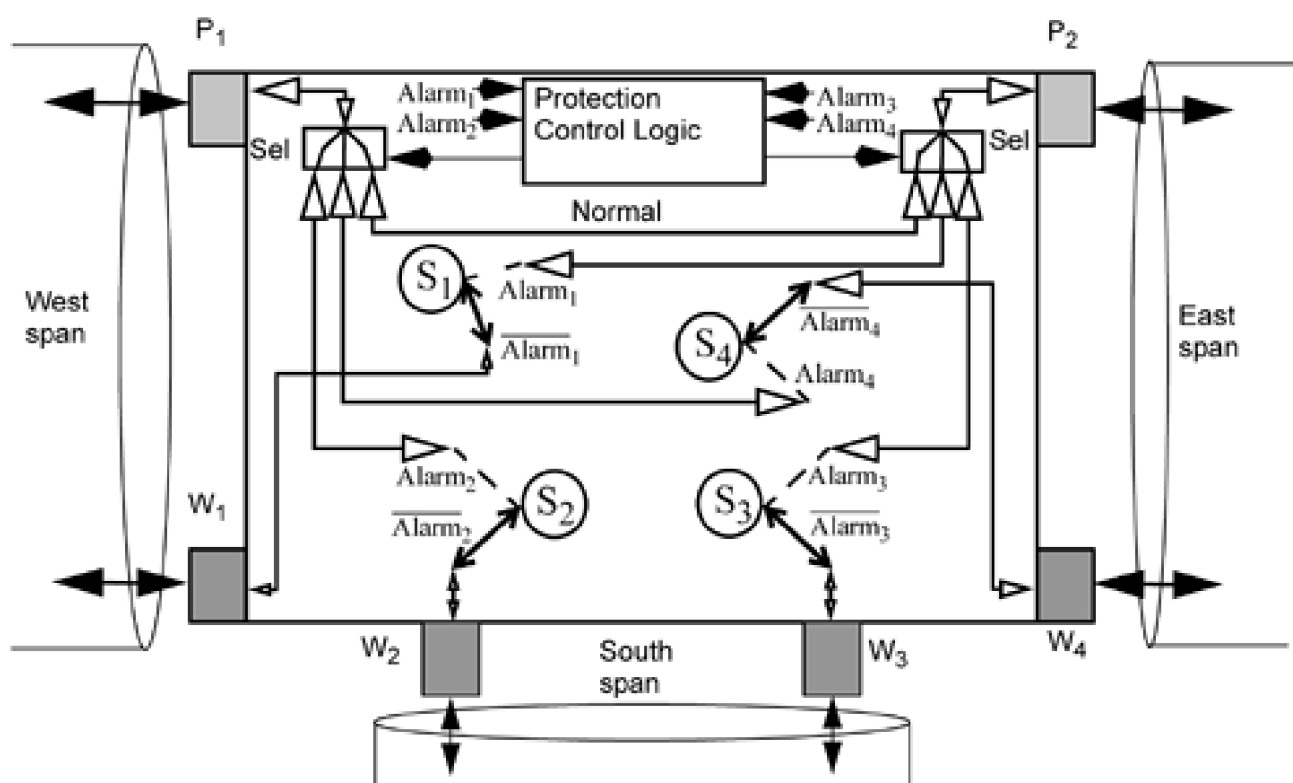


Table 10-9. The Predefined Switching Logic of the ADM-like p -Cycle Nodal Element.

Physical Span failure	Causes Working Failure on	Predefined Switch action (to / from)	Comment
West	W ₁	W ₁ --> P ₂	BLSR-like "loopback to protection" in other direction. Port P ₁ is assumed failed with W ₁
East	W ₂	W ₂ --> P ₁	BLSR-like "loopback to protection" Port P ₂ is assumed failed with W ₂
South (straddling)	W ₃ and W ₄	W ₃ --> P ₁ W ₄ --> P ₂	Identical to BLSR switching into each protection span direction, but from straddling port sources, not opposite side port as in normal ADM.

In [Chapter 11](#) on ring-mesh hybrids and ring-to-mesh evolution, the description given here of an ADM-like nodal element for p -cycles is complimented (in [Section 11.8](#)) by explanation of how the overall function described above can be emulated by addition of a *Straddling Span Interface Unit* (SSIU) to an existing ring ADM, allowing conversion of existing rings into more efficient sets of "capacity-slice" p -cycles. This is part of a strategy of supporting ongoing growth in network demand without adding new capacity while converting from rings to either a mesh or p -cycle target architecture that is the theme of [Chapter 11](#).

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

10.12 Self-Organized p -Cycle Formation

With methods of this chapter we can obviously conduct centralized design and configuration of a p -cycle based network. But a promising concept to support a dynamic demand environment is a self-organizing strategy for continual adaptation of a set of network-protecting p -cycles. The *Distributed Cycle Preconfiguration* (DCPC) protocol is an adaptation of the statelet processing rules of the Self-Healing Network (SHN) protocol ([Gro97](#)). A *statelet* is embedded on each spare channel (or a designated signaling channel for the link as a whole) and contains a number of semi-static information fields (like the K1-K2 bytes in the SONET APS protocol). Each logical channel (any managed capacity unit), as viewed by a node attached to it, has an incoming statelet and outgoing statelet. An incoming statelet arrives at a node on a channel, and originates from the adjacent node connected through the channel. As in the SHN, each outgoing statelet has an incoming statelet which forms its precursor and there is no nodal database of network topology nor is there any centralized coordination required other than long-term observation and capacity augmentation for growth.

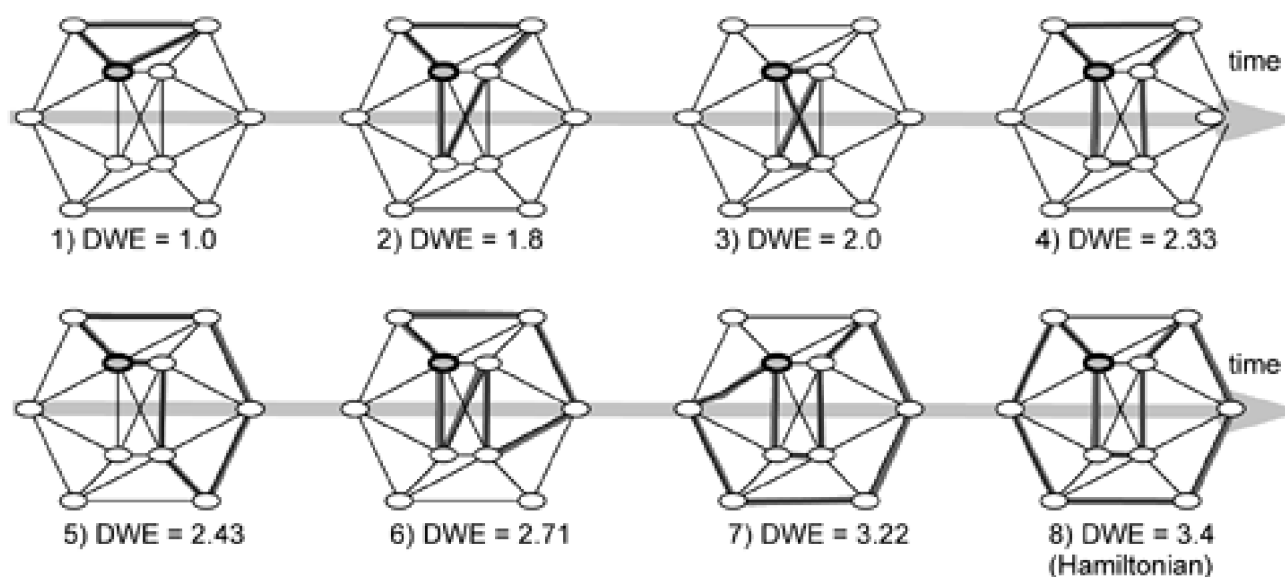
The DCPC protocol is a set of event-driven rules for nodal interaction via such statelets. All interactions happen within the spare capacity of the network only and are all in non-critical time before any failure has arisen. At the large scale DCPC first allows each node to explore the network for p -cycle candidates that are discoverable by it from its location in the network. After completion of its exploratory role as "cycler node" (below), it hands off to the next node by a one-time "next-node hand-off" flood-notification. After all nodes have assumed the role of the cycler once, each "posts" its best-found cycle candidate in a distributed network-wide comparison of results. In this step all nodes hear the performance metric, and other details, of the globally best p -cycle candidate discovered by any of their peers. The competition flood expands through the network as each node locally relays the statelet with the best metric, or asserts its own best if and while the latter is superior to anything else it has yet received notice of. Eventually, the globally best cycle candidate dominates, everywhere. Upon learning of the winning candidate, the cycler node who discovered this p -cycle candidate goes on to trigger its formation. All nodes on the new p -cycle update their local store of restoration switching information to exploit the new p -cycle. The whole process repeats, spontaneously, adding one p -cycle per iteration until a complete deployment of near-optimal p -cycles is built. Thereafter, it continually adapts the p -cycle set to changes in the working capacity layer.

All statelet broadcasts originate at the current cycler node. To initiate the cycle exploration process, the cycler places an outgoing statelet on one spare channel in each span at its site (or signaling can be adapted to a common per-span signaling channel such as an OSC). Each of these primary statelets has a unique index number. After the primary statelet broadcast, the cycler node invests a predetermined time in sampling of the returning statelets. The Tandem node rules determine what p -cycle candidate the cycler node will discover in a given round of global cycle comparison and formation. A tandem node will broadcast each incoming statelet to the largest extent warranted by the statelet's *numpaths* score within the context of the available outgoing channel resources and other statelets currently present at the node. If all outgoing spare channels on a span are occupied, a new incoming statelet can displace an outgoing statelet if it has a *numpaths* score better than the precursor with the lowest current score. Statelets on a given index can also only be forwarded to adjacent nodes which are not already present in the accumulating route of the corresponding precursor. The single exception to this rule is that a statelet may be broadcast from a tandem to its own cycler node, which is present in all route fields. These rules, and a hop limit, restrict the process to consider only simple cycles of allowed sizes.

The emergent effect of the rules (more fully detailed in [[GrSt98](#)] [[GrSt98b](#)]) is that, shortly after triggering the process, a cycler node receives incoming statelets whose route fields trace out cycles that begin and terminate at the cycler node. As returning statelets arrive, the cycler maintains a record of the received statelet with the best score, defined as the ratio of accumulated protection relationship to channels consumed. It also records the route information accumulated on that statelet. The cycler persists in observing the incoming statelets because a cycle tends to provide a higher number of useful paths as it is allowed to evolve under the collective interactions of the tandem nodes. Usually it grows in size as it improves its score, but this is not always the case because unprotected working capacity must remain on each protected span for the prospective p -cycle to get credit for proving a protection relationship for the span. So after a few p -cycles have already been formed subsequent p -cycles will conform more to the remaining capacity distribution than to the topological temptation to grow all the way into a Hamiltonian (if possible on the graph). Although the terms were not used at the time, the DCPC "score" can be either what we now call the demand-weighted efficiency (DWE) of [Section 10.7.1](#), or just the demand-weighted TS measure. The process of self-organizing evolution of each p -cycle stops when hop count or available working capacity for protection prohibits further improvement. [Figure 10-44](#) is an illustration, from DCPC simulations, of how one prospective p -cycle evolves—improving its score with time. The cycler node for the example is the shaded node. The example stops changing after about a second when the Hamiltonian at step 8 arises. At that point the DCPC node rules stabilize because, although they do not see the overall picture, they will not find any incoming statelet with score measures higher than those they are currently promoting to the available spare capacity output channels, so everything stops. About another second later the cycler node senses the stabilization and records the p -cycle route and ultimate score. It will then "hand off" to another node to function as cycler. Note that as a background self-planning process (not itself

directly a restoration process), DCPC speed is not critical.

Figure 10-44. Example of the real-time self-organizing evolution of p -cycles under DCPC.



DCPC performance is measured in terms of the restorability level it achieves through self-organized p -cycle creation. A set of results obtained by OPNET simulation is given in [Table 10-10](#). The test networks are efficiently modularized span-restorable mesh spare capacity designs. Where "non-mod" is indicated, the restorability level is 100% prior to the modularization. In the other cases the DCPC-achieved restorability level was less than 100% (though over 90%) in the non-modular minimal spare capacity designs, then rose to the levels shown when the small extra amounts of unused capacity due to modularity were added. It is stressed that these test results are achieved within the theoretically minimum amounts of feasible spare capacity. A real network would generally have more margin of spare capacity simply due to modularity and fill levels lower than those assumed for the planning horizon for which the nominal design was produced. The DCPC process inherently also serves as an ongoing self-audit on the restorability margins within any network and can thereby generate the inputs needed for ongoing provisioning of new capacity as growth occurs, to always retain 100% restorability.

Table 10-10. Performance of DCPC in self-organizing p -cycles for restorability in minimal-spare capacity test networks

Network	Modularity	DCPC p -cycle Restorability
Net1	non-mod	100 %
Net2	non-mod	100 %
Net3	OC-24	91.5 %
Net4	OC-48	100 %
Net5	OC-12	95.1 %

10.13 Virtual p -Cycles in the MPLS Layer for Link and Node Protection ^[10]

[10] This section is partly based on material previously published in [StGr00](#).

IP/MPLS networks are already restorable in the sense that OSPF type routing protocols will, through dissemination of link-state and route advertisements, eventually update the routing tables for IP forwarding and GMPLS supports reprovisioning paths that are lost, or shared-backup MPLS path arrangements. Some of these methods—OSPF reconvergence or MPLS redial—are, however, slow processes compared to what we have in mind for transport protection. Shared-backup MPLS may be much faster but has no direct regard or ability to foresee the for capacity congestion effects that arise from mass activation of MPLS backup paths arising from a single cable cut. p -Cycles provide an interesting additional option for use in the IP/MPLS layer to give greater speed and certainty of restoration response and compliment the normal recovery mechanisms of the IP layer.

The initial approach to use p -cycle ideas for more rapid IP or MPLS restoration uses them in the same logical way that they would be used for a SONET or WDM network but we adapt the implementation to either a pure IP routing or MPLS environment and we adopt a controlled oversubscription framework for capacity design. In a second step we introduce the concept of *node-encircling p -cycles* for network protection against router node failure. In both contexts the p -cycles are envisaged as the "fast" part of a "fast plus slow" process of overall recovery. The ordinary routing update and path establishment protocols still proceed as usual at their natural pace following a failure. But in the real-time interval before these processes reconverge, p -cycles act much more quickly to minimize traffic loss. When the slower routing update processes do converge, most or all of the affected packet flows inherently return to normal routes and traffic on the p -cycles drops off (back to zero assuming sufficient capacity in the reconfigured IP/MPLS network).

10.13.1 IP Link Restoration with MPLS p -Cycles

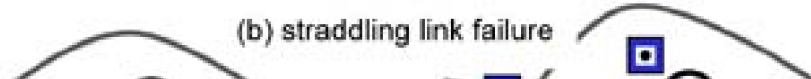
In a pure IP-routed network there is no identifiable spare capacity per se. IP links simply have a high or low utilization relative to their current installed (or allocated) bandwidth. It is therefore not initially clear how to exploit the p -cycle concept with its preconfigured circuit-like logical structures of protection bandwidth. We need some kind of virtual circuit construct to create p -cycles with. MPLS provides exactly such a mechanism although any IP-tunneling technique can support creation and operation of virtual p -cycles. The first class of failure for which p -cycles can offer restoration is the loss of a logical IP link between a pair of adjacent routers. Note that now ~~were~~ talking about logical inter-router *links*, not the physical *spans* of the underlying network as the environment where the MPLS p -cycles are implemented. We will therefore later have to consider the expansion of physical faults into possibly multiple IP layer links.

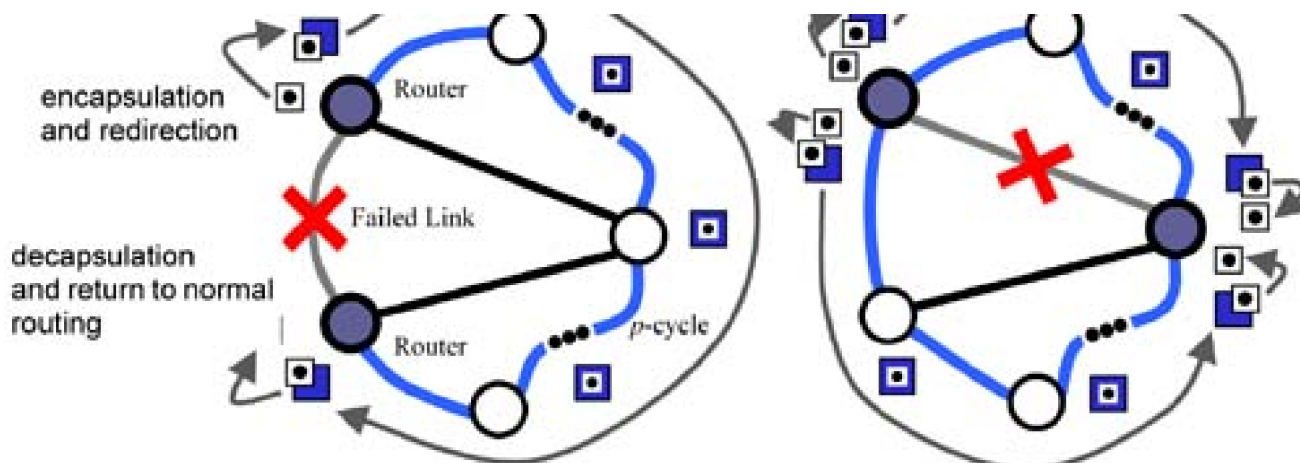
Link failure could be caused by a cable cut or a single interface card on a router. Physical layer failures will, however, appear in the IP layer only if there is no physical layer restoration mechanism or its capability is exceeded for some reason. In general, physical layer restoration of span cuts remains a fast and effective line of defense against span failure and may be preferred in practise over router-based recovery against physical span cuts. We intend our explanation of router-based restoration to be seen primarily as providing protection against link (and later node) failure in the IP/MPLS layer, neither of which can be restored by any reconfiguration in the physical layer.

Restoration of link failures in an IP network with virtual p -cycles operates as follows, with reference to [Figure 10-45](#). The details vary depending on whether the environment is one of pure IP routing or an MPLS environment, but the overall process is the same. Our explanation describes the operation in a generic IP routing environment but it is easy to see how MPLS can be used to support the implementation.

Figure 10-45. p -Cycle restoration of IP / MPLS packet flow affected by a link failure.

(a) an on-cycle failure

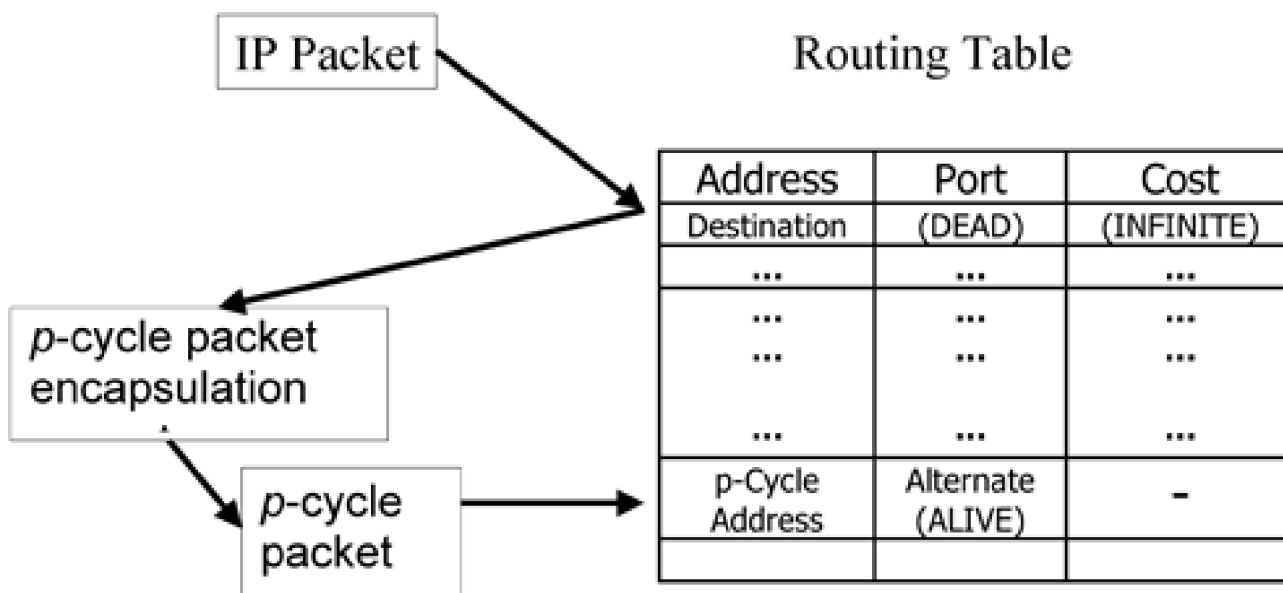




Flow Redirection into the p -Cycle

When link failure has been detected, the router ports which terminate the failed span are marked as dead (and the usual link-state advertisement update process is triggered). Until a global routing update is effected, any packet whose next hop, as indicated by the normal routing entry for the packet's destination address (or its label), would have been directed into the dead port, is, instead, deflected onto a p -cycle which has been assigned to protect the link. "Deflection onto p -cycle" occurs by *encapsulating* the original IP packet in a " p -cycle packet" and reentering the same routing table as illustrated in [Figure 10-46](#). For affected LSPs it is just a special label reassignment or label-stacking operation. When reentering the routing (or label) table, the encapsulating IP address (or new path label) matches a surviving port where a *virtual p-cycle* has been previously established. The packet is forwarded into the corresponding surviving port at the initial encapsulating node and travels through the p -cycle following either label switching or routing table entries at other nodes that have been preestablished on the p -cycle IP address or label sequence.

Figure 10-46. One forwarding cycle redirects packets to a local p -cycle entry for link failure.



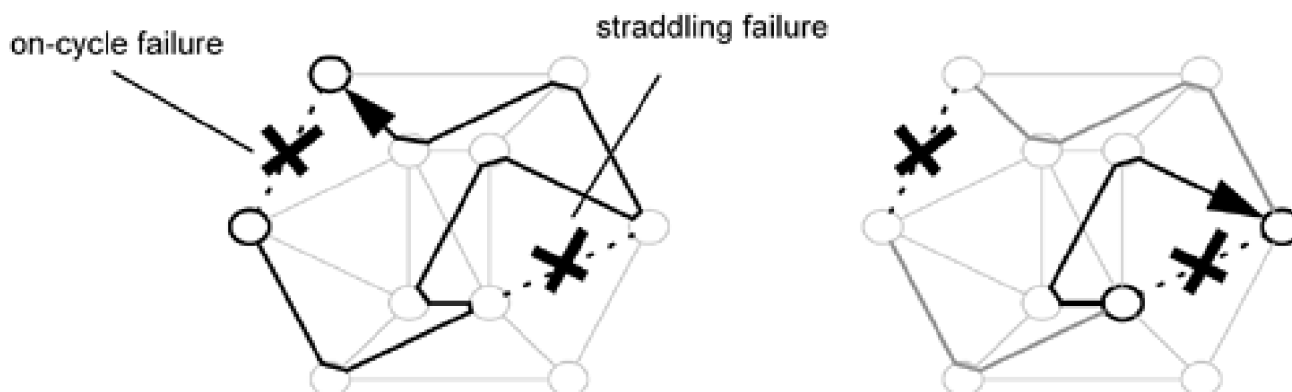
Return to Normal Routing

The p -cycle packets traverse all intervening routers along the p -cycle until they arrive at the router at the other side of the failed link where the original IP packet is decapsulated. Conceptually, the other router knows to do this since the packet bears the encapsulating address of a p -cycle on which the deflecting node is present and there is currently a local port alarm on the IP link to that neighbor. Once decapsulated, the IP packet is routed normally toward its final destination. More generally, however, a return to normal routing can occur at any node on the p -cycle where the continuing route cost of the original IP packet is lower than that where it entered the p -cycle. This is actually the preferred technique, which we discuss in more detail in the context of node-encircling p -cycles.

There are of course two classes of link failure that a p -cycle can restore: failures on a link of the p -cycle itself, and straddling failures. As so far described an on-cycle failure has been considered, as in [Figure 10-45\(a\)](#). [Figure 10-45\(b\)](#) shows a straddling failure situation. The local nodal action for rerouting into and from the p -cycle at the two end nodes is identical for either on-cycle or straddling failures. Straddling failures, however, allow the restoration flow to be split (for instance simply by even/odd IP address or label) in two directions around the failure to distribute the restoration load, reducing oversubscription effects. Although [Figure 10-45](#) and related discussion speaks as if the p -cycle is bidirectional, each virtual p -cycle is actually defined by a pair of routing table or label switching entries—one points to the local port which corresponds to entry of the p -cycle in the clockwise direction, the other is the counter-clockwise use of the same logical p -cycle. Compared to the typical number of entries in a router (now often more than 100,000), p -cycles require at most $2d$ additional entries (one for each direction on the p -cycle) where d is the IP link degree of the node.

Note at this point some of the multiple failure properties of the virtual p -cycle concept. The p -cycle shown has itself been disrupted by an on-cycle link failure, while a straddling link failure is also present. Straddling link failures do no damage to the p -cycle itself, but on-cycle failures "open" the p -cycle into a surviving linear virtual-path segment. In the case of [Figure 10-47](#), a bidirectional MPLS virtual p -cycle can still offer restoration to both failures because it still offers one restoration path in the other direction around each failure. The restoration path for the on-cycle (disruptive) failure utilizes the intact portion of the remaining p -cycle, while the straddling (non-disruptive) failure utilizes only a surviving arc of the remaining p -cycle. Both restoration paths connect the end routers of their respective failures and provide an alternate routing for affected packets. This extra multi-failure tolerance is available to IP/MPLS layer p -cycles because these are only uncapacitated routing options in this argument (and the combined imposition of flows may cause congestion). In the corresponding SONET or WDM context only one of the two failures could be supported at a time because either fully uses the p -cycle capacity (assuming the straddling span has two working channels).

Figure 10-47. Multiple failures restored simultaneously by the same p -cycle.



10.13.2 Network Design using MPLS p -Cycles for Link Restoration

In a WDM - IP network, lightpaths in the WDM layer are used to setup logical connections ("links") between IP routers and/or LSRs. From the perspective of the IP network, these appear to be direction connections between routers, while in reality a single physical span may carry sections of multiple logical links between routers. Failure of a single physical span can then translate into the apparent failure of multiple links in the IP layer. This must be taken into consideration when designing a set of IP network p -cycles, so that any single physical span failure has a controlled or bounded maximum impact on the simultaneous failure of logical links within the same p -cycle.

If the physical to logical fault mapping data is actually available, it can be incorporated in the MIP which follows: where all physical span failures are enumerated, one adds all the corresponding multiple link failure scenarios to the constraints set. However, there is also a simple way to ensure the effectiveness of p -cycles under *unknown* physical to logical fault escalation. That is to restrict IP layer p -cycles to

being formed only from IP links that traverse a single physical span; i.e., form p -cycles using only hops between physically adjacent routers, not all possible logical links between routers. In other words, in the graph of all IP layer links, one formulates the design problem to form the required p -cycles only on the physical layer subgraph of the IP link layer topology. This ensures p -cycles are only formed over logical links that have a 1:1 correspondence to a single underlying physical transmission span. Not only does this reduce computational complexity but it ensures that for any single physical span failure, any p -cycle would suffer, at most, a single logical link failure and so would continue to be able to offer at least one restoration path.

In the IP context, restorability design also needs to consider the convergent flow effects arising from restoration. This is an aspect that does not exist for SONET or WDM where every working signal is either exactly replaced or not. In contrast, where stat-muxed packet or cell flows are being redirected upon restoration, one can take the "oversubscription" based design approach of [Chapter 7](#) to control the worst case simultaneously imposed flows on any link during any restoration scenario. In the following formulation we do this, the aim being to determine a set of IP p -cycles that minimizes the worst case oversubscription factor on any link, over all failure scenarios. This allows the network designer to accept a slightly lowered (but an assured worst case) QoS during a restored network state in return for economic savings, because less capacity is required to provision the network compared to the corresponding 100% restorable SONET or WDM network. This is also an improvement over allowing ordinary IP routing protocols to be used solely for restoration since they do not directly take congestion effects into account at all.

The input parameters and sets are:

- M is the maximum number of p -cycles permitted in the design (a user input ^[11]).

^[11] This parameter may be set to reflect the maximum number of logical p -cycles considered practical to administer.

- C_j is the total capacity allocated to link i .
- P is the set of candidate p -cycles (formed only over links with physical span correspondences).
- S is the set of IP links in the network.
- w_i is the amount of working traffic flowing through link i during normal operation.
- $b_{i,j,k}$ is a precomputed "imposed load" ratio corresponding to $x_{i,j}$ coefficients for previous discrete-circuit p -cycles. Each $b_{i,j,k}$ is a constant that gives the fraction of the working flow from link i that is carried on link j if using cycle p for restoration. It can either be 0, 0.5, or 1. It is zero if cycle p does not pass over link j , 0.5 if the p -cycle offers two restoration paths to link i and traverses link j (the traffic is split), and 1 if p -cycle p offers only a single restoration path for link i and also crosses link j .

The primary decision variables are:

- d_p is a 1/0 decision variable which is 1 if cycle p is used in the design, 0 otherwise.
- $a_{i,p}$ is a 1/0 decision variable which is 1 if cycle p is assigned to the restoration of network link i and 0 otherwise,

Two intermediate variables are produced which give useful diagnostic data about the design solution, but are actually wholly dependent real-valued terms computable from the primary variables and the model parameters. (With suitable substitutions they can strictly be eliminated from appearing explicitly in the model if desired, but it is more instructive to employ them.):

- $X_{i,j}$ is the oversubscription factor on link j during the restoration of link i .
- h_M is the maximum oversubscription ratio on any link during any restoration event.

The design model is:

Minimize

Equation 10.50

$$\eta_M$$

subject to:

1. No more than the stipulated number of design p -cycles:

Equation 10.51

$$\sum_{p \in P} \delta_p \leq M$$

2. Failure i may be restored by the p^{th} cycle only if that cycle is chosen as a p -cycle:

Equation 10.52

$$\alpha_{i,p} \leq \delta_p \quad \forall i \in S, \forall p \in P$$

3. Any chosen cycle must be used for at least one span failure:

Equation 10.53

$$\sum_{i \in S} \alpha_{i,p} \geq \delta_p \quad \forall p \in P$$

4. Every link failure must be restored by a single p -cycle:

Equation 10.54

$$\sum_{p \in P} \alpha_{i,p} = 1 \quad \forall i \in S$$

5. Definition of the oversubscription factor on link j when link i fails:

Equation 10.55

$$X_{i,j} = \frac{w_j + \sum_{p \in P} \beta_{i,j,p} \cdot w_i \cdot \alpha_{i,p}}{C_j} \quad \forall (i,j) \in S^2 | j \neq i$$

6. "Ceiling" type of constraint for minimization of the peak oversubscription:

Equation 10.56

$$\eta_M \geq X_{i,j} \quad \forall (i,j) \in S^2 | j \neq i$$

The objective function, η_M , is the maximum oversubscription ratio on any link during any restoration event. Equation 10.52 allows a link i to use cycle p for its restoration (as decided through the binary decision variable $a_{i,p}$) only if cycle p is used in the design (as decided by variable d_p). Equation 10.53 permits a cycle to be used in a design only if it is conversely used in the restoration of at least one link. Equation 10.54 permits any link failure to use only a one p -cycle in its restoration. Equation 10.55 effects the measure of restoration-induced oversubscription for all links, for each other link's failure by summing the link's normal working flow with all the other fractional (or whole) restoration flow(s) imposed on the link through p -cycles that traverse it. Finally Equation 10.56 links all the oversubscription ratios to the objective function which is being minimized. In this way, the $\max(X_{i,j})$ is minimized.

For research purposes we obtained optimal solutions for OS PC-1 in three networks having $(N,S) = (10,22), (15,28)$ and $(20,31)$, respectively. Each test network was provisioned with link capacities that just met the working demand requirements of the shortest-path mapped demand matrix plus an amount of excess (or "spare") capacity given by a SCA for a span-restorable mesh. In other words, the formulation is being challenged to minimize the peak restoration-imposed oversubscription effect on any link, with varying numbers of p -cycles allowed, when all links have no more than the theoretical minimum capacity required for a corresponding span-restorable mesh. The results were obtained with no hop limit on the size of cycle which could be used as p -cycles and the formulation was run with all simple cycles of the graph as p -cycle candidates. Table 10-11 shows the maximum oversubscription ratios on any link over all restoration events, in dependence on the number of virtual p -cycles the design is allowed to use. As the number of p -cycles is increased, the restoration-state flows closely approach the performance of a conventional mesh-restorable network in that 100% link restoration is achieved with almost no oversubscription-related impact on any other working flows of the network (i.e., oversubscription ratio of 1.0). For illustration, the optimal set of five p -cycles for Net 2 is shown in Figure 10-48. These five p -cycles offer 100% link restorability with a peak restoration-induced oversubscription factor of 20%. Thus, application of p -cycles to IP restoration could ensure QoS limits under restoration without requiring more capacity than does a span-restorable mesh network with the same links. However, related formulations, similar to those of Section 7.9 on oversubscription-based SBPP design can also take a maximum oversubscription factor as an input and minimize the network total capacity while satisfying this limit on worst-case restoration impact. Capacity versus worst-case oversubscription trade-offs similar to those of Table 7-4 result. Such designs deliberately exploit oversubscription to require less "excess" capacity than the corresponding SCA design has in terms of explicit spare capacity.

Figure 10-48. Five p -cycles for 100% IP link restorability with minimum oversubscription.

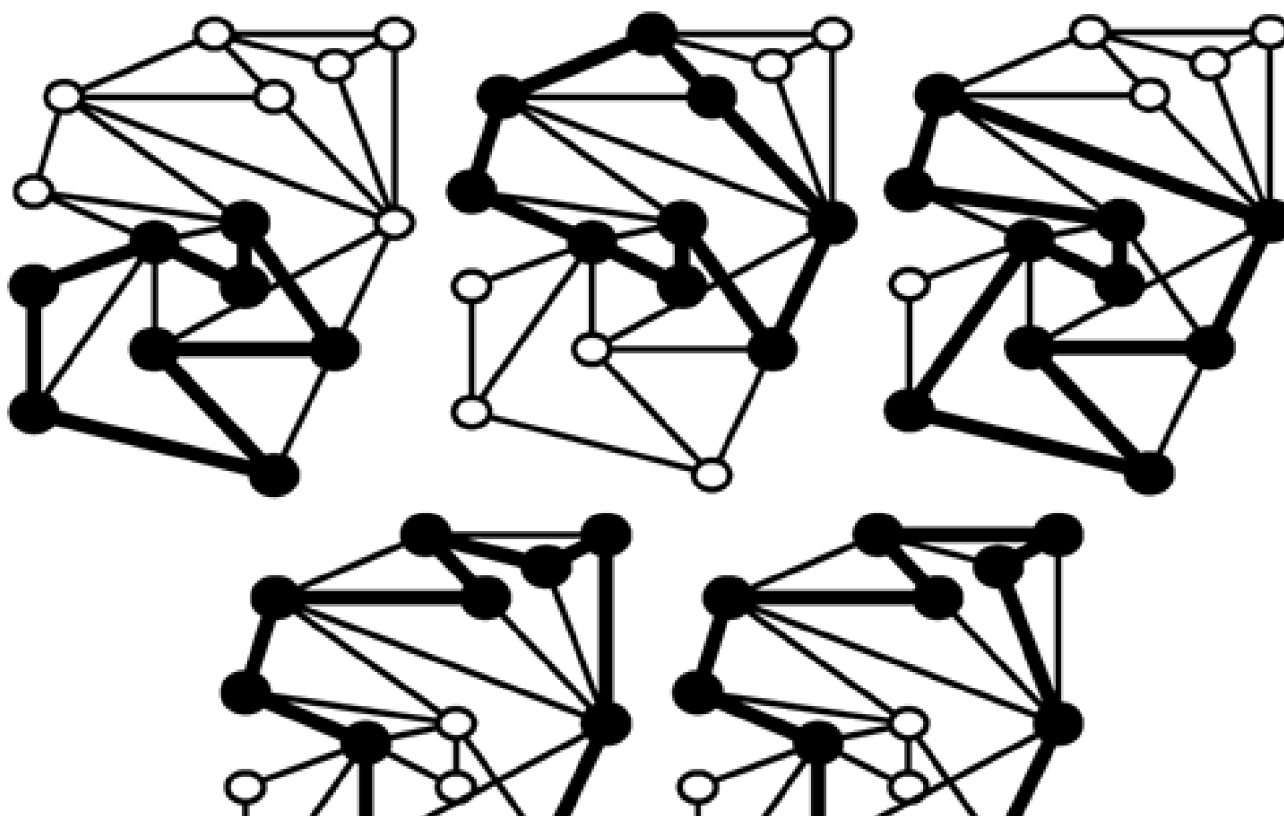




Table 10-11. Results with virtual p -cycle IP link protection design

# p -cycles allowed in design	Maximum restoration-induced oversubscription factor		
	Net 1 (10,22)	Net 2 (15,28)	Net 3 (20,31)
1	1.625	Infeasible [a]	Infeasible [a]
5	1.0447	1.2	1.097
10	1.0417	1.045	1.0449
15	1.0328	1.019	1.0277

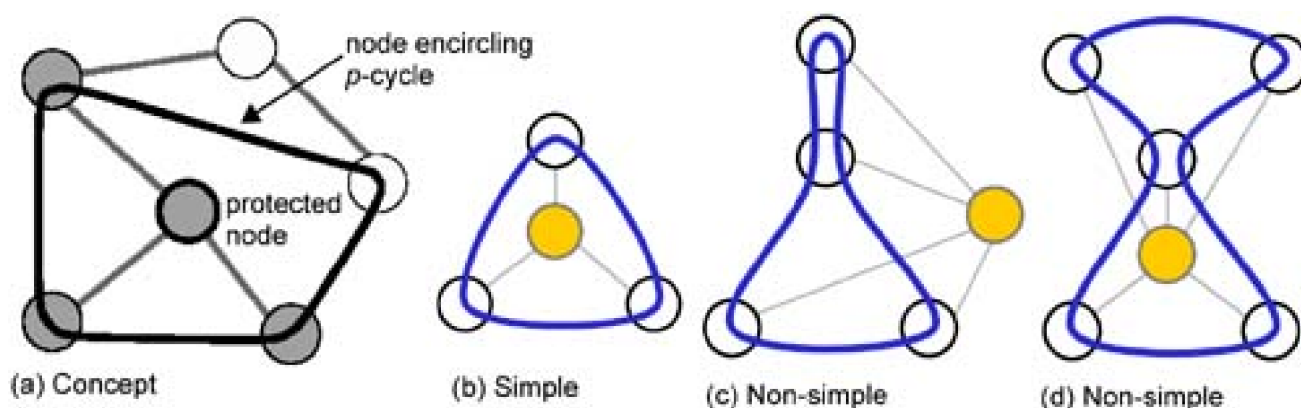
^[a] These two networks are not Hamiltonian so one p -cycle is infeasible for link protection regardless of oversubscription factor.

10.14 Node-Encircling p -Cycles for Protection Against Node Loss

Unlike experience with SONET or WDM transmission networks, IP networks reportedly suffer "node failures" about as frequently as link failures. Usually these are not real failures such as power loss or fire. Router restarts for software patches/upgrades or software crashes apparently generate the majority of router outages. In principle, existing IP routing protocols disseminate the news of the routers disappearance through link state advertisements by all adjacent nodes of the failed node, and the network as a whole converges to a revised routing plan. In practise, however, the volume of link state update flooding messages, the time required for complete reconvergence, and the resulting congestion effects could all be improved upon by a much faster and more localized response to router outage. Node-encircling p -cycles can provide such rapid protection of *transiting* flows affected by a node outage. Traffic terminating at the failed node itself is inherently unrestorable by any type of network rerouting.

A node-encircling p -cycle can protect against router failure by providing an alternate path among *all* of the routers that are *adjacent* to the failed router as [Figure 10-49](#) illustrates. Thus, each node encircling p -cycle can provide a readily available replacement detour path for up to $n(n-1)/2$ router-pairs, where n is the number of routers adjacent to any node considered as a prospective failure node. Thus, for a p -cycle to provide restoration for all of the transit flows affected by a router failure, *it must contain all of the routers adjacent to the failed router, but not the failed router itself.*

Figure 10-49. Basic concept and canonical topologies for node-encircling p -cycles.



In effect a node-encircling p -cycle constitutes a kind of perimeter highway or control volume that is assured to be intersected (by definition) by all transiting flows that may be affected by the given node's failure. By virtue of this property it is also in a position to provide a substituting detour between all adjacent node pairs that may have been exchanging pre-failure flows through the lost node. A node-encircling p -cycle must contain *all* topologically adjacent routers as, otherwise, it cannot substitute routes for all of the possible pre-failure flows that traversed them via the failure node. But the p -cycle must *not* contain the router it is protecting, so that it is not itself disrupted when the router fails. These are the properties of what we call a "node-encircling p -cycle."

10.14.1 Types of Node-Encircling p -Cycles

A p -cycle, which protectively encircles a node, must be constructed within the subgraph that results when the protected node (and all its incident links) is itself removed from the network. It may or may not be possible to form a *simple* cycle on the set of nodes adjacent to the failure node within the resulting subnetwork. A simple cycle, illustrated in [Figure 10-49\(a\)](#) and [\(b\)](#) crosses each node only once. There is however, always a logically encircling p -cycle construct that is possible if two special considerations are dealt with assuming only that all pre-failure graphs were at least two-connected. [Figure 10-49\(c\)](#) and [\(d\)](#) give examples of the non-simple p -cycles that can result when protecting routers. [Figure 10-49\(c\)](#) also points out that a logically encircling p -cycle does not necessarily encircle the protected node

physically. With a simple cycle, removal of the protected node does not disrupt the overall two-connectedness of the resulting network. In such cases the node encircling p -cycle is visually apparent and easy to find. In the second example [Figure 10-49\(c\)](#), removal of the protected node results in a singly connected remaining network. The degree-1 node can only be included in the encircling p -cycle through a segment that the cycle passes twice. The last example (d) results when removal of the protected node creates a subnetwork with a bridge node; that is, a node whose removal would disconnect the graph. Here the logically encircling p -cycle still exists but takes on a figure "8" form, as it is forced to pass twice through the bridge node in the surviving graph. Note that in any of these cases a node-encircling p -cycle may have to visit nodes that are non-adjacent to the protected node, in order to form a cycle that does include all its adjacent nodes. This is the case in [Figure 10-49\(a\)](#). In the worst case, a network of N nodes would be fully protected against any single router outage by establishment of N node-encircling MPLS p -cycles. Each node would have a logically encircling p -cycle established for protection of its transiting flows through the set of its immediately adjacent nodes.

10.14.2 Rerouting Mechanism with Node-Encircling p -Cycles

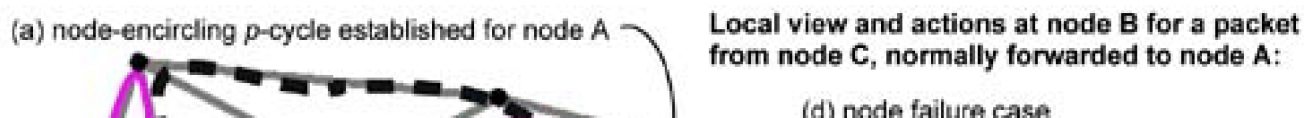
First, let us briefly consider the problem of rapid detection of a router failure. If the node failure is some kind of catastrophic physical damage or outage, then link failures are associated with the node loss and rapidly detected by neighbors. In this failure model, the node is physically destroyed, or loses all power, so that physical-layer link interfaces at neighbors generate loss of signal or AIS-detection alarms right at the hardware level. The more difficult, but more frequent, type of router failure is a silent or software-related failure: the node is not physically damaged or unpowered, and all its physical link interfaces are operating properly, but the router simply stops forwarding packets. With conventional methods this type of failure will take four missing "hello" packets (at 10 second intervals) for neighbors to detect the router loss.

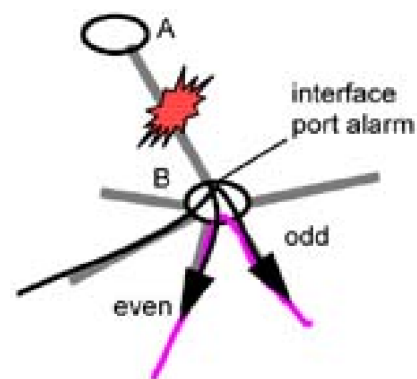
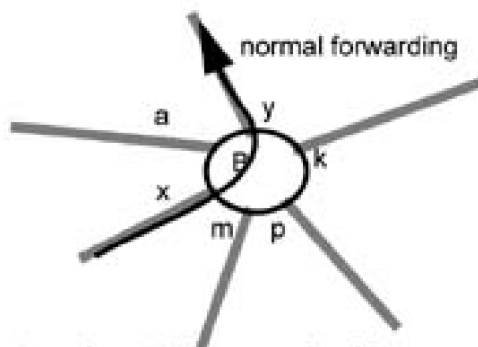
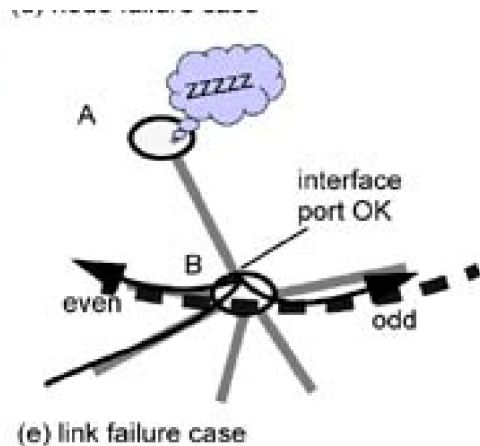
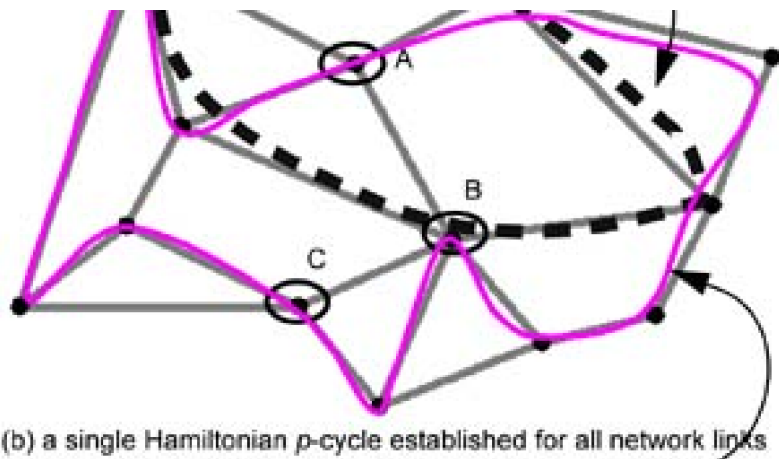
Neighbor-Node Failure Detection

Much faster strategies for neighbor-node failure detection are conceivable, however; one is to opportunistically insert "idle-fill (hello)" packets in any outgoing port at any time the queue for the link is not in use with traffic and combine this with detection of a sudden drop in packet arrival rates. Insertion of idle packets should not be a built-in function in the outgoing port interfaces, but by definition it should be the idle task of the main CPU or forwarding engine to generate such idle-fill packets. The fast detection scheme would then be to note either the sudden cessation of *idle-fill (hello)* packets during times of non-peak utilization, or, when link utilization is high (starving out opportunistic hellos), to note any sudden total cessation of real packet arrivals. In other words, a router's job is to forward real packets and if not at full utilization doing this, to forward dummy packets as continual evidence of its availability to its neighbors. Total packet load (real plus dummy) on links is, thus, only allowed to fall to some minimum composite level of packet fill at all times. At some threshold below that an adjacent node can determine its neighbor to have failed. Using this method, far less than 40 seconds would be required to reliably detect a failed router by a neighbor node regardless of the real link utilization level. Other possibilities arise when one vendor has design control of both router or LSR and underlying WDM or SONET transport services. For instance, a hardware watchdog circuit, implemented independently of the router CPU or forwarding engine, can watch the program counter or the outgoing forwarding stream or router table activity, and so on. When criteria arise based on such hardware-level observations of the node function that indicate node failure, neighbors can be advised of this in microseconds via overhead signaling in the attached OC-n or GbE links.

Using some such fast failure detection scheme, each adjacent surviving router can tell the difference between a link or neighbor-node failure. As above it marks the port to the failed router as dead, but if it deduces node (not link) failure it changes the routing entry to direct subsequent packet arrivals onto a label for an MPLS path that constitutes the node encircling p -cycle. Thus each router table in a conventional IP router, or label swapping table in an LSR, has in effect three basic routing options preprepared for each outgoing port. We describe the LSR case, but the conventionally routed case follows easily. The label-stacking feature of MPLS easily supports the "encapsulation" function mentioned above when redirecting packets onto a virtual p -cycle. The overall logic is as follows, where the outgoing port names refer to the transmitting link interfaces at node B in the example of [Figure 10-50](#):

Figure 10-50. Coordinated use of both link-protecting and node-encircling p -cycles.





Link and node protected p -cycle forwarding

Repeat

Receive_packet (inport_x, inlabel);

Look_up(outport_y, next_label);

Case status of (outport_y):

normal:

{queue packet for outport_y on label next_label}

link_bad:

{push_label (next_label <-- p-cycle label 1 for outport_y);

if label even queue packet for outport_p

else queue packet for outport_m

neighbor_node_bad:

{if dest(packet)= neighbor, then drop packet, else

{push_label (next_label <-- p-cycle label 2 for outport_y);

if label even queue packet for outport_k

else queue packet for outport_a

Forever

The initial lookup produces the normal forwarding information of outgoing port and next label. If the outgoing port is in a link failure state, the normal forwarding label is "pushed" and the label of a preplanned p -cycle is pushed on the MPLS packet. The pushed label and new outport k correspond locally to the virtual p -cycle of the network that has a link-protecting relationship to the particular outport at that node. If the normal outgoing port is in a neighbor-failed state then the packet is dropped if destined for that node, otherwise it is similarly encapsulated and redirected to a new outgoing port. This alternate port need not be the same as for the link failure case as it corresponds to activating a node-encircling p -cycle for the neighbor node. The total flow deflected onto the respective p -cycle for each failure is split over the two directions on each p -cycle by adding an additional test such as if the next_label is even or odd. In all cases nodes that receive a

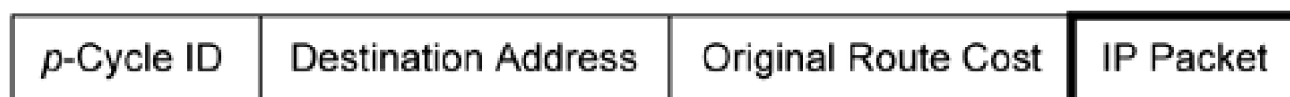
packet on a p -cycle label are either programmed to inspect the original IP destination address to see if they have a surviving normal routing table entry for that address with lower ongoing cost than the entry point cost or, continue forwarding the packet on its p -cycle label.

It may seem complex to have both link and node-encircling p -cycles but there are always only N node encircling p -cycles to establish and we have seen that if the network is Hamiltonian, as few as one link-protecting p -cycle can strictly suffice to protect against any single link failure. In practice, five or even ten additional link-protecting p -cycles is still not a large number of logical structures to establish.

Generalized Return to Normal Routing

For simplicity above, in discussing both link and node p -cycles, we have so far assumed a return to normal routing when an encapsulated packet reaches the other node on the p -cycle through which it would normally have been routed prior to the failure. It is, however, possible and generally advantageous to return a p -cycle deflected packet to a normal routing or LSP at any node on the p -cycle that has a lower continuing route cost to the destination than the route cost at the point of encapsulation. To support operation of node encircling p -cycles, and generalized return to normal routing for link protecting p -cycles, we define a p -cycle packet to contain a p -Cycle ID field, an Original Route Cost field, a Destination Address field, and, as payload, the original IP packet (Figure 10-51).

Figure 10-51. Encapsulated p -cycle packet format.



The p -Cycle ID field contains a unique identifier for the p -cycle to which the packet belongs. The Destination Address field contains the IP address of the packet's destination, and the Original Route Cost (ORC) contains the cost of the route the packet would have used prior to failure, as indicated by the local routing table entry for its destination address. The route cost is necessary to determine when it is "safe" (i.e., assured to be loop-free) to remove the packet from the p -cycle, and continue routing it normally to its destination. Under an MPLS-based implementation the p -Cycle ID field can be simply the local label at a LER or LSR node that defines the preestablished LSP that embodies the corresponding p -cycle.

As the encapsulating packet travels along the p -cycle, each router tests the packet to determine if it should remove it from the p -cycle, decapsulate it, and continue it along the remainder of its normal route. If the test fails, the router forwards the packet along the p -cycle (this is the normal routing entry for packets arriving on the p -cycle address); if the test is met, the original packet is extracted from the p -cycle packet and forwarded normally from that node using the local routing entry for the original IP address of the decapsulated packet.

The test that each router applies to any packet arriving with a p -cycle IP address is as follows: the router enters its local routing table using the Destination Address field of the p -cycle packet. If there is no router entry or a preestablished label for an LSP to that IP address, it does nothing and relays the packet on along the p -cycle. If there is a route entry that points to a currently intact port, the router then considers the route cost in the local routing table, against the ORC in the p -cycle packet. If the local route cost entry is less than the ORC it decapsulates the original IP packet and forwards it accordingly. If the local routing cost is greater than or equal to the ORC, or the port indicated by the entry is non-functional, the router continues relaying the encapsulated packet along the p -cycle.

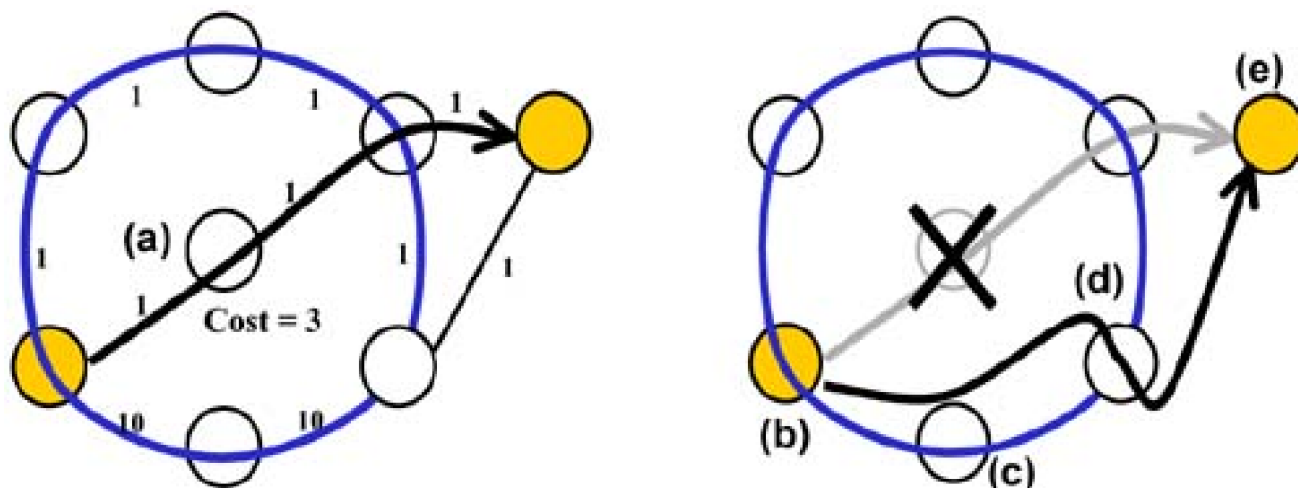
These rules ensure that the packet is only returned to normal forwarding on an existing routing that is closer to its destination than its entry point into the p -cycle. Without this mechanism to detect the first appropriate decapsulation node, a packet could get in a loop where it would continuously be introduced into a p -cycle at one point, be removed from the p -cycle at another point, and then be routed normally back to the first point where it could reenter the p -cycle. In other words, we ensure that the IP packet is reintroduced to the network only at a point closer to its destination than where it entered the p -cycle.^[12] After this, the packet continues to move away from the failure point as it is routed toward its destination, and there is no danger of it reentering the p -cycle.

^[12] Note that such other point always exists on the node-encircling p -cycle by virtue of the fact that the pre-failure route of the packet (or LSP) transited the encircled node in the first place.

Figure 10-52 gives an example of an IP packet being automatically detoured around a router failure, then restored to its original (or in general a preferred) continuing route. For the example, all links have a cost of 1, except two links which have a cost of 10 to permit illustration of the points above. In Figure 10-52(a), a packet follows its normal route to its destination. The packet's pre-failure route has a cost of 3. After being disrupted by a router failure in (b), the IP packet is encapsulated within a p -cycle packet (with original route cost of 3)

and is injected into the p -cycle. The p -cycle packet is relayed along the p -cycle until it reaches a router at (c). The router compares the original route cost in the p -cycle packet (3) to the route cost from its routing (11) and since the local cost is not less than the original cost, it allows the p -cycle packet to continue. At (d), the p -cycle packet reaches another router where the same cost comparison is performed. However, here the local cost of 1 is less than the original cost of 3; therefore, the router unencapsulates the IP packet and routes it normally. In (e), the packet arrives at its destination.

Figure 10-52. p -Cycle redirection of flows upon router failure.



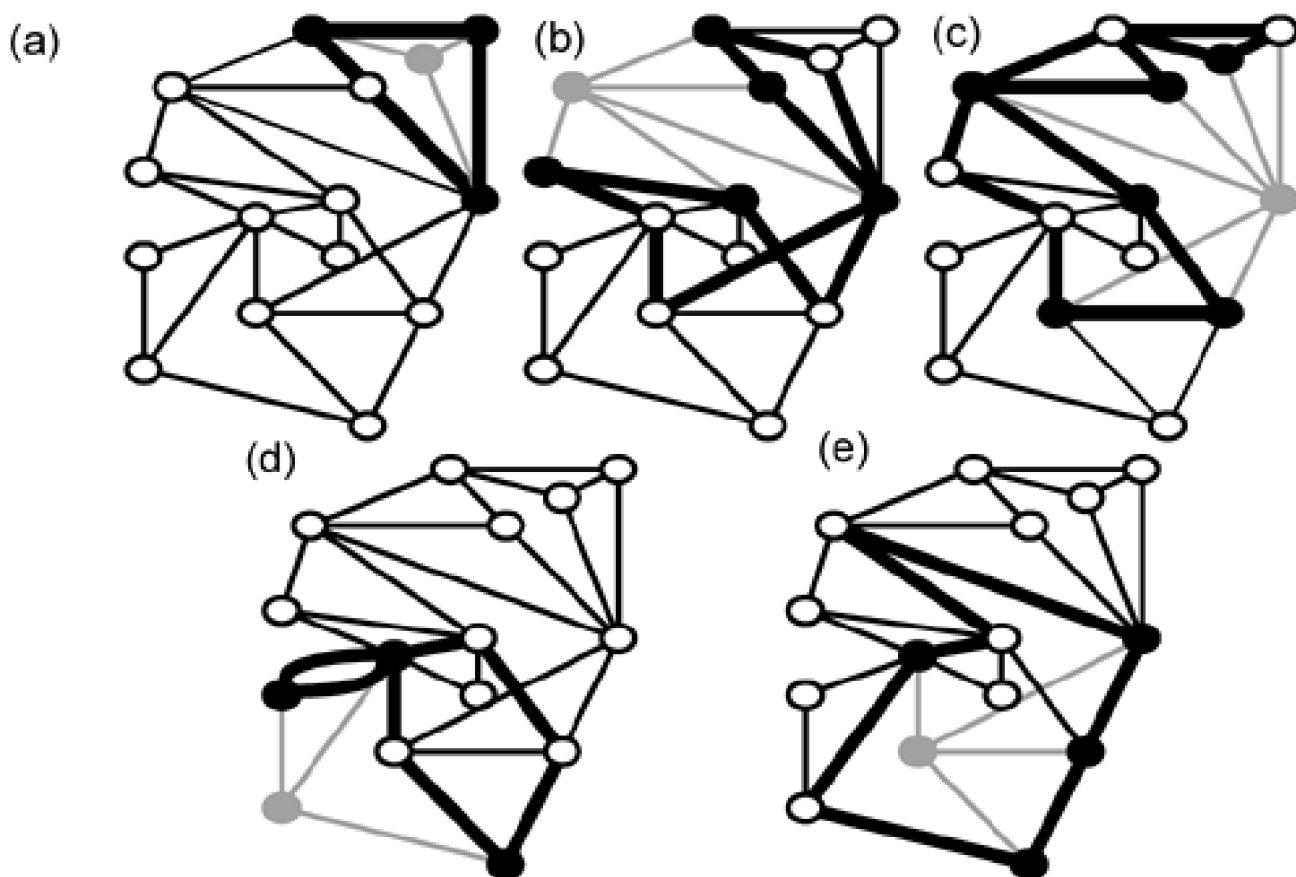
10.14.3 Generating Node Encircling p -Cycles

A simple process to generate a set of node encircling p -cycles is based on decomposition of the network into its biconnected components (see [Section 4.8](#)) following removal of the protected node and all its incident edges from the graph. A preliminary step is to generate a file of all distinct simple cycles of the graph (see [Section 4.10](#)). Let us say node j is the protected node with neighbors $N(j) = \{x, y, z \text{ and } t\}$. If the remaining graph has a single biconnected component, then a simple encircling p -cycle exists and is found by filtering the file of graph cycles to find the least-hop cycle that includes x, y, z and t but excludes j . It may include other nodes as well, but is chosen to be least-hop. Thus the encircling cycle $Enc(j)$ requires $N(j) \subseteq Enc(j) \subseteq \{N - j\}$ where N is the set of all nodes.

If the original graph is biconnected, no node removal results in a disconnected graph, but it may result in more than one remaining biconnected component. In this case the biconnected components algorithm identifies such components and the associated bridge (or articulation) nodes that join the component subgraphs. Any such bridge node, k , will be a neighbor of j (otherwise a bridge node would exist in the original graph, contradicting our assertion that it is biconnected). If so, two biconnected components N_1 and N_2 are found. Within each bicomponent a least-hop cycle $Enc_i(j)$ is similarly found that includes the nodes $N(j) \cap \{N_i - j\}$ where i indexes the biconnected components. The cycles $Enc_1(j), Enc_2(j)$ are then merged to form the protecting p -cycle: $Enc(j) = \{Enc_1(j) \cup Enc_2(j)\}$ which automatically includes bridge node k as a "Figure-8" point in the encircling p -cycle.

[Figure 10-53](#) shows the first five of 15 node-encircling p -cycles that result from this algorithm in the 15-node network above [Figure 10-53\(a\)](#) is a simple cycle but requires one inclusion of one non-neighboring node. (b) is a case where two biconnected components exist after node removal and the encircling cycle shows the resulting bridge node. (c) is a case where three biconnected components existed after node removal. (d) manifests a stub-node situation in the remaining graph. Although conceptually this is a special case that implies we establish an MPLS path down and back over the same link, this is logically still handled by the algorithm above: it simply reflects the emergence of a single-node subgraph in the bicomponent decomposition. In practice, we can include a specific check for any degree-1 node arising after initial node removal and include them as neighboring nodes in the logical encirclement through a segment through which the cycle passes twice or through a special relay interface arrangement to pass on only restored packets to surviving nodes on the stub (since there is no real need for any other redirected flows to pass down to the end of the stub and back). [Figure 10-53\(e\)](#) is another simple encircling p -cycle.

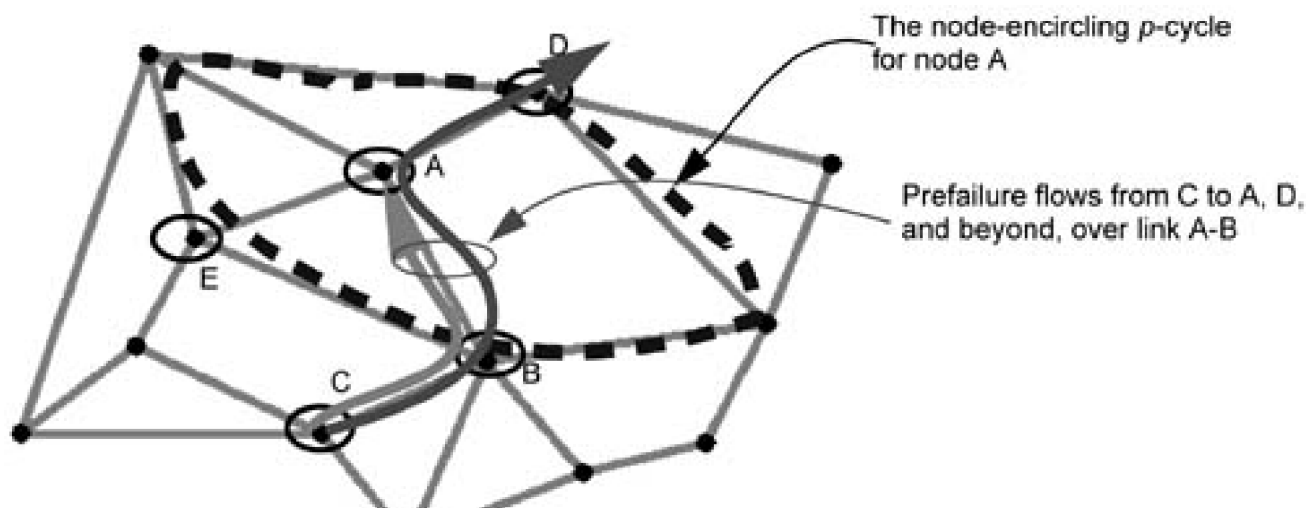
Figure 10-53. The first five node-encircling p -cycles for the 15-node test network.

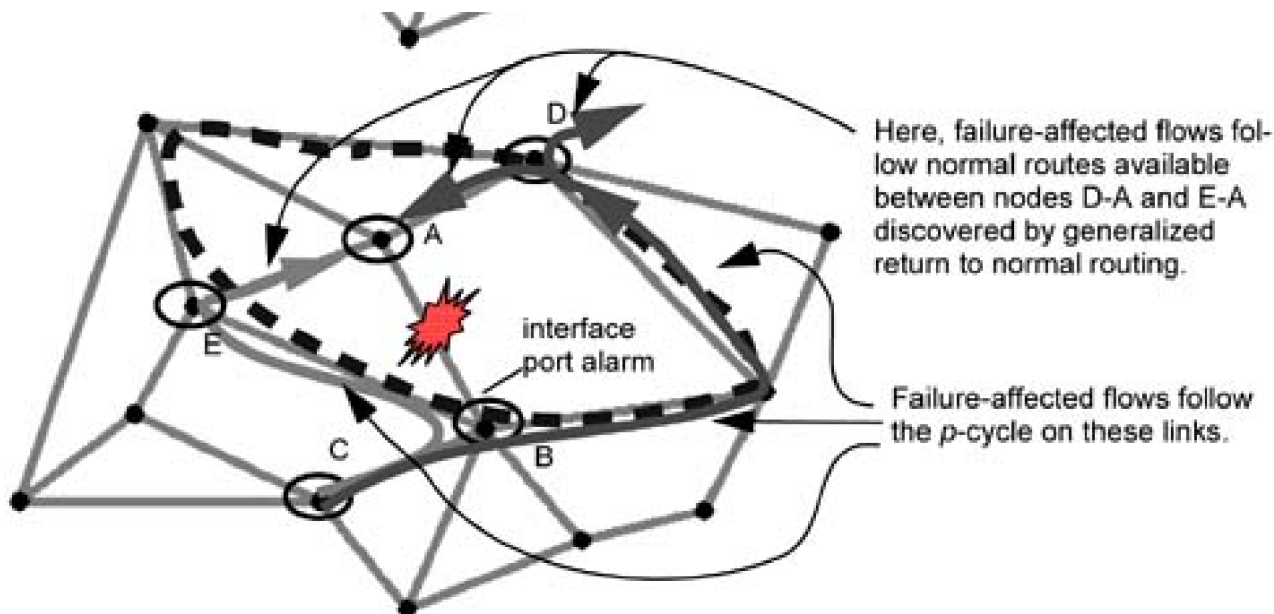


10.14.4 On the Prospect of Using Only Node-Encircling p -Cycles

A closing observation is to note that the mechanism of generalized return to normal routing from a p -cycle would make it possible to operate *solely* with node-encircling p -cycles. In response to *node* failure, the node-encircling p -cycles would be employed as described above. But when operating only with node encircling p -cycles, flows onto any failed link would also be deflected onto the same node-encircling p -cycle as if the corresponding flow had failed due to failure of the encircled node that it transits. The generalized return to normal routing option makes this feasible. How it works is shown in [Figure 10-54](#) where we revisit the node p -cycle shown previously in [Figure 10-50](#) and reconsider the failure of link A-B.

Figure 10-54. Node encircling p -cycles for both node and link failure protection.





For illustration we consider working flows from node C to nodes A, D, and beyond to the rest of the network normally routed over the A-B link, which has now failed. When using only node-encircling p -cycles, router B is not even faced with making a distinction between link or neighbor-node failure. In either case it just deflects flows that would have gone into the dead link onto the node-encircling p -cycle that is known at node B to be used for *any* restoration of the link to node A or node A itself. This simplification is offset by the somewhat greater processing needed at other nodes on the p -cycle to determine if they are in a position to decapsulate the p -cycle packets and return them to normal routings. Although by its nature a node-encircling p -cycle will not provide a p -cycle route all the way to node B from node A for the A-B link failure, we can see that under generalized return to normal routing, nodes such as E and D take it upon themselves to strip these flows off of the p -cycle and direct them on normal routes or LSPs to node A (effecting link protection) or directly onward to the network as a whole. If the failure scenario is actually node failure, the forwarding of packets to A is pointless and can be stemmed off by A and D, as well as B itself once they learn the scenario is loss of node A, not link A-B. This is, however, the best initial default action to take in the event that it is a link failure. The main operational advantages of this scheme are that link and node failures do not have to be immediately distinguished in real-time and only N virtual p -cycles are established for the network as a whole.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

Chapter 11. Ring-Mesh Hybrids and Ring-to-Mesh Evolution

A fitting way to close the book is to acknowledge that while most carriers are planning mesh-based networks, or are at least interested in mesh-based options for the future, the fact remains that in the 1990s an enormous investment of financial and intellectual capital went into ring-based transport in both metro and long haul networks. Extensive SONET and WDM ring-based networks have already been deployed—some of which are still only lightly loaded. Is this investment supposed to be abandoned in a switch to mesh-based transport? Thousands of engineers and planners have also developed intimate knowledge of ring-based planning and operations principles. Is this investment in intellectual capital to be simply lost? We would say "No" in both cases. Despite the swing of the pendulum toward mesh in popularity, rings have some intrinsic advantages in terms of simplicity and cost, especially in the metro and access applications. Therefore, in contrast to the ring *versus* mesh orientation of the 1990s, this chapter considers ways to use both technologies and how to get from rings to a mesh future in the most graceful way possible. This chapter is therefore about methods and architectures to exploit the best aspects of *both* ring and mesh-based approaches. There are four main ideas that we develop:

- Ring-access mesh-core hybrids (including access chains to replace access rings).
- Forcer-clipping ring-mesh hybrids.
- Ring-to-mesh evolution ("mining the rings").
- Ring-mining to p -cycles as the target architecture (conversion of rings to p -cycles).

Ring and mesh techniques each have their technical and economic advantages. Rings operate in a fast and simple way and each ring operates as a self-contained system. Depending on the operator, the view has sometimes been that managing a number of self-healing rings is easier than managing a self-healing *network*. Rings have also historically often been more attractive than mesh because the nodal elements, ADMs, are relatively inexpensive compared to cross-connect systems. The "capacity-slice" aspect of ring ADMs, allowing a "pay as you grow" feature, is also an often-stated advantage of rings in comparison to mesh solutions with larger one-time get-started costs for cross-connects. On the other hand, ring exhaust situations can be expensive because a whole new ring must be added to keep growing. In contrast, mesh networks offer higher capacity efficiency and provisioning flexibility, but may involve distributed rerouting mechanisms. Especially in the core network, the choice between ring and mesh technologies has tended to be based on "pure mesh versus pure ring" comparative studies. This is partly because adoption of a single pure architecture is perceived as being less complex to operate and manage, but it is also partly because truly integrated ring-mesh operational and design methods weren't known.

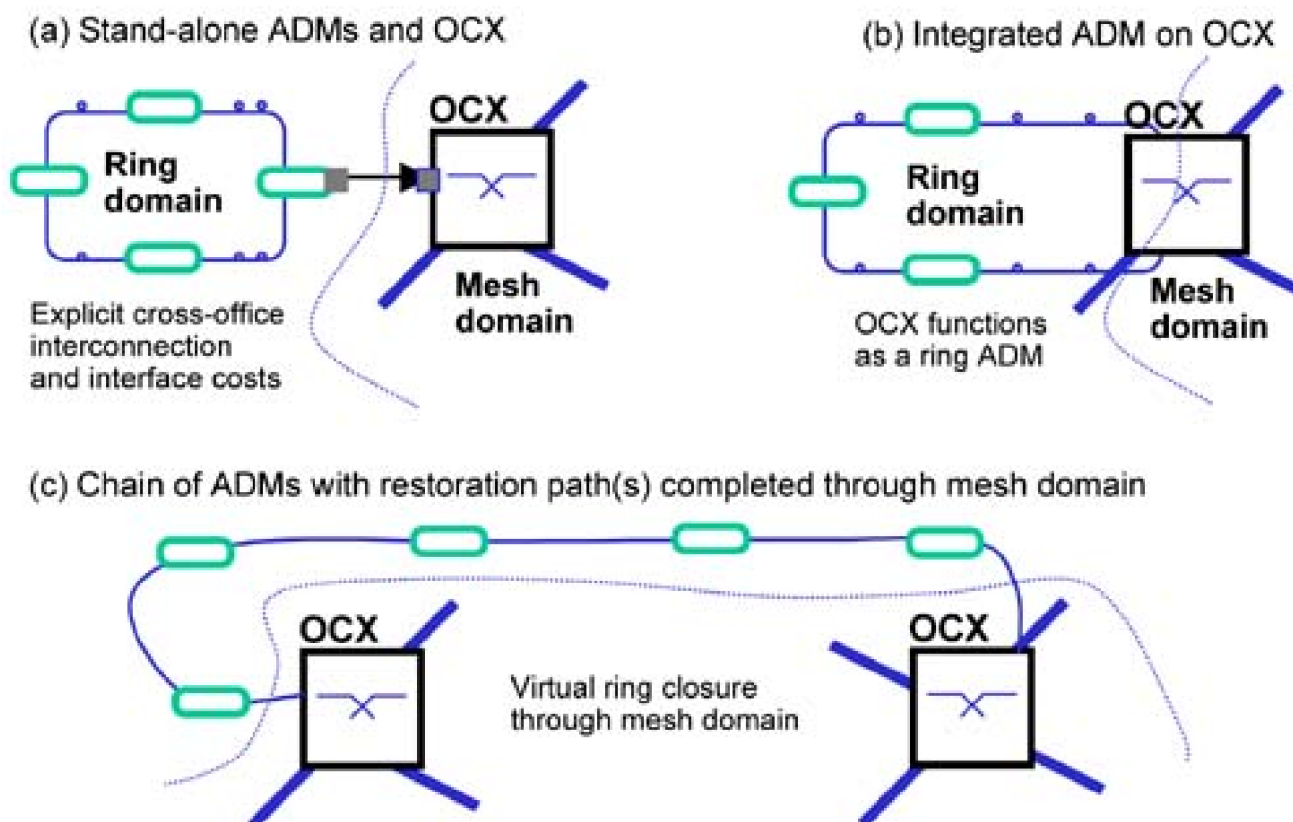
In this chapter we treat two architectural principles for ring-mesh hybrid network design and outline two strategies for *converting* some or all of the capacity of a ring-based network into a mesh network. The first architectural idea for ring-mesh hybrids is based on the fairly widely recognized principle of division of ring and mesh functionality into access and core network domains, respectively. To this we add the possibility of chain-mesh hybrids as opposed to strictly ring-mesh. The main difference is that access chains of ADMs home onto a mesh core at two locations, not requiring the use of strictly closed access rings. The second main idea about ring-mesh hybridization is more general and is based on a particular insight about how ring placements could be coordinated with the design of a mesh network so as to "clip" off strong forcers of the mesh spare capacity design. Finally, we look at a strategy referred to as "ring mining." Ring mining offers a path for ring to mesh evolution and defines a further type of ring-mesh hybrid architecture. "Ring mining" strategies are of special interest in that they provide for handling growth in demand while simultaneously converting to a mesh-based architecture.

11.1 Integrated ADM Functions on DCS/OXC: an Enabler of Hybrids

An important issue for the economic viability of the types of ring-mesh hybrids that follow is the cost for a path to transition between ring and mesh domains. To facilitate ring-mesh hybrids, it should be technically and economically easy for any demand to make a transition between transport environments en route. SONET and WDM standards assure the technical compatibility at add/drop interfaces for either ring or mesh network elements but the interface circuits and short-reach "cross-office" fiber systems to transition from mesh into a ring, or vice-versa, could be a significant economic impediment to ring-mesh hybrid networks.

In this regard, integrated ADM functionality on DCS or OADM functionality on OXC is a practical enabler of the kinds of hybrids we study in this chapter. Technically, an integrated ring node is a shelf of equipment that is housed within a DCS or OXC that allows it to interface directly at the optical line level of the ring and to function as a ring ADM node. Such integration of the ring and mesh nodal elements eliminates the need for explicit physical interfaces for ring-to-mesh signal transitions and it puts the administrative aspects of establishing the transition under one control complex. [Figure 11-1\(a\)](#) shows the traditional interfacing of standalone ADM and cross-connects and the cost elements that arise for a ring-to-mesh demand transition or for a DCS-managed ring-to-ring transition. Explicit add/drop circuit packs, short-reach cross-office optics, and interface ports are employed on both ADM and cross-connect. At OC-12 or higher the cross-office connection typically also has to be 1+1 APS protected making the overall transition expensive.

Figure 11-1. Integrated ADM functionality on the DCS: an enabler of hybrid networks.



[Figure 11-1\(b\)](#) shows how these costs are eliminated when the cross-connect is given the functionality to serve as a ring node itself. Line-level ring switching functions are performed by the integrated ADM shelf but any add/drop functions to/from the ring to the mesh (or to another ring) are effected through the cross-connect core. All costs for explicit add/drop and cross-office optics are eliminated. Note, however, that the ring is still logically complete and that protection capacity and operations for survivability remain ring-based. Ironically, the motivation for this feature has not been to facilitate hybrids. Rather, it is so-far seen as a cost-reduction measure for ring-based networking where inter-ring transitions are cross-connect managed. The signal routing between rings is DCS or OXC-managed, but

protection remains entirely based on rings. The ring-based protection architecture is not really changed from that in (a), only the cost of its realization is lowered.

[Figure 11-1\(c\)](#) shows another arrangement that is logically facilitated by physical hosting of the ring node functionality on the cross-connect. This is mesh-chain networking (as opposed to ring-mesh), which is one of the hybrids we will be considering. The OCXs terminate the optical line systems and emulate the ring signaling protocol on their ring interfaces but are fully mesh-capable looking into the mesh domain of which they are also nodes. The ADMs in the chain do not know that they are not part of a complete ring. The access ring is thus partly virtual: the part of the ring that is not really needed for the access function is omitted, but the ring protection switching—which is still required—is emulated through the mesh domain for restoration purposes.

Thus, the advent of cross-connects with integrated ADM functionality makes it possible to consider a range of new hybrid networking strategies that can use mesh or ring principles to take fullest advantage of their respective characteristics. The technical infrastructure for such ring-mesh hybrids is in place but the full economic opportunity is awaiting realization.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

11.2 Hybrids Based on the Ring-Access Mesh-Core Principle

The most obvious and fairly widespread form of hybrid arises from the use of *rings in the access and mesh in the core*. The characteristics that distinguish "access" and "core" regions are qualitatively recognized by planners as described in [Flan90]. In the access network, distances are generally not great, so unrepeated, often even unamplified, fiber sections are possible between customer access nodes. In such circumstances a ring gives single failure survivability for a daisy-chain of customer access nodes with relatively little expense in terms of transmission equipment. The implementation requires only ADMs and dark fiber. Indeed some widely used metropolitan network planning systems have a default cost of zero for inter-nodal fiber transmission, which reflects the situation of short-reach and ducts full of already-placed dark fiber. In access applications the ADMs are typically also UPSR type and are even simpler and cheaper than BLSR ADMs. The line capacity of the ring transmission system is typically much greater than any one customer's demand so the ring acts as a structure for sharing the "economy of scale" in bandwidth over all the accessing customers. In an environment where there is essentially no cost associated with transmission, the entire planning problem is about reducing *node* costs. These are all reasons why ring systems, even the capacity-inefficient UPSR logical structure,^[1] can nonetheless be economically attractive for access network applications.

^[1] Under a pure single-hub pattern of access demand, a UPSR is as efficient as a 2-fiber BLSR in the same role. This is the reason, combined with low cost and simplicity, that UPSRs are so common in access applications.

In contrast, in a long haul network distances may require several optical amplifiers, regenerators, splice casings and other physical housings, and many miles of right-of-way and conduit investment on each span. In this case distance and capacity-related transmission costs are much more important. A significant fundamental investment is made by long haul carriers in right-of-way, ducts, fiber, and equipment buildings. This is typically based on a business plan that assumes that such basic infrastructure will last for a certain time period and provide a certain ultimate growth capacity. It is important, therefore, that the real prorated cost of consuming this asset is reflected in network planning. Typical estimates to put more fiber in the ground are in the range of \$100,000 per mile. In these circumstances, mesh networking principles offer potentially large cost reductions by keeping working paths as short as possible and by network-wide sharing of spare capacity, both of which reduce the total investment in transmission equipment, fiber, conduit and route miles. Despite the huge capacity advances provided on a single fiber by DWDM, the popular saying of futurists and prognosticators that "bandwidth is free" has certainly never been true for a long haul network operator. On a pennies-per-bit-per-second basis, the cost of a fully loaded 10 GbE lightpath may indeed seem numerically small, but the cost of turning up a new lightpath is still a great real expense. Moreover, even a medium-sized long haul operator can face *annual* capital expenditure requirements in the range of \$300 to \$500 million just for the *incremental* acquisition of new transport equipment. There is therefore no actual case to be made against trying to be efficient in the use of such capacity. This efficiency, plus the business and operational value of greater flexibility with mesh networking, is why regional or long haul backbone networks especially, are well served by the mesh approach.

Thus, a natural notion for a first kind of hybrid is "rings in the metro or access, mesh in the core." However, within this class of hybrids we want to consider two additional principles:

- a. Hybrids in which the access rings are replaced by ADM chains, which are not fully closed access rings. This is the variant we call a "mesh-chain" hybrid.
- b. Hybrids in which the concept of "span elimination" is employed to enhance the economics of the access portion.

11.2.1 Core and Access Network Partitioning

An aspect that is common to either ring-mesh or chain-mesh hybrids based on the ring-access mesh-core principle is partitioning to define the access and core regions. In this section we provide some heuristic principles for defining this partitioning to suit the resultant deployment of a set of access rings and a mesh core. The task is to partition the original network into core and access portions by assigning nodes and spans to form core and access subgraphs. In doing so we require the core to retain biconnectivity while separate disconnected access regions are permitted. Each access subnetwork must itself remain biconnected to support the deployment of

conventional closed rings and each access region should preferably have two or more nodes in common with the core.

One principle for defining the regions on which to implement ring and mesh techniques is that the degree of core nodes is generally higher than that of access nodes. In addition, from ring network design principles, we know it is beneficial for nodes to have even degree because this permits a ring cover to avoid span overlaps. A third principle is that any degree 2 site is a natural candidate for an ADM rather than a DCS. An ADM is functionally equivalent to a DCS when in a degree 2 site. The partitioning procedure assumes that all available facility routes are admitted as spans at the start of the problem (later some may be eliminated) and that all demands are initially shortest-path routed over the graph, except where later noted.

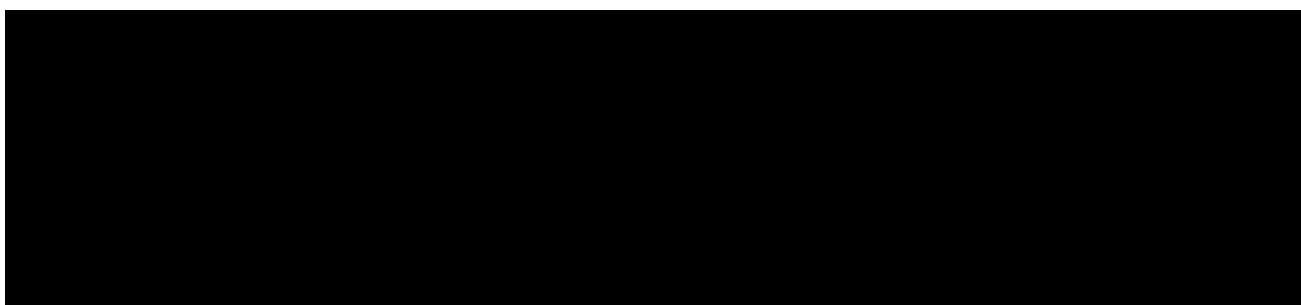
The procedure is essentially a search for a "threshold" of nodal degree, d_{th} , which leads to lowest cost when used as the criterion for assigning nodes to access or core regions. The search provides a systematic exploration of a range of access/core trade-offs as follows:

- a. Set $d_{th} = 3$.
- b. Assign all nodes with nodal degree $d < d_{th}$ to the access region. The complement set forms the core for this iteration.
- c. Inspect the core network subgraph to ensure biconnectivity. "Repair" the core subgraph if needed by selecting a node for readmission into the core. Nodes with degree $(d_{th}-1)$ or the highest odd degree nodes are given preference in this.
- d. Admit a "copy" of any core network span that is needed to assure biconnectivity in an access subnetwork, back into the access network.
- e. Apply a ring network design tool on each connected segment of the access network.
- f. Produce a mesh capacity design (e.g., SCA, JCA, MJCA, etc.) on the core network.
- g. With reference to the original demand matrix, trace the routes of demands that make a transition from access to core components and add costs (if any) for ADM add/drop interfaces and DCS tributary ports for cross-office ring-to-mesh transitions (or similarly any ring-to-ring transitions).
- h. Record total design cost.
- i. Increase d_{th} and repeat, starting at b. (Stop at $d_{th} = d_{max}-1$ where d_{max} is the highest nodal degree in the original network.)

In practice, we expect minimum total cost to emerge at $d_{th} \leq 4$, so typically there are only two or three iterations in the main loop. This means that if a "repair step" is encountered in defining the core at one iteration, it may be feasible to test all repair options for design improvement.

The following figures illustrate various stages of the process so far described. [Figure 11-2](#) shows a succession of the partitioning results on an example topology. [Figure 11-3](#) is an example to illustrate how the process can potentially result in a non-biconnected graph for the core, and how a node is selected for return to the core in this case. [Figure 11-3](#) also illustrates the spans that are implicitly included in both core and access subnetworks so that access rings can be formed on closed cycles. (In mesh-chain networking such spans are a key source of cost-savings). The annotations nn/xx give the number of working channels on the span arising from shortest path routing and the length of the span in km, respectively. A helpful exercise in conjunction with the partitioning search is to plot a histogram of nodal degree of all nodes in the network. This gives some insight into the distribution of nodal degree within the network and may show breakpoints where nodes cluster in a somewhat natural separation of prospective core and access regions. [Figure 11-4](#) is such an example.

Figure 11-2. Searching through nodal degree to partition the access & core regions.



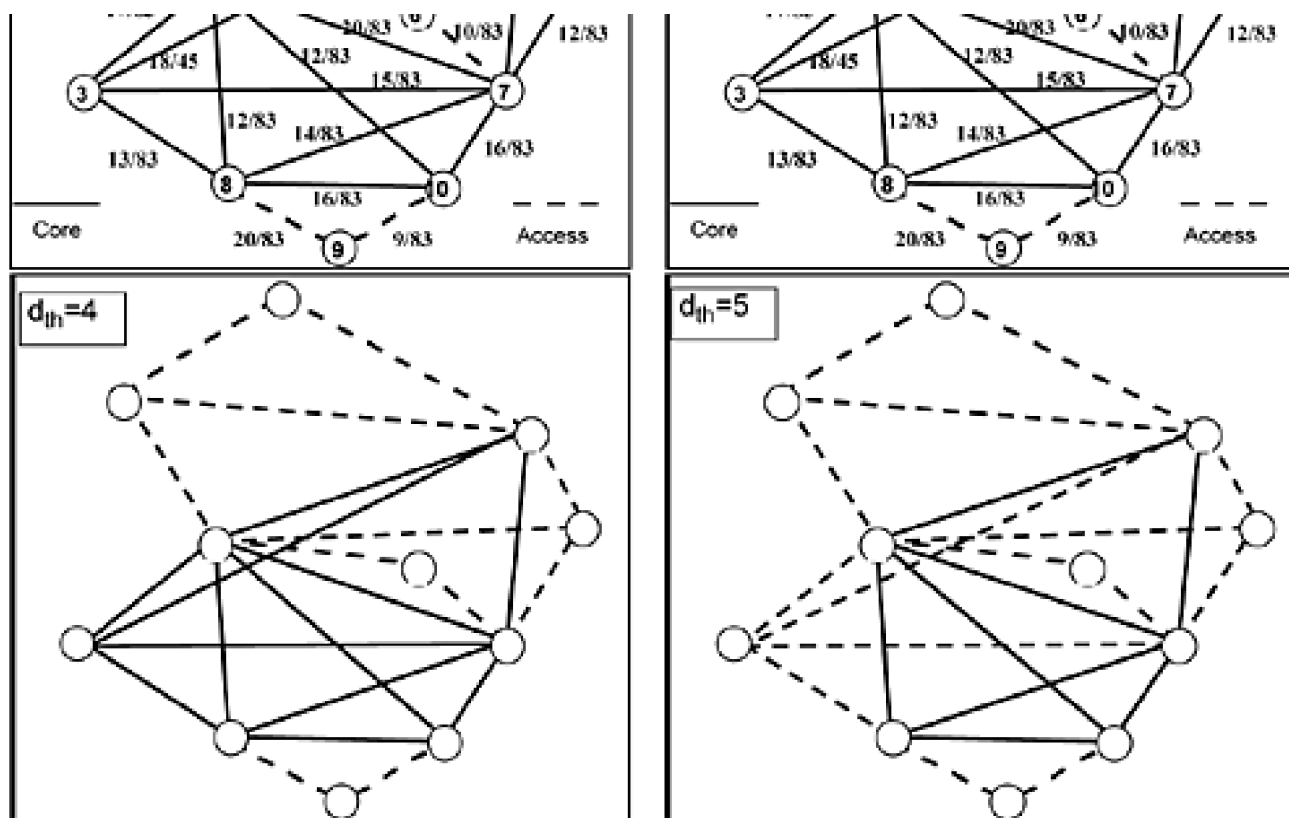


Figure 11-3. Example where $d_{th}=3$ partitioning would leave the core one-connected. Node "A" is returned to the "core" node set.

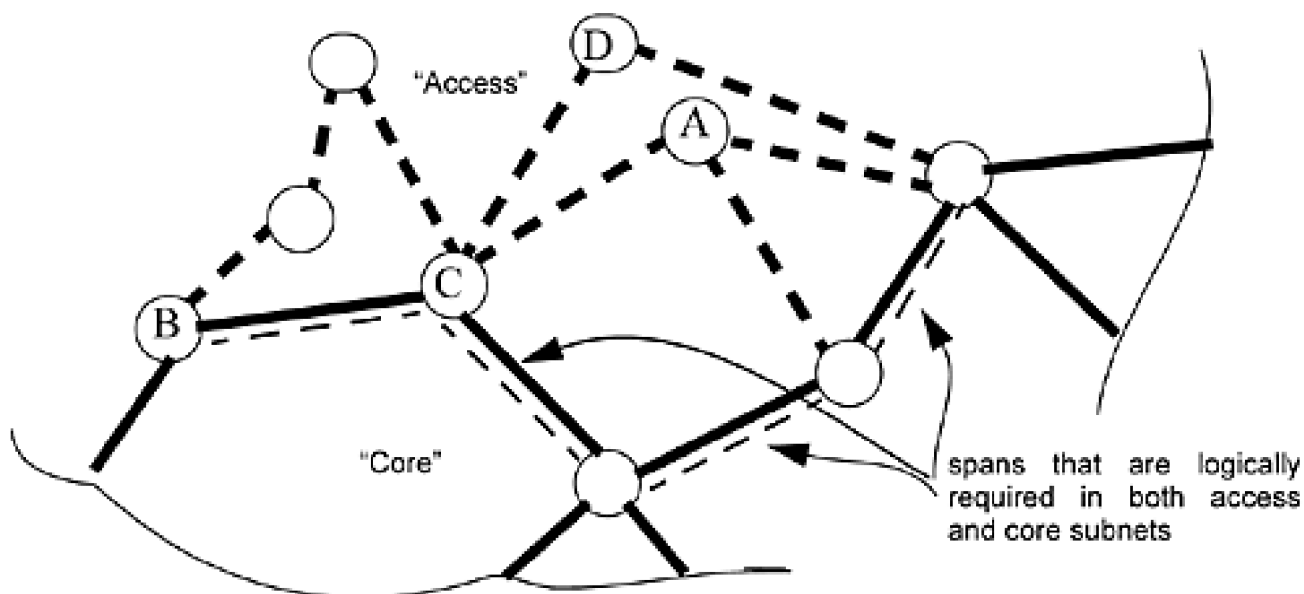
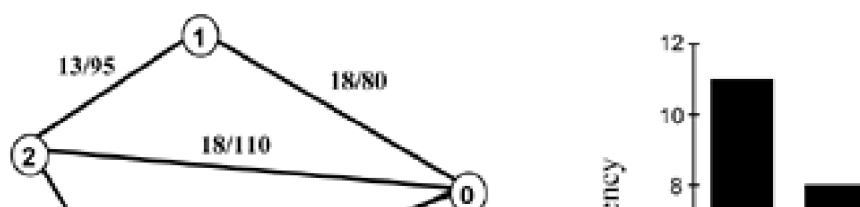
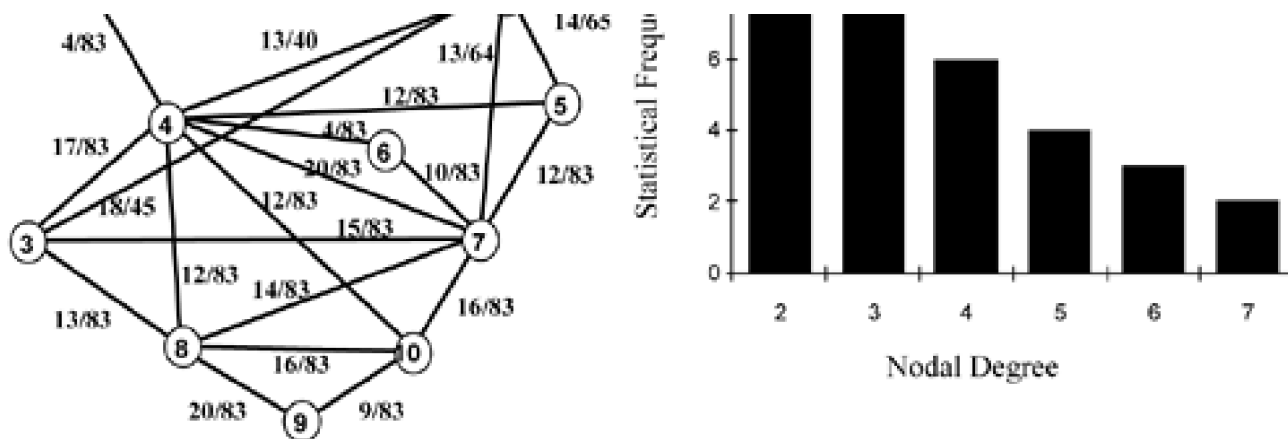


Figure 11-4. Example of nodal degree analysis of the overall network topology.

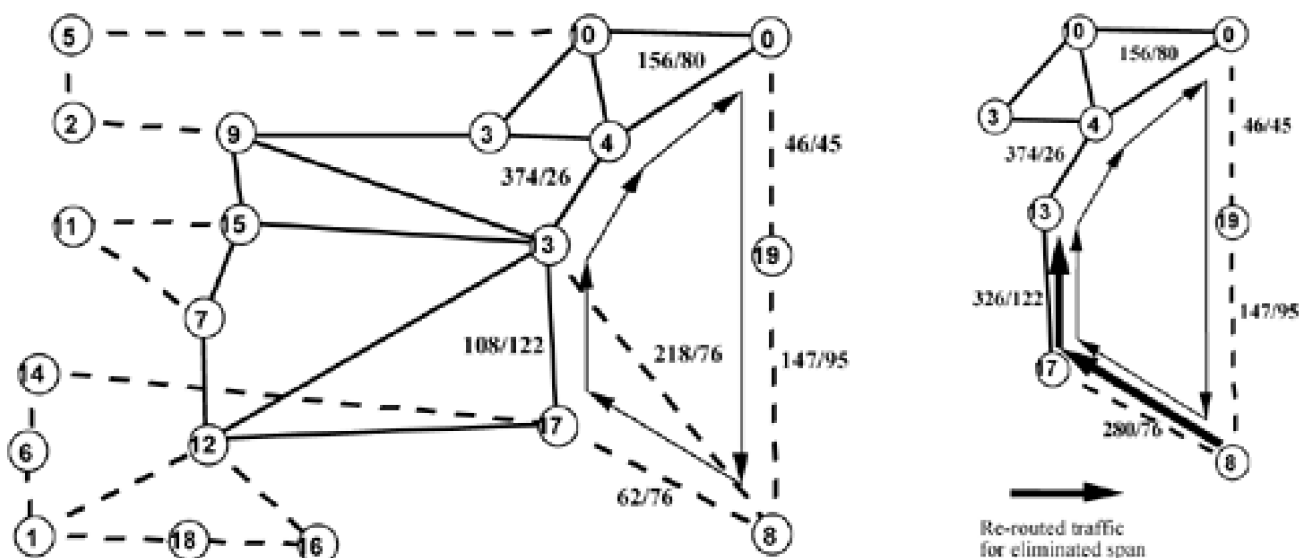




11.2.2 Access Ring Formation and Span Elimination

To evaluate each access/core partitioning that is generated we must arrange a cover of the remaining access nodes and spans with rings that intersect the mesh region at two or more core nodes so no that single-node failure can isolate an access ring. This is illustrated by the example in [Figure 11-5](#) for the core/access partition shown in the left-hand panel. An access ring on cycle {0,19,8,17,13,4} that intersects the core at four nodes (i.e., nodes 17, 13, 4 & 0) could be formed but this would leave access span (8-13) unprotected. In fact the odd degree of node 8 in this access network partitioning means fundamentally that at least two rings would be required to cover all of its incident spans. Spans (8-13) and (8-17) are therefore candidates for "span elimination."

Figure 11-5. Illustration of span elimination in access ring formation.



Span elimination is a principle of efficient ring-based network design in which the working flow over a span may be diverted over a longer route when doing so permits fewer ring systems in total. The "eliminated" span is still physically present (and may have other uses), but it is effectively excluded from the graph on which the current stage of ring-cover planning is conducted. Working flow on such spans is detoured to take a route over spans that are included in the ring cover. This means assigning a longer route to some working flows, but this inefficiency is not of much concern in the ring-oriented access design for the reasons explained above. Using the fewest ring systems in total is often the key to a minimum-cost design because nodal costs are reduced—and this is what is of primary importance to the access design. Further discussion on this aspect of ring-based network design is available in [\[LeGr99\]](#).

In [Figure 11-5](#), span (8-13) are considered candidates for logical elimination from the access network by rerouting its 218 units of working capacity along (8-17-13). This increases working capacity totals on spans (8,17) and (13-17) to 62+218 and 108+218, respectively, and a single access ring can be placed requiring at least 316 units of working capacity. Alternatively, we could also have rerouted via the

four-span route (8-19-0-4-13) but this would have increased working channel counts on 4 spans by 218, which increases the total working count for the network more than the two-span route selected. It is a general principle of BLSR ring network design to route the demands in the direction that yields the lowest w_{max} on all spans of the ring. This applies here for the incremental demand on the ring arising from the span elimination. (For a UPSR-type ring, the capacity implications are identical whichever direction the primary working path takes.) In general, the whole body of techniques and knowledge for multi-ring network design (which are beyond the present scope) can be brought to bear at this stage on the access subnetwork design problem.

As a result of the logical elimination of span (8-13), node 8 as well as node 9 can be served by a single ADM of suitable line capacity. If we had retained span (8-13) then node 8 would be a degree 3 node requiring a DCS or at least two ADMs at its site.

Of course any span elimination increases the total working capacity mileage required in the access network. But this is of little real cost as long as the fiber distances remain below optical amplifier spacings and when the capacity total produced on spans does not exceed the line capacity of the available rings. The counteracting savings in the example is at least one ADM in node 8 of the example. Thus, span elimination makes a trade-off between fiber-km cost and nodal equipment costs. In practice, the trade-offs can only be judged with detailed costing and ring modular capacity information. An alternative that produces lower total capacity requirements in the access rings is to keep span (8-13) in the access network and form two rings that necessarily overlap on at least one span. This could incur a lower capital cost than a single ring with span elimination if the line capacity requirement of the latter solution exceeded an important modular capacity threshold, such as OC-192 for example, requiring a second stacked ring to serve the total capacity. The corresponding alternative in the situation above would be to form two access rings, namely (8-17-13) and (8-13-4-0-19) rather than the single ring (8-19-0-4-13-17). This principle can be taken further to where a degree-4 access node could be used as a common node for three intersecting rings whereby three ADMs would be installed and manually cross-patched at the node or a cross-connect provided.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

11.3 Mesh-Chain Hybrid Networks

Upon studying the ring-mesh hybrids above, one notices the rings that are formed may still be quite inefficient. If, for example, in the (8-19-0-4-13-17) access ring above most of the demand from nodes 8, 19, and 0 is to distant destinations, then an efficient route to the destination can start immediately from nodes 0 or 17 in the mesh domain. With respect to such working flow there would be no actual need to have ring-spans (17-13-4-0) present at all. Similarly on spans such as (17-13), (13-4) where the access ring and a span of the mesh domain overlap, it would always be more efficient to place demands exchanged or routed through these nodes in the mesh domain, not in the parallel access ring. In fact the only reason that ring spans (17-13-4-0) would be present in the ring is to complete it as a closed cyclical subnetwork as required for ring protection to operate.

With this in mind [BrGr94] proposed a variation on the access-core hybrid principle in which a distributed mesh restoration algorithm (DRA) is used in the core and with signaling conversion where chain subnetworks using ADMs interface to the core. In this architecture the core is mesh-based and similar or identical to the core in a strict ring-mesh hybrid, but the access network consists of incomplete portions of rings; really just chains of ADMs, each end of which is anchored on a different node of the mesh core. The ADMs in the chains do not know that they are not still in closed rings and continue to execute their own BLSR loopback protection protocol. At mesh nodes where the ends of such a chain of ADMs terminates, the ring protection protocol (for example SONET K1-K2 byte signaling) is supported on the ring side, but looking into the mesh domain, the more general rerouting possibilities of the mesh core are exploited. This architecture offers capacity savings arising from two effects: (1) elimination of inefficient segments of access rings where such segments are present only for ring closure, and (2) sharing of access and core protection capacity.

The first efficiency arises from not investing the extra ring capacity required to simply *close* each access ring. The less obvious efficiency is that the protection capacity of the access chains becomes available for use by the mesh core and vice-versa. With strictly closed access rings, the ring protection capacity only serves the access ring itself and cannot be accessed to assist in restoration of faults arising in the core. Another practical simplification is that mesh-chain hybrids can be efficiently designed with a single (mesh-oriented) planning tool. A capacity design process such as SCA or JCA need not make any distinction between access chains and the higher degree mesh core. It inherently makes efficient integrated use of the whole topology and all capacity placed on it. The only difference outside the scope of the routing and capacity design itself and is that we use ADMs to implement the degree-2 nodes. In fact if MJCA is used, the mesh design even takes into account the cost versus capacity characteristics of available ring systems. In contrast, the ring-mesh hybrids create two separate design problems; one for the mesh core using mesh design tools, and one for the ring access subnetwork, which is a separate instance of the more difficult problem of multi-ring network design. The key to understanding why a mesh design tool will completely serve for a mesh-chain hybrid is that, from a capacity allocation standpoint, a span-restorable mesh inherently requires the same loopback-type of spare capacity allocations on degree-2 chains. This is the same capacity as needed for the ADMs to function as if they were still part of a (BLSR) ring. [Figure 11-6](#) and [Figure 11-7](#) help illustrate these points.

Figure 11-6. (a) Ring-mesh design for example above, (b) mesh-chain design.

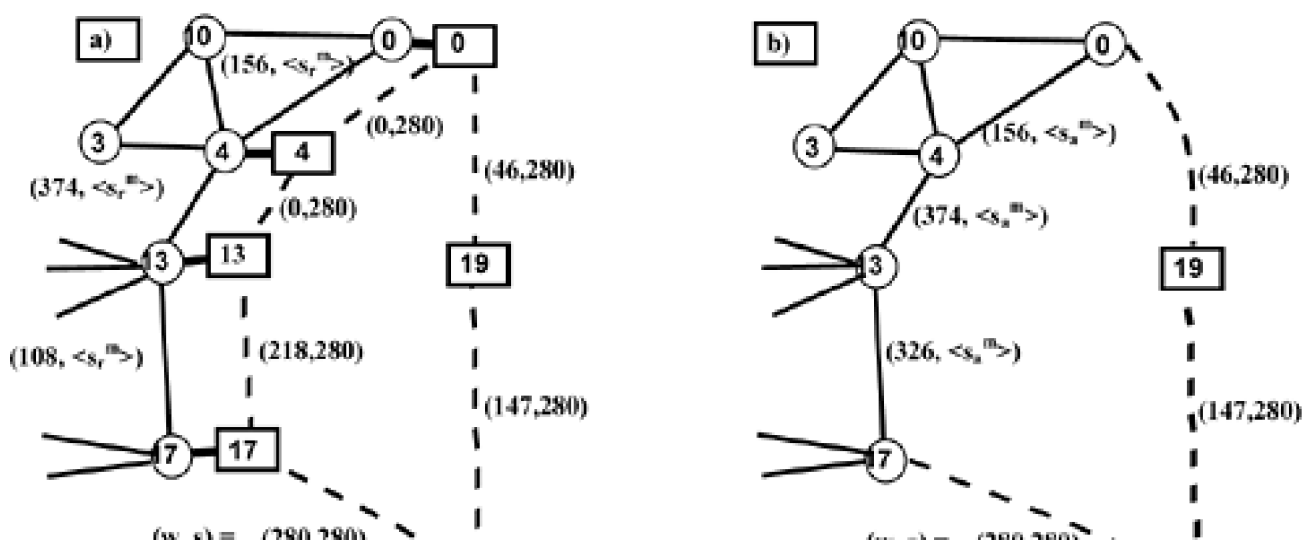
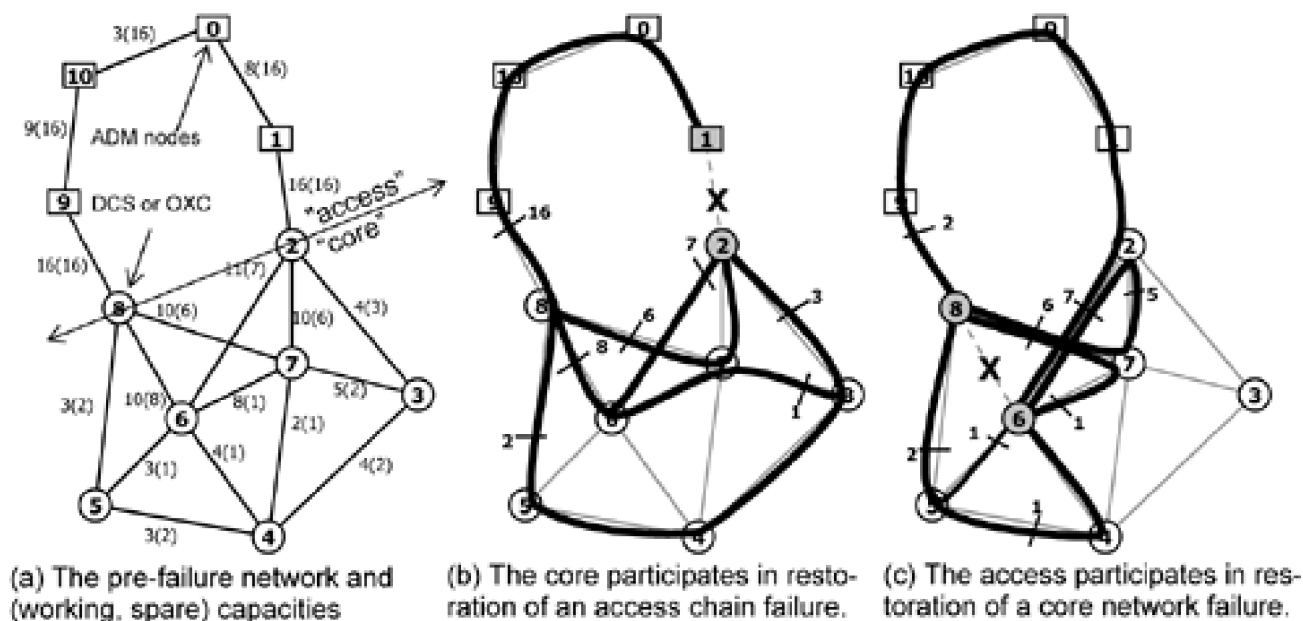


Figure 11-7. Illustrating cooperative interaction of OXC-based core with ADM-based access.



In [Figure 11-6\(a\)](#) we have a detailed view in on the (0-19-8-17-13-4) access ring discussed in [Figure 11-5](#). The number pair by each span indicates required working and spare capacity. Assuming the access ring to be of a BLSR type, it requires 280 units of spare capacity on all six of its spans. In contrast [Figure 11-6\(b\)](#) portrays a three-span ADM chain that performs exactly the same demand-serving and protection functions as the above ring. On the three spans that are common to the ring and the chain, working and spare capacity values are the same. But on the four spans where the ring overlies spans of the mesh core, we combine the working capacities and solve for a single allocation of spare capacity to each such span from a global mesh-oriented standpoint. These values are not determined numerically in [Figure 11-6](#) as they depend on the rest of the topology and working capacity details of the mesh core, but it suffices to say that from first principles the sum of the $\langle S_{ai}^m \rangle$ spare capacity allocations in the integrated mesh-chain hybrid will be less than the sum of

the corresponding ring protection allocations plus the separately solved mesh core subnetwork spare capacity values $\langle S_r^m \rangle$. Although the mesh-chain architecture is still a hybrid in terms of its use of ADM and cross-connect node functionalities, we see that the efficiency of the concept comes from the fact that capacity requirements for access, including access protection, become logically integrated and globally optimized as if the whole network was a span-restorable mesh.

[Figure 11-7](#) gives a corresponding example of how the investment in spare capacity for a mesh-chain hybrid is shared—to the benefit of both access or core network failures. The network of [Figure 11-7\(a\)](#) is an example of a mesh core on nodes 2, 8, 6, 7, 3, 4, 5 to which four other nodes 0, 1, 9, 10 are connected via an access chain. The set of working span capacities is given and the spare capacities shown were produced by the SLPA algorithm [\[GrBi91\]](#) (an algorithmic precursor to the MIP methods in this book, corresponding to SCA) [Figure 11-7\(b\)](#) shows an access network failure and the restoration pattern obtained by *ksp* routing as a model for the restoration process. 100% restoration is achieved (16 paths). [Figure 11-7\(c\)](#) shows a corresponding example of a core failure where its restoration paths conversely use the protection capacity on the access chain. Again 100% restoration is achieved via 10 restoration paths.

The behavior exhibited by the ADMs in these examples is exactly that of a BLSR loopback reaction. The point is that under a span restoration routing model, the reaction to failures within any chain subnetwork is inherently a BLSR-like reverse-direction loopback. It is dictated by the topology and is the same whether ADM or DCS nodes are present at the chain nodes. At the ends of the chain, the span restoration process again flows more generally over the reserve network of the core, but within the chain it naturally emulates a BLSR. Such cooperative interaction between the ADM chain and the mesh core can be implemented by signaling conversion at the mesh nodes where the chains terminate. These nodes act as ring-mesh gateways. They act both as ring line system terminators emulating and supporting the K1-K2 byte protocol specific to the ring, while looking the other way into the mesh domain are capable of the more generalized restoration protocol signaling that forms mesh-based restoration path set. The ADM protection logic is only a constrained instance of the more general mesh signaling and so it is completely feasible for the cross-connects to convert the signaling as needed and deduce the required cross-connections to effect either of the restoration path-sets in [Figure 11-7](#). It can be left to the mesh cross-connects, therefore, to close the ring *virtually* and to emulate the signaling the ADMs expect to see—thinking they are configured in a ring as usual. Thus, we really only require ADM chain segments for survivable access. We do not strictly require closed rings for

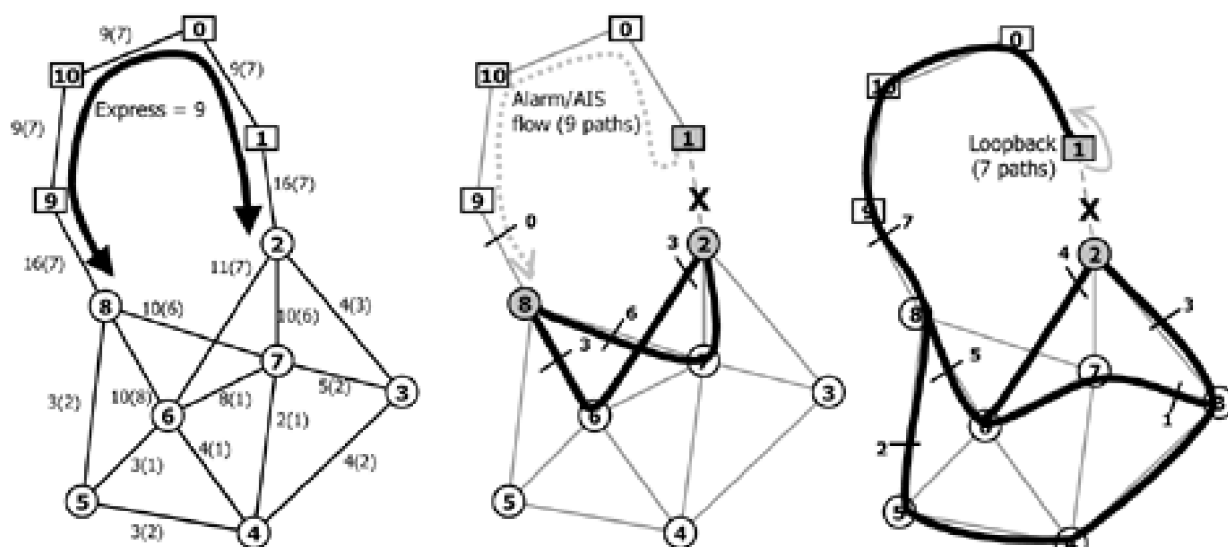
survivable ADM-based access. The mesh-chain hybrid avoids the significant duplication of protection capacity that would be required for a corresponding ring-mesh hybrid.

Adding the Meta-Mesh Concept to Mesh-Chain Hybrids

Let us digress briefly here to add a useful linkage to the meta-mesh technique of [Section 5.8.1](#). The arguments and examples of this section have so far focused on conventional ring and span-restorable mesh techniques operating together. The point is that in the failures in [Figure 11-7](#), the aim of the restoration process is to achieve complete w_j capacity replacement between the two end nodes of the failed span. Consequently the access chains in [Figure 11-7](#) have what is called "full loopback" spare capacity. That is to say that $\max\{w_j\}$ in the chain is 16—so s_j is 16 everywhere on the chain spans.

But the meta-mesh technique shows how we can improve on this if we know that some of the flow through the chain is contiguous express flow between the end nodes of the chain. That is, working demands that go entirely through the chain without add/drop at any node. In this case, the meta-mesh technique teaches that the spare capacity in the chain can be reduced to support loopback only of the $\max\{w_j\}$ -express span quantity. In the event of a chain span failure, local flow is looped back and express flow simply "fails back" all the way to the corresponding chain anchor nodes—here nodes 8 and 2. At those nodes it is treated simply as part of the total working flow to be mesh-restored between nodes 8 and 2. [Figure 11-8](#) revisits the access-chain failure case above to illustrate the added differences if meta-mesh networking is also employed.

Figure 11-8. The Meta-Mesh technique added to the mesh-chain hybrid design.



(a) An express flow of 9 reduces the access chain spare capacity from 16 to 7.

(b) The express flow of 9 is entirely restored in the mesh core.

(c) The local flow loops-back ring-like. The mesh core deploys 7 paths that complete the access "ring" virtually.

The example assumes a known express flow of 9 units through the chain. (Some of the w_j quantities are increased just to support the example.) There is no spare capacity provided on the chain itself for the express flow, so s_j values drop everywhere in the chain from 16 to 7. When the same failure in the access chain shown in [Figure 11-7\(b\)](#) occurs, the alarms arising from LoS (or AIS insertion at upstream nodes that see the LoS) are allowed simply to propagate back to be seen by the anchoring OXC nodes. An example of how those nodes might use the available mesh spare capacity to replace this working flow as shown in [Figure 11-8\(b\)](#). The non-express flow damaged at the failure span is, however, explicitly looped-back over chain spare capacity. It too is then treated as a mesh restoration flow requirement when looped-back to the mesh anchor nodes. The overall combination of techniques offers a hybrid design strategy with all of the following attractive features:

1. ADMs are used in degree-2 sites. This is the benefit arising from ring access architecture.
2. Chain spare capacity is required only for loopback of local flows—the meta-mesh benefit.

3. Spare capacity in the core is available for both core and access failures, avoiding duplicate protection capacity in access rings and mesh core. This is the mesh-chain hybrid benefit.

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

[\[Team LiB \]](#)

◀ PREVIOUS

NEXT ▶

11.4 Studies of Ring-Mesh and Mesh-Chain Hybrid Network Designs

In this section we complete the discussion of ring-mesh and mesh-chain hybrids with quantitative results comparing these schemes and corresponding pure mesh and pure ring reference designs. The considerations here do not further involve the meta-mesh aspect, however. The data presented are based on a case study [BrGr94] on the New Jersey LATA study network model [Bell93] and the working span capacities originally published with that network model. The w_i values and associated span distances (respectively) are portrayed in the upper left panel of Figure 11-4.

11.4.1 Design of the Ring-Mesh Hybrids

The comparative study employed two of the possible core/access partitionings shown in Figure 11-2—those for $d_{th} = 3$ and 4. The spare capacity allocation for the core subnetworks was performed independently of the access ring design, using the previously mentioned SLPA algorithm. In the $d_{th} = 3$ partitioning there were only three relatively small access rings and no opportunities for span eliminations. On the $d_{th} = 4$ partitioning, four access rings were formed and each of these involved a span elimination. The rings, span eliminations, and capacity rerouting details are given in Table 11-1. The two hybrid designs that result are portrayed in Figure 11-9. Notice that the $d_{th} = 4$ design makes much more extensive use of ADM equipment, requiring only five cross-connect systems, but uses more total capacity than the $d_{th} = 3$ design. This is how the hybrids provide a trade-off that can lead to lower total cost than either a pure ring or pure mesh design in circumstances where nodal equipment costs and transmission capacity costs are both significant.

Figure 11-9. (a) Ring-mesh hybrid design based on (a) $d_{th}=3$ (b) $d_{th}=4$ access/core partitioning.

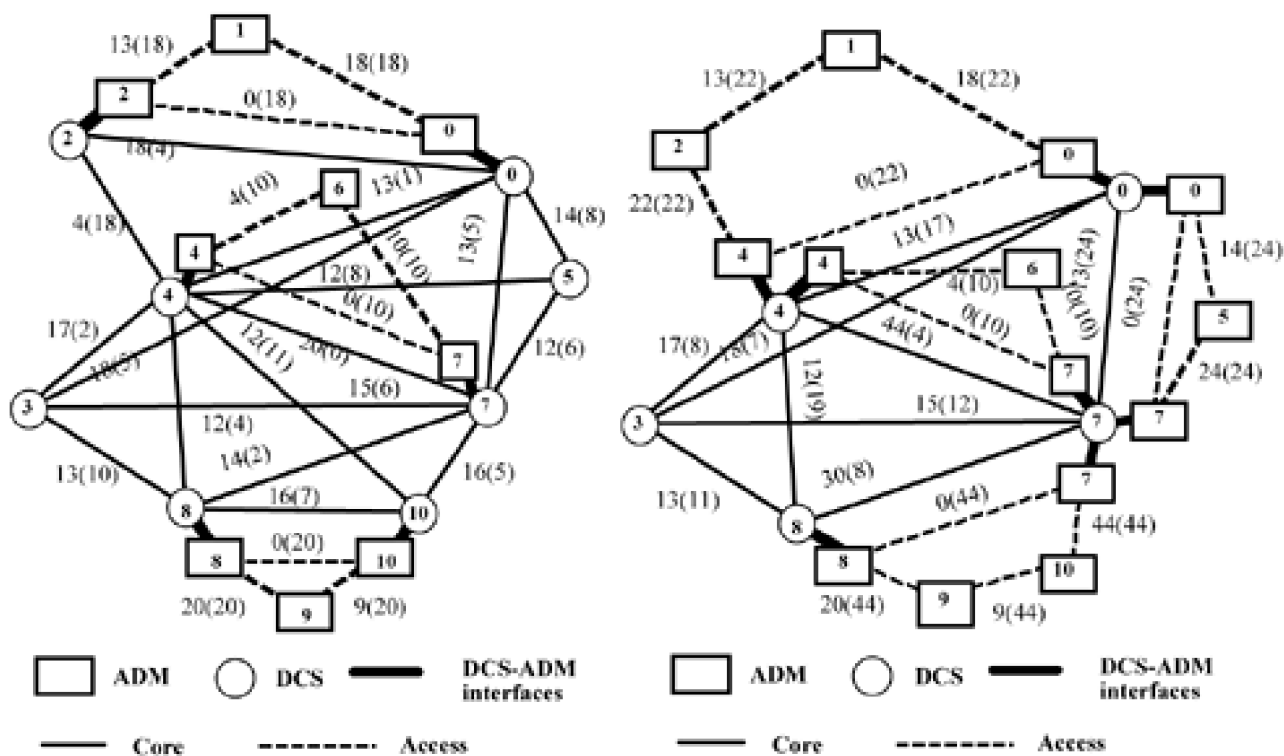


Table 11-1. Details of the Access Ring Designs

Partitioning	Core Nodes	Access Rings	Eliminated Spans	Working rerouted	Re-routed on:
$d_{th} = 3$	(0,2,3,4,5,7,8,10)	(0-2-1) (7-4-6) (8-10-9)	None	n/a	n/a
$d_{th} = 4$	(0,3,4,7,8)	(0-4-2-1) (0-5-7) (4-6-7) (8-7-10-9)	(0-2) (8-10) (4-5) (4-10)	18 16 12 12	(0-4-2) (8-7-10) (4-7-5) (4-7-10)

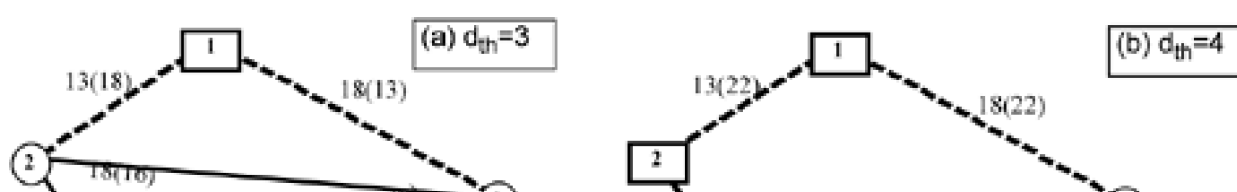
In the case of spans that were part of the core which were later overlaid by access ring spans, we kept the working demand associated with the mesh core. An improvement would be to refer some or all of such capacity from the mesh into any overlying access ring(s) as long as the ring is required on its own right and the amount of shifted capacity does not increase the $\max(w_i)$ of the ring. For example, given the formation of access ring (0-1-2) in [Figure 11-9\(a\)](#), the 18 units of w_i capacity on mesh span (0-2) could strictly be referred into the ring from the mesh, because the given ring already requires 18 units of protection capacity. This may or may not further reduce the spare capacity requirement of the mesh core design, depending on the forcer status of the span in question. If span (0-2) with 18 units of w_i capacity is a forcer of the core mesh design, then there would be a net capacity reduction on the complete hybrid design. Such opportunistic displacements of mesh capacity into overlying access rings was not done in the study reported, however, because at the time it was assumed that a large interface cost would be associated with the DCS-to-ADM transition.

11.4.2 Pure Mesh and Mesh-Chain Designs

For the reasons explained in [Section 11.3](#), the method for spare capacity design for the pure mesh and mesh-chain hybrids is identical once the topology and w_i values are determined after any span eliminations. In the results here, the pure mesh and mesh-chain designs were produced by the SLPA algorithm [\[GrBi91\]](#) which is typically within 10% or less of optimal. A hop limit of six spans was used in the mesh capacity designs. Although the capacity design method for mesh-chain and pure mesh are the same, the designs differ in general because the formation of access chains may involve span eliminations. For the present mesh-chain designs, the same span eliminations apply as in the corresponding set of access ring choices in the ring-mesh hybrids. In other words, the access chains formed for the mesh-chain correspond to the same access rings as in the ring-mesh hybrids, but all ring-mesh overlap spans are deleted to form chains instead of rings.

The other difference between the pure mesh and mesh-chain designs is in their types of nodal equipment and technology assumptions. In this study, a pure mesh design is assumed to require a full cross-connect capability in every node, to participate and function without modification in a mesh restoration protocol. In contrast the mesh-chain designs have the same capacities but employ ADMs in each node of the access network partition and we assume that the cross-connects at the mesh boundaries emulate the ring switching protocol and convert its signaling states to/from the more general rerouting protocol of the mesh core. [Figure 11-10](#) shows the mesh-chain designs for the two partitionings that are considered. [Figure 11-11](#) gives the corresponding pure mesh design for comparison. The annotation on spans gives the working and spare capacity respectively.

Figure 11-10. Mesh-chain hybrid designs based on (a) $d_{th}=3$, (b) $d_{th}=4$ access/core partitioning.



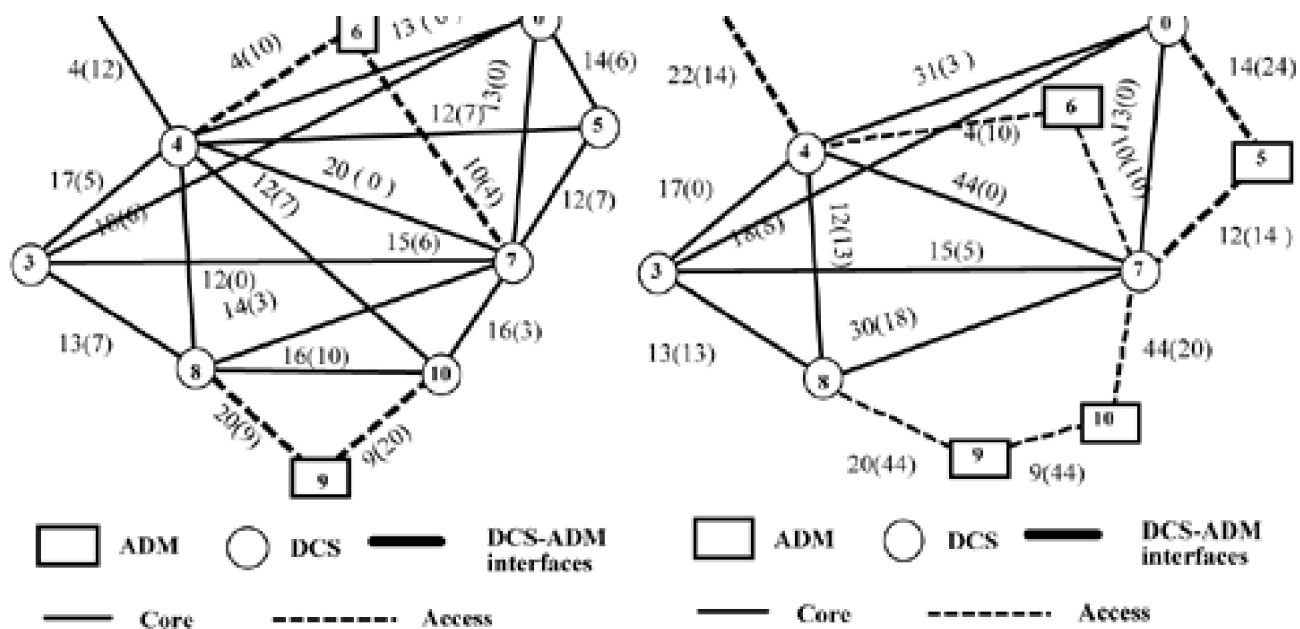
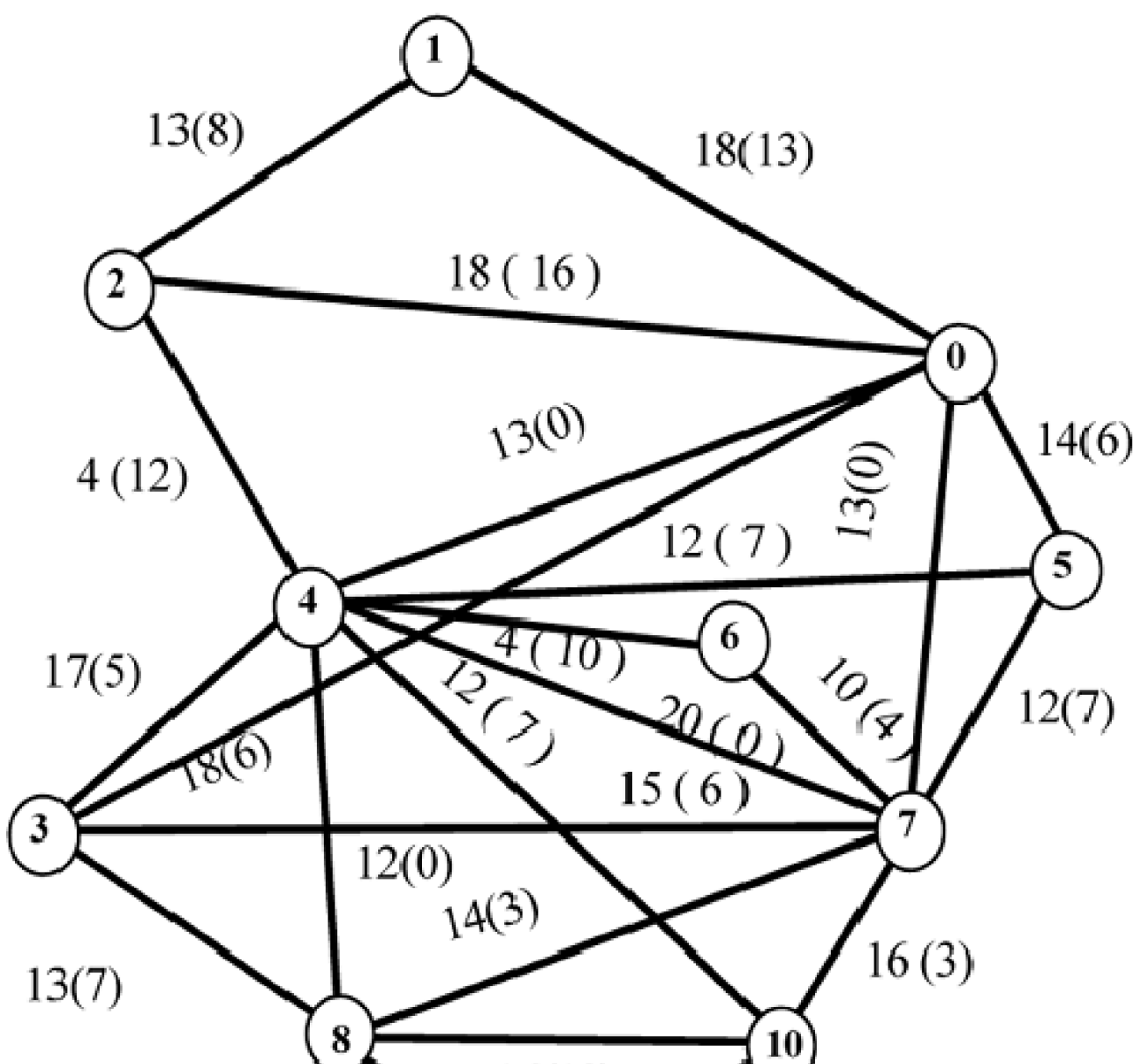
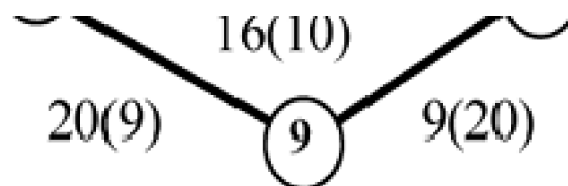


Figure 11-11. Pure mesh reference design (by SLPA [GrBi91]).





11.4.3 Pure Ring Reference Designs

To complement the treatment of the ring-mesh and mesh-chain hybrids it is relevant to also consider a pure ring reference design. To do so, we used an optimization model for minimum-cost span coverage using idealized unit-capacity rings. The unit-capacity aspect is artificial in the sense that real ring systems are modular, but to produce a fair comparison to the mesh and hybrid designs, we need to adopt a unit-capacity orientation because that is the basis of the other designs. The optimization model, called Span Coverage Integer Program (SCIP) is drawn from separate work on the multi-ring network design problem [MoGr99][Morl01]. It solves the problem of placing a set of capacitated BLSR-type rings on a specified graph so that w_i quantities on each span are fully protected, with minimum total protection capacity on the network. A unit-capacity ring cover actually represents a lower bound on the capacity of any real (modular-capacity) ring cover. In the results here, there was no limit imposed on cycle length for the rings. The pure ring design is completely based on ADMs as the nodal equipment elements. It is, however, problematic to sum ADM counts from the idealized unit-capacity rings of the SCIP solution because the unit-capacity design implies a high number of individual logical rings. For a more practical estimate of the number of ADMs in the pure ring designs we rely on another pure ring design for this network from [Gro92], which employs five modular-capacity rings. Both the SCIP design and the five-ring design are based on ring formation to cover all span capacities with no span eliminations. Additionally, the ADM count given is based on the assumption of an ADM in each node traversed by a ring. It is therefore an upper bound on the actual number of ADMs in a practical design. At sites served by several rings, demand grooming into rings can remove the necessity for every ring through the location to be equipped with a full ADM.

11.4.4 Results and Discussion

Table 11-2 summarizes the comparative measures of the pure mesh, pure ring and hybrid designs. The second and third columns give the total working and spare capacity (in terms of logical channel-hops) in each design. As expected, the pure mesh design uses the least capacity and the most cross-connects. The ring-mesh design based on $d_{th}=4$ partitioning (and its four span eliminations) has the greatest replacement of cross-connects with ADMs (other than for the pure ring) and also the highest total capacity of any design.

Table 11-2. Comparison of ring-mesh, mesh-chain, and pure reference designs

Design	# working	# spare	# total	#ADM	#DCS
Pure mesh reference	313	159	472	0	11
Ring-mesh ($d_{th}=3$)	313	246	559	9	8
Ring-mesh ($d_{th}=4$)	371	476	847	14	5
Mesh-chain ($d_{th}=3$)	313	159	472	3	8
Mesh-chain ($d_{th}=4$)	371	288	659	6	5
Pure (unit-capacity) BLSR Ring SCIP solution	313	336	649	n/a	0
5-ring design [Gro92]	313	440	753	27	0

The pure ring reference designs (with no span eliminations) uses the most ADMs, no cross-connects, and interestingly requires less total

capacity than the $d_{th}=4$ ring-mesh hybrid design. This is an interesting outcome arising from the effects of span elimination in forming the access rings, and would be an indication in further studies to inspect alternative designs using more access rings and fewer span eliminations for this case. The high spare capacity of the $d_{th}=4$ ring-mesh design (476) is also an indicator that it is suffering from too much referral of deflected span-eliminated working flow into the mesh core design. The overall economic merit of this design could still be good, however, as the fewest cross-connects of any design are used.

The mesh-chain designs are clearly advantageous, however, over the ring-mesh alternatives. They use the same low number of cross-connects as the corresponding ring-mesh hybrids, but also use fewer ADMs and less total capacity. Regardless of the cost vector for capacity and nodal equipment types, the mesh-chain designs are therefore *categorically lower in cost* than the ring-mesh because they use fewer of every type of resource. Whether either of the hybrids is also less costly than the corresponding pure-mesh or pure-ring designs depends on the relative costs of capacity and nodal equipment. In the extremes where nodal costs outweigh transmission costs, pure ring designs should always win cost-wise, on the assumption that ADMs are significantly less costly on a per-unit-capacity basis than cross-connects. At the other extreme where transmission cost dominates, pure mesh should always prove-in for the converse reasons. The middle ground is where ring-mesh or mesh-chain hybrids have the potential for costs to be minimized below either pure architecture by finding the right combination of mesh attributes for transmission efficiency and use of ADMs for nodal cost reduction. A clear conclusion, however, is that in terms of *both* capacity and nodal equipment costs mesh-chain hybrids are superior to ring-mesh hybrids. To access this opportunity, however, we need to implement ring-mesh signaling protocol interworking—at nodes such as 2 and 8 in [Figure 11-7](#)—so that mesh-oriented restoration processes can inherently cooperate and compliment the existing ring-oriented restoration process that ADMs support.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

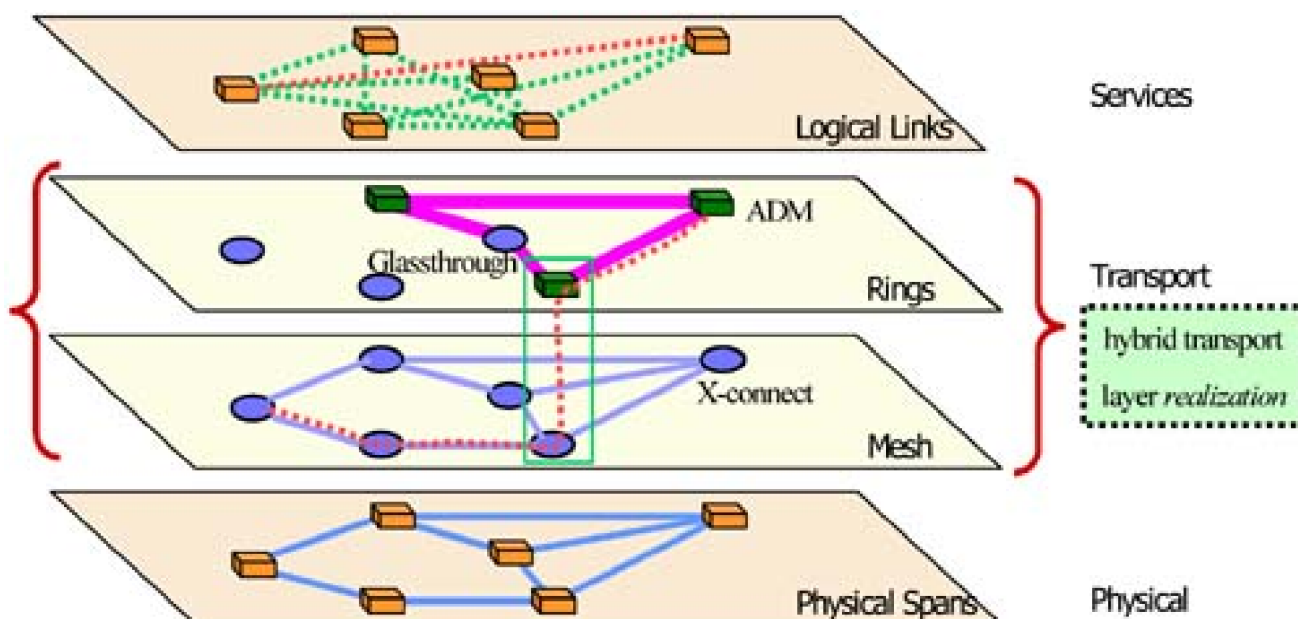
11.5 Optimal Design of Ring-Mesh Hybrids

This section is devoted to establishing an optimal model (in the sense of cost) for a completely generalized ring-mesh hybrid. It is a general model in that no *a priori* principle (such as an access/core partitioning) is assumed. The practical difficulty with the model is its computational complexity for network sizes of practical interest, hence the need for an effective heuristic, which follows. From a research standpoint, however, the optimal model serves to define the problem and, where solutions are feasible, it allows us to check our later hypothesis about what constitutes a synergistic set of rings and co-designed mesh. The model that follows envisages a span-restorable mesh in conjunction with BLSR or UPSR type rings. Demands are routed before the optimization of ring placements and mesh capacity. All w_j capacity crossing a span is either included within a ring or is protected by restoration flows in the mesh component.

11.5.1 Concept of a Single-Layer Ring-Mesh Hybrid Transport Network

An important initial clarification is to stress that the generalized hybrid of rings and mesh that we now consider does not involve any *escalation* or a multi-layer successive response to failures. This is a common misunderstanding. While it will be natural to speak of the rings overlying the residual mesh in this type of hybrid, they are really embedded in the mesh and do not constitute a multi-layer restoration scheme in the usual sense of works such as [Dem99] for example. In this hybrid architecture each length-wise segment of an end-to-end path is protected by *either* the mesh or ring components of the hybrid. The action of both hybrid components is logically and temporally concurrent, not successive, and each acts independently and autonomously for protection of the working capacity in their domain. Restoration in this type of hybrid is not first attempted in one component and then handed "up to" another layer. These hybrids still constitute a single layer restoration scheme, but have a mixture of ring and mesh logical components. Figure 11-12 illustrates. Faults occurring in the physical layer are restored in the transport layer before reaching the service layer. The realization of the transport layer considers a generalized mixture of rings amidst a corresponding mesh. There is, however, no new interlayer communication, coordination, or escalation of responses required.

Figure 11-12. A generalized single-layer ring-mesh hybrid transport network.



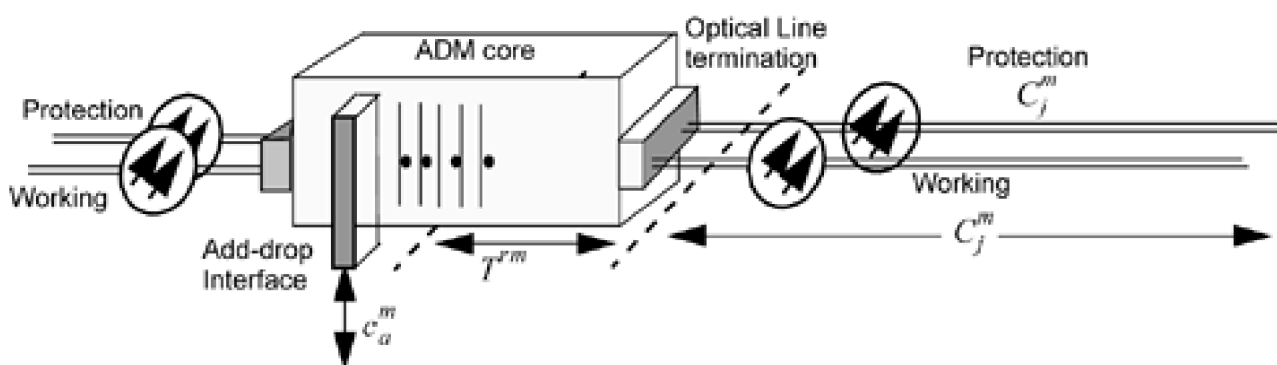
11.5.2 Cost Modeling for Ring-Mesh Hybrids

A challenge in developing an optimization model for assemblies of ring and mesh network elements is that the cost structure for the two types of equipment are not the same. In comparing one mesh approach against another mesh alternative, details of the cost structure may not be so important. As long as nodal equipment types are the same then total capacity indicates which alternative is economically superior, regardless of absolute costs. But to consider hybrid architectures, we need a more detailed consideration of the cost structure for ADMs, cross-connects, and transmission capacity. The modeling of cost for ring systems particularly requires attention. The most important way that the cost structure of ADM rings differs from mesh network models is that ring systems are fundamentally modular. In contrast to a cross-connect that can manipulate single channels as the basic unit of spare capacity, an ADM is quite modular in its protection capacity. In comparing mesh designs we often adopt an integer channel capacity model but the fundamentally modular nature of rings requires that in approaching ring-mesh hybrids we must adopt a fully modular capacity view for the rings, and preferably for the mesh domain as well.

Let us define the following parameters for cost modeling in a ring-mesh hybrid environment. First are the attributes of pure optical line transmission, where there are no differences between ring or mesh. These are characterized as follows and as indicated in [Figure 11-13](#):

- C_j^m = This is the cost of a transmission module of the m^{th} capacity size, regardless of its use as part of a mesh span or to help form a ring.
- M is a set of modular transmission capacity sizes, for example OC-48, OC-192, OC-768 for SONET ADMs or 40, 80, 160I for OADMs, index m .
- Z^m = The number of capacity units for the m^{th} module size.

Figure 11-13. Generic Equipment Cost Model for a Add-Drop Multiplexer (ADM).



The following attributes reflect the type of nodal equipment that terminates the optical line:

- $T^{\text{mesh},m}$ = cost of terminating a transmission module of the m^{th} capacity on a cross-connect, including a fully-allocated cost of the OXC core. For instance, if a fiber pair bearing 80 (bidirectional) ls terminates on an OXC of ultimate size of 4096 fibers then $T^{\text{mesh},m}$ reflects the direct cost to interface the fiber pair plus an allocated cost of the overall OXC core capacity that it consumes.
- $T^{\text{ring},m}$ = cost of terminating a transmission module of the m^{th} capacity on a ring ADM including half the cost of the ADM node core of the corresponding total capacity.

In practice, the allocation of core cost to each line system terminating on the OXC may involve separate consideration of both the number of fiber ports consumed and the number of wavelengths the fiber brings to the core for switching. This recognizes that the OXC may have separate ultimate capacity limits in terms of the number of fiber terminations and the ultimate number of ls that can be switched. The cost accounting for consumption of the ADM common core infrastructure is much simpler. It is a discrete investment that is completely consumed by its two optical line interfaces. Building up from these primary parameters we also define:

- $\alpha_k^n = 1$ if the k^{th} member of a set of candidate ring systems includes an ADM at node n , 0 otherwise. This recognizes that there may not necessarily be an ADM at every node of a ring. When no ADM is present a glass through simply continues the fiber continuity through that location.
- $r(k)$ = the cycle of the graph on which ring candidate k is formed.
- $C_r^{ring, m}$ = Complete cost of constructing the r^{th} candidate ring at module size m , including the one-time cost of the ring ADM cores and line-rate terminations (in $T^{ring, m}$) plus line transmission costs, but excluding any per-channel add/drop interface costs (which are dependent on actual capacity assigned to the ring):

Equation 11.1

$$C_k^{ring, m} = 2 \cdot \left(\sum_{j \in r(k)} C_j^m + \sum_{n \in r(k)} \alpha_k^n \cdot T^{ring, m} \right)$$

- C_a^m = cost of add/drop for each demand that enters or leaves a ring at an ADM of modular capacity m . This cost may be essentially zero under the integrated OADM/OXC equipment model of [Section 11.1](#).

As written, [Equation 11.1](#) is for a 4-fiber ring where the ring needs two of the basic line capacity modules for each of its spans; one for working, one for protection. The leading factor of two can be dropped from the cost coefficient for a 2-fiber BLSR but then only half the corresponding line-rate capacity is obtained as working capacity. It should be noted at this point that the strict representation of ring candidates for the hybrid design is complicated by the fact that associated with *each cycle* on the network graph there are

Equation 11.2

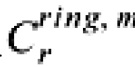
$$|M| \cdot \sum_{h=2 \dots n} \binom{n}{h}$$

distinct *ring systems* that could be built, where n is the number of physical node locations on the ring and h is the number of these that have ADM terminals (as opposed to glass-throughs) on the ring. M is the set of different line rates for ring systems.

At least in the case where demand routing follows the shortest path over the graph prior to any ring or mesh placement decisions, the latter consideration is not so explosive in the combinatorial sense as it may first seem. In this case we know ahead of time which demands will be loaded into any prospective ring if the ring is placed and we can base its cost as a candidate ring on this information. More specifically, we can assume that if a ring is placed on cycle r of the graph, it will be used to displace directly underlying working capacity from the graph to its fullest extent. We therefore know which underlying path segments would be displaced into the ring and are able to deduce the total for add/drop interface costs (if any) for each ring by noting where the route of a demand enters and leaves the ring of

interest. Thus, a preprocessing produces the cost coefficients to set up the MIP tableau. In particular, $C_r^{ring, m}$ represents the total cost for the proposition of building a specific ring type, of the m^{th} modularity, over cycle k on the graph, loaded with maximal amount of working capacity flows from the underlying spans of the original shortest-path routing plan, and accounting for add/drop costs at each ADM, if relevant. Any site where an ADM core is not implied by the routing of demands over the graph, is considered to be a glass-through

site. ^[2] It follows therefore, that although it is a detailed process, the $C_r^{ring, m}$ coefficients are precomputable for each candidate ring. Later, we use a heuristic to predetermine a relatively small number of high-promise candidate rings for the hybrid, thereby limiting the total

workload required for  evaluation.

[2] Even the view of having *either* a glass-through or an ADM terminal is over simplified here. Locations on an optical ring may actually have a splice (a glass-through), an EDFA, an OADM configured in pass-through mode serving as a regenerator, or a full OADM node with one or more local add/drops.

Two other cost components can be acknowledged for completeness, but dismissed because they provide constant terms in the total cost function for any hybrid that we consider. These are the first cost of establishing a cross-connect infrastructure in every node, plus the cost of one pair of tributary rate add/drop interfaces on a DCS for every demand at its origin and destination. These contribute a constant total cost that does not change across the hybrid designs here and are therefore omitted from the optimization model.

11.5.3 Ring-Mesh Hybrid Design Model

Having above acknowledged some of the more difficult aspects of a precisely modeling cost for accurate optimization of ring-mesh hybrids, we can nonetheless realize a fairly useful model for generalized hybrids. By way of summary, the following assumptions apply:

1. Demands are shortest-path routed over the graph and will stay on these routes whether associated with either ring or mesh components in the final design.
2. The mesh component operates on the basis of span restoration.
3. There is no cost to make a transition from ring to mesh en route of a demand, by virtue of the integrated ADM functionality hosted on the cross-connects (as in [Section 11.1](#)).
4. Under the assumption that all inter-ring transitions are DCS-managed, and the assumption of integrated ADM functionality, there is also no cost for a demand to make a transition from one ring to another.

In addition to the cost modeling parameters defined above, the following parameters are common to both ring and mesh components of a hybrid:

- S is the set of spans in the network, individual spans are indexed by i and j , where a second span must be simultaneously specified, by j .
- W_i^0 is the logical working capacity on span i if the demands are shortest path routed (or routed by any other method) over the facilities graph prior to the hybrid design.
- Z^m is the number of *usable* units of working capacity provided by the m^{th} size of transmission module. In the mesh this capacity is fully usable to support accumulations of both working and spare channels. In a ring Z^m actually represents half of the line capacity, i.e., the *working* capacity of the ring. A matching amount of protection capacity is reflected in the corresponding ring cost coefficients, but only the working capacity of the ring appears explicitly in the model. Furthermore, the ring working capacity is only the same as the corresponding module size in the mesh if the ring is a 4-fiber BLSR. For 2-fiber BLSRs, the ring use of the corresponding transmission capacity is reduced to $Z^m/2$.

Parameters specific to the mesh component design are:

- P_j is the set of all eligible routes for mesh restoration of span failure i , indexed by p .
- $\delta_{i,j}^p = 1$ if the p^{th} restoration route for span i uses span j , 0 otherwise.

Solution variables that describe the mesh network component of the hybrid are:

- H_j^m

- n_j^m =Number of modules of the m^{th} size added on span j (integer variable).
- w_j =Mesh logical working capacity needed on span j to serve demands.
- f_i^p =Restoration flow through the p^{th} route for failure of span i .
- s_j =Mesh logical spare capacity needed on span j to support restoration flows in the mesh.

The ring-related design parameters are:

- R = Set of all distinct simple cycles of the graph (i.e., templates for rings), index r .
- $\delta_j^r = 1$ if the r^{th} cycle includes span j .

And the ring-related solution variables are:

- n_r^m =Number of instances of an m -capacity ring placed on cycle r in the design.

The variables specify the ring-mesh hybrid structure in terms of the number and size of self-contained ring systems to place, the number and size of modular transmission systems between OXC of the mesh component, the logical partitioning of this capacity into working and spare, and the routes to be used for restoration of the working capacity in the mesh component. The ring restoration routes and protection capacity are implicit in specification of the ring cycle, r and modularity m . The model for a minimum cost ring-mesh hybrid in terms of the above is:

Hyb-1:

Minimize:

Equation 11.3

$$\sum_{m \in M} \left\{ \sum_{r \in R} n_r^m \cdot C_r^{\text{ring}, m} + \sum_{j \in S} n_j^m \cdot (C_j^m + 2 \cdot T^{\text{mesh}, m}) \right\}$$

Subject to:

1. Mesh restoration flows protect all working capacities in the mesh component:

Equation 11.4

$$\sum_{p \in P^i} f_i^p = w_i \quad \forall i \in S$$

- 2.

Spare capacity on spans of the mesh suffice to support the restoration flows:

Equation 11.5

$$\sum_{p \in P^j} f_i^p \cdot \delta_{i,j}^p \leq s_j \quad \forall i, j \in S^2$$

3. Any working capacity on a span not covered by the mesh must be in an overlying ring:

Equation 11.6

$$\sum_{m \in M} \sum_{r \in R} n_r^m \cdot \delta_j^r \cdot Z^m \geq w_j^0 - w_j \quad \forall j \in S$$

4. Modular capacity in the mesh network:

Equation 11.7

$$\sum_{m \in M} n_j^m \cdot Z^m \geq w_j + s_j \quad \forall j \in S$$

All variables are nominally integer although the relaxation of mesh-restoration flow variables f_i^p is reasonable. Additionally, depending on demand volumes and ring capacities, it might be reasonable to assume at most one instance of any ring candidate in the solution,

thereby replacing n_r^m by a binary decision variable, i.e., instead of deciding how many copies of an m -modular ring to commission on cycle r , we decide simply whether such a ring is—or is not—in the solution.

The objective function is the total cost of ring systems placed plus the total cost for modular transmission systems and line rate termination on OXC. Ring OADM node costs are included in $C_r^{ring, m}$ and allocated costs of OCX cores in the mesh are reflected by $C_r^{mesh, m}$. As mentioned, routing-dependent add/drop costs are ignored as either a constant or zero cost item under the assumptions above. The rings are inherently modular and their cost includes the built-in protection capacity for ring-type survivability. For this reason the model needs no explicit expression of survivability considerations for the ring component of the hybrid. Any working capacity referred into a ring is automatically protected.

Set R contains all the simple cycles on the network graph, up to some limit on hop length. The significance is that R contains all the possible topological layouts for rings that could be considered in the design. Each cycle is notionally a template for the layout of a possible ring (or stacked ring(s)) of any size or type considered. The preprocessing for both the MIP formulation and the later heuristics require finding the set R , or a working set of candidate cycles that is limited by some criteria such as hop limit. The algorithm by Johnson [75] or the related algorithm in Section 4.10.2 are suitable for this.

The first set of constraints ensures that the working channels on each span in the mesh are restorable by asserting that there are enough individual paths assigned to eligible routes for restoration of the span to cover its working capacity. This assertion of required restoration paths is countered by the second constraint system, which ensures that the assignment of paths to routes is feasible under the spare capacity assigned to each span. The spare capacity variables influence the cost of mesh capacity in the objective through the modularity constraint, Equation 11.7.

In Equation 11.6 we assert that any working capacity traversing span j that is not handled in the mesh must be covered by the capacity of overlying rings on that span. This involves the ring modular capacity sizes, the ring placement decision variables and the interception of

ring capacity that overlies span j . This constraint system interacts with both parts of the objective function: through n_r^m it has a drive to effect the fewest and smallest rings. But any ring capacity placed can displace working capacity from the mesh and so also has the

prospect of reducing n_j^m through increasing n_r^m .

What makes this trade-off quite intricate, however, is that an increment to an n_r^m variable implies an entire ring addition to the network,

whereas R_j^{opt} speaks only of the capacity on one specific span of the mesh. Intuitively, it may seem that there would be only relatively rare opportunities, with a high degree of coordination between ring and mesh components, to justify triggering a ring addition. Any such synergy would have to arise at the network level as a whole because placing a whole ring to displace capacity from *one* mesh span would rarely, if ever, make sense in its own right. In fact a particular insight about just when this formulation would find it advantageous to displace capacity from the mesh into overlying rings follows, and is the basis for the heuristic in the next section. Results with the optimal design model follow in [Section 11.6.4](#) where it is used as a reference model to assess performance of this heuristic.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

11.6 Forcer Clipping Ring-Mesh Hybrids

In the Hyb-1 model above it is interesting to think further about the interactions between constraint systems. In particular, we need to ask when would the solver trade an increase in H_r^m for a decrease in H_j^m on some span(s) of the mesh? After all an increase in H_r^m means a decision to completely place one additional ring system, whereas decrementing H_j^m saves mesh capacity on only one span. A reasonable hypothesis would be that trading H_r^m for H_j^m only happens when some new ring placement would have a more than proportionate impact on the design of the mesh counterpart. The concept of forcers, introduced in [Section 5.5](#), and the idea of *forcer clipping* gives us just such a view of how certain rings can be especially synergistic with the surrounding mesh in which they are embedded as a hybrid. The key idea is that by clipping off strong forcers of the mesh, certain rings can contribute to enhancing the efficiency of the mesh and, if they themselves are well loaded by the capacity they have "clipped off" of the mesh, then the placement of such a ring may be of overall economic benefit. In the access / core partitioned hybrids the use of ring and mesh principles is more or less separated spatially. In contrast, the heuristics that we now develop based on the forcer clipping idea are much more general and have no *a priori* bias as to how and where to use rings. Here, rings and mesh will not be topologically separated, but selected rings will be commissioned within the body of a mesh-restorable network that have a forcer clipping effect on the efficiency mesh network design.

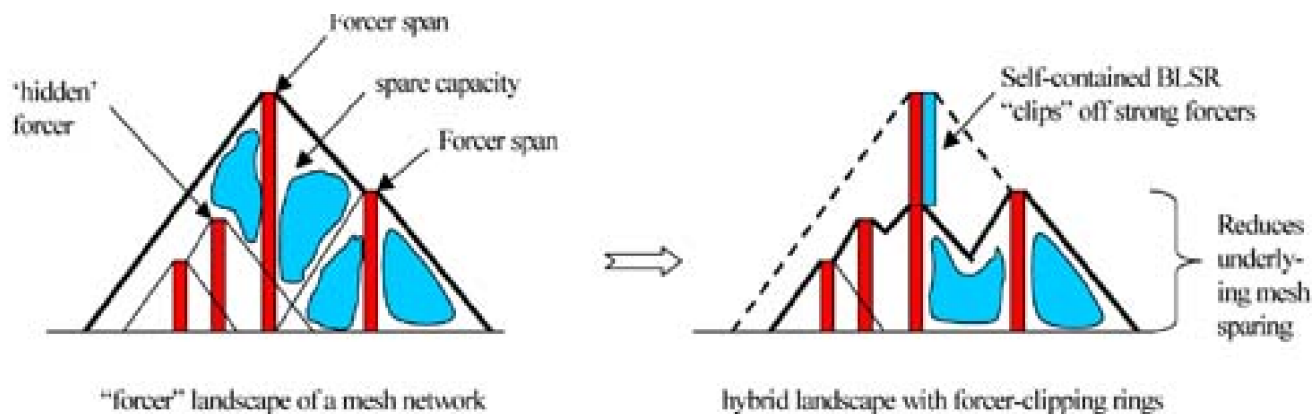
11.6.1 The Forcer Clipping Hypothesis

[Chapter 5](#) introduced the forcer concept and in [Chapter 8](#) it was applied to issues involved with nodal bypass and survivable design with express routes. The idea now is to use forcer analysis to tell us which spans are most advantageous for ring placement by virtue of having the greatest network-wide effect of reducing spare capacity in the remaining mesh component. The direct reduction of working capacity in the mesh by ring placement is another beneficial effect to the mesh component but, by itself is generally detrimental to total cost because the ring use of protection capacity is less efficient. Good hybrid ring placements must therefore be at least cost neutral in terms of the direct shift of working capacity from one domain to the other, but have some more than proportional cost saving effect on the spare capacity of the mesh from which the working capacity was displaced. This is where the forcer concept comes into play.

Recall that a basic implication of the forcer concept is that reduction of the working capacity on certain spans of a mesh-restorable network would yield no corresponding reduction of mesh spare capacity. Even if these non-forcer spans are reduced, working flows across other spans, i.e., the *forcers*, require retention of the spare capacity to ensure restoration. Measures such as Fx^* let us pinpoint which spans most drive the spare capacity requirements of the surrounding restorable mesh. In effect Fx^* reflects the "height" by which span's working capacity is above the point at which it would no longer be a forcer (i.e., other spans would become forcers, halting the relief of spare capacity).

What can we do with knowledge of how the spare capacity of the mesh would be relieved if some specific working span quantities were reduced? One use could be to reroute working paths, detouring them from their shortest paths so as to reduce w_j quantities on the strongest forcers. This is conceptually valid and is essentially the explanation of why joint optimization of working path routes and spare capacity placement in a mesh network is beneficial. But another way that strong forcers of a mesh network might be reduced to improve the overall efficiency of the mesh would be to try *clipping these forcers down with ring systems*. Especially if one ring can group together several strong forcers and provide for their survivability within one ring, there might be a more than proportionate cost reduction in the remaining mesh network. Hence the term "forcer clipping" for the hypothesis that a ring might be placed on the mesh network to clip the tops off of one or more of the forcer spans, thereby more than proportionally reducing its total working and spare capacity cost. [Figure 11-14](#) is a conceptual illustration. Net cost reductions would arise if the cost of the forcer-clipping ring is less than the net savings in the underlying mesh layer after its working capacity is reduced and its spare capacity is re-optimized.

Figure 11-14. The concept behind "forcer clipping": rings that make a mesh more efficient.



This is the central idea from which we proceed toward construction of heuristic for generalized ring-mesh hybrid design. *A priori*, there are several reasons to expect that certain rings could pay in on the basis of forcer-clipping effects. One is a similarity to the concept of express rings already known in ring network planning. An express ring is one that is sized and situated, typically with relatively few add/drop terminals, to handle just a few of the largest demand flows in the network. In fact a 1+1 DP can be viewed at a two-terminal express ring and is sometimes used as such to handle single particularly large demands in a ring network design. By getting these dominant flows out of the way in their own semi- or fully-dedicated express rings, the rest of the multi-ring design problem can be more effectively solved for the remaining more similarly sized demand bundles. In that sense the forcer clipping ring concept is like proposing a special kind of express ring to relieve the effects of certain demands on the mesh network design.

Another line of reasoning that supports the forcer clipping hypothesis in principle is the nonlinear dependence of the redundancy of a mesh network on the diversity of its w_i values. Consider one node in isolation that has d spans. A necessary (but not by itself sufficient) set of conditions for full restoration of every span cut is that at any node adjacent to a span failure i :

Equation 11.8

$$\sum_{j \neq i} s_j \geq w_i \quad \forall i \in 1 \dots d$$

which further implies that

Equation 11.9

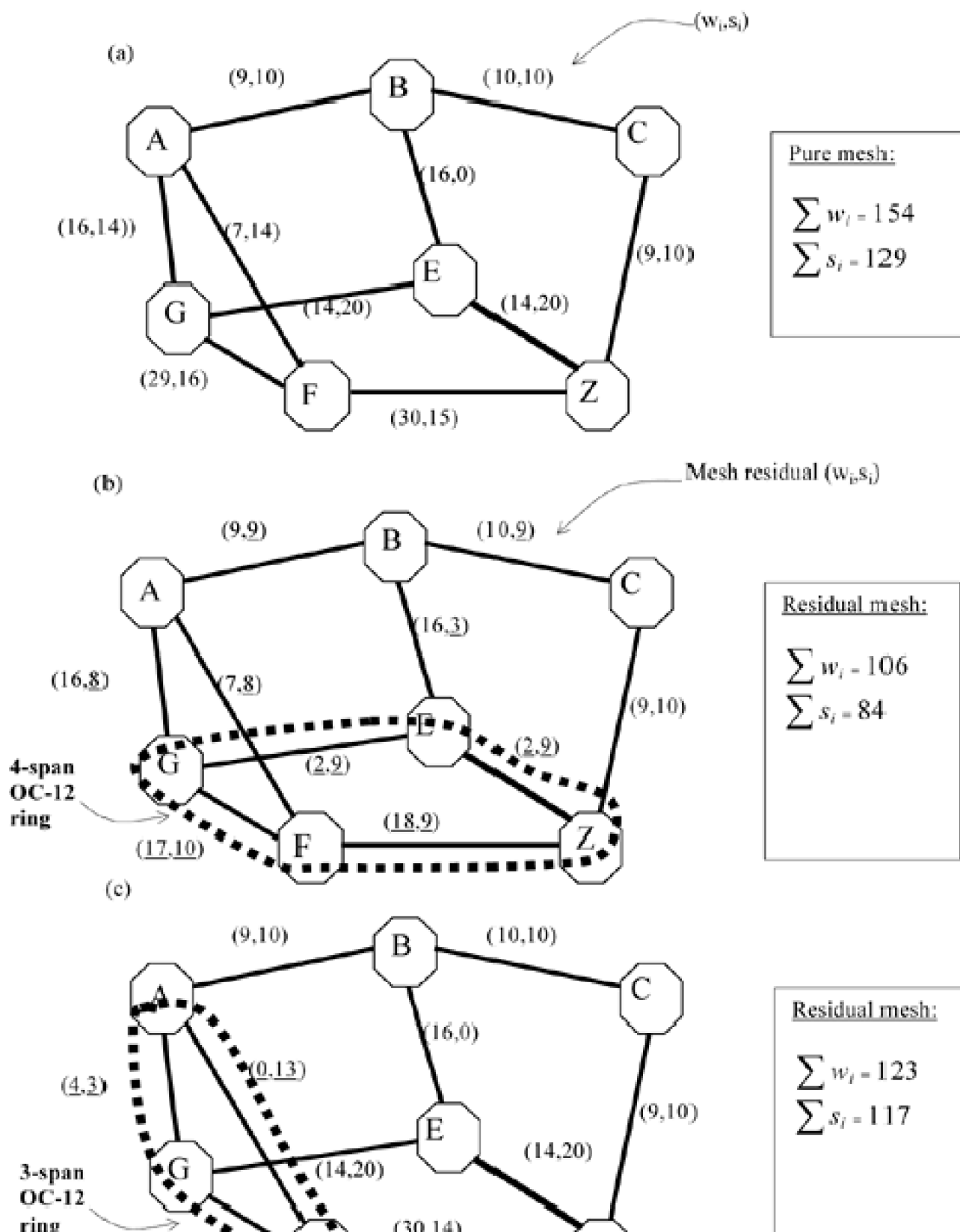
$$\sum_{i \in 1 \dots d} s_i \geq \max(w_i)$$

is a lower bound on the total sparing required at the node. With a few more steps this leads again to the lower bound on the redundancy of a node being $1/(d - 1)$ ([Section 5.4.2](#)).

Given [Equation 11.8](#) and [Equation 11.9](#) it is possible to show how important it is to "level" the w_i quantities. Consider for example, a node of degree $d=4$, let $w_i = (3, 15, 3, 15)$. The minimal sparing vector will be $s_i = (5, 5, 5, 5)$ for a node redundancy of 55.5%. Now consider the node being overlaid with a 12-channel ring to "clip off" most of the high flow of 15 units going through this node. In the residual mesh this node has $w_i = (3, 3, 3, 3)$ and requires $s_i = (1, 1, 1, 1)$ for a redundancy of 33%, which is actually at the lower bound for $d=4$. This illustrates the notion of "forcer leveling": in the latter case, every span is an equal co-forcer of all other spans at the node. The potential economic benefit is in the trade off of the ring cost against the direct savings in displaced working mesh capacity plus the benefits of redundancy reduction in the residual restorable mesh. In this isolated node example, the ring provides 12 working capacity units on two spans but the total $(w_i + s_i)$ that it displaced from the mesh is $24 + 16 = 40$ units. Even if the exchange of the forcer clipping working capacity from the mesh to the ring by itself was cost neutral, net savings still arise whenever the ring placement has the added leverage of reducing spares in the mesh due to forcer-leveling effects which inherently increase the mesh efficiency.

Figure 11-15 moves up from the isolated node considerations to show a small quantitatively exact example of effects from rings embedded in a mesh. Figure 11-15(a) shows a 7-node, 10-span, pure mesh span-restorable network of unit capacity modularity. The (w_i, s_i) note by each span gives the working capacity on the span (generated by shortest path mapping of the demand matrix onto the topology), and the spare capacity assigned (by SCA) for 100% span restorability with a hop limit of five. The pure mesh reference design of Figure 11-15(a) has a total of 283 (working plus spare) channels.

Figure 11-15. Illustrating the effects of various forcer-clipping ring candidates.





[Figure 11-15\(b\)](#) shows the effect of overlaying one ring that turns out to be fairly effective in forcer clipping. On a simple bulk capacity basis this 4-span 12-channel ring represents an investment of $4(2 \times 12) = 96$ (working plus spare) capacity units. Its placement alters four of the w_j values in the residual mesh, and the corresponding changes in spare capacity are shown (underlined values). The residual mesh has a revised total of 190 ($w_j + s_j$) channels. Thus, the placement of this particular ring reduces the mesh by 93 channels in return for a bulk equivalent capacity of 96 channels invested in the ring. The ratio, called the *capacity return ratio* (CRR) is $= 93/96 = 0.969$ in this case. The high CRR in this case is an indication of a ring placement that is most likely economic because, on a bulk bandwidth basis including nodal costs, most ring systems cost considerably less than 97% of the cost for the equivalent point-to-point capacity and termination in a mesh technology environment. Thus, the CRR numerically represents the discount factor for ring capacity (relative to mesh) below which a given ring placement would yield a net savings. More detailed inspection of the example in [Figure 11-15\(b\)](#) shows that it is a combination of good working fill on the ring spans and good forcer clipping effects that make this ring effective. [Figure 11-15\(c\)](#) is a counterexample to show that by no means will all possible ring placements necessarily be economic. Through the same process it can be seen that the CRR of the ring in [Figure 11-15\(c\)](#) is 0.60. Thus, if the cost of the ring was more than 60% of the cost of the corresponding capacity in the mesh, the placement of this ring would be a losing proposition. The subsequent heuristics mechanize the process of finding near optimal sets of several such rings to embed in a mesh residual.

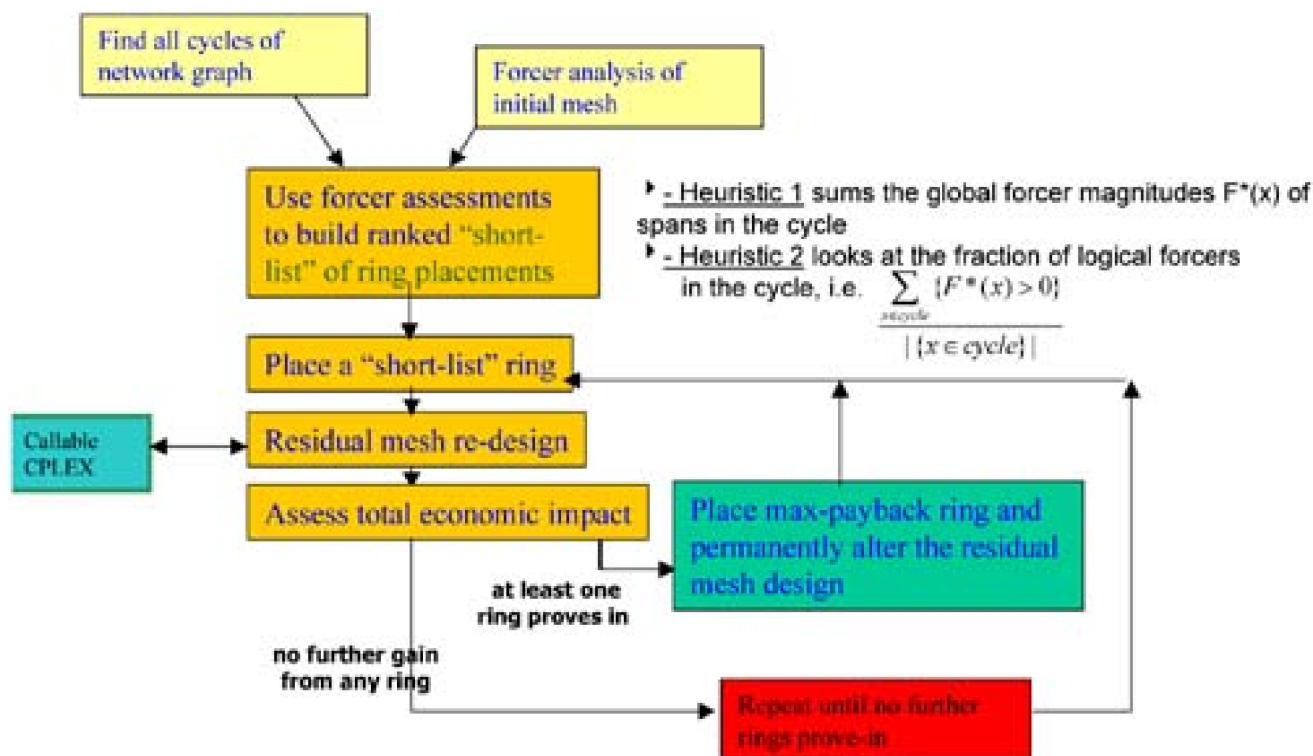
11.6.2 Forcer Clipping Heuristics

Two closely related heuristics are now discussed to place rings within a mesh network based on the forcer clipping hypothesis.^[3] Both use information from forcer analysis of the initial pure mesh design to identify a set of potentially good forcer clipping ring placements. After having identified this set of potentially good candidates the algorithms make trial ring placements, redesign the residual mesh network in detail, and apply a cost model to assess the net benefit of the particular ring trial placement. The ring with the best economic return is placed in the design and iteration repeats until there are no remaining ring candidates that produce an economic return. The basic algorithm is outlined in [Figure 11-16](#). First, the following preliminary processing is performed:

^[3] Work in [Section 11.6.2](#) to [11.6.4](#) was in collaboration with R. Martens and presented in preliminary form at [\[GrMa00\]](#).

- 1. Initial mesh design:** From the given graph topology and demand matrix a pure mesh restorable design is created as a starting point. In this case it is an SCA design using the methods of [Chapter 5](#). Demands are mapped onto the network spans by shortest path routing, generating the w_j quantities and SCA is then solved giving the initial mesh s_j values.
- 2. Forcer analysis of mesh:** Forcer analysis of the network is done as described in [Chapter 8](#). We obtain the forcer skeleton and Fx^* magnitudes of the forcers in it.
- 3. Cycle finding:** All elemental cycles of the network graph, up to a maximum hop size, are generated. This step identifies all the topological possibilities for ring placement on the network graph, populating the set R . The procedure and data is identical to that used in the optimal MIP model discussed in [Section 11.5.3](#).
- 4. Build working set:** Since the set R is typically large, it is at this stage that we use our ideas about forcer clipping to select a smaller set of "high promise" cycles that will act as ring system templates. This is also where the two heuristics differ. In both methods every cycle in R is inspected at this stage with respect to the forcer information from step 2 to obtain a forcer clipping figure of merit.
 - a.** In the first heuristic the metric used is the sum of the Fx^* magnitudes on spans that the cycle overlies. Promising rings are those that group together many forcers (or at least a few strong forcers), producing large positive Fx^* totals.
 - b.** The second heuristic looks only to see what proportion of all spans in a cycle have logical forcer status.

Figure 11-16. Basic heuristic for ring-mesh hybrid design based on forcer-clipping



The heuristics differ only in the criterion for populating the working set. The second metric has the computational advantage of requiring determination of forcer versus non-forcer status only, without assessing forcer magnitudes. The added simplicity of this permits an option to recalculate the forcer landscape and revise the working set after each ring placement iteration. On the other hand, the first heuristic responds to the strongest forcers that may be especially important to relieve in some cases. In each case, the working set is populated by choosing the N cycles with the highest ranked figures of merit. N , the working set size, is user selected. It is also possible to use both criteria to populate the working set, taking the top-ranked $N/2$ by each criterion. Following these one-time preliminaries, iterative ring selection and placement follows until no further economic payback is found:

5. Main loop:

[\[View full width\]](#)

Repeat

For every cycle in the working set:{

For every modular ring size:{

a) Place candidate ring of working capacity Z^m ;

b) Remove $\min(Z^m, w_i)$ capacity from underlying mesh spans;

c) Create ILP (or LP relaxation) of the residual mesh SCA problem;

d) Solve the SCA problem with CPLEX;

e) Obtain total spare capacity for the residual mesh;

f) Calculate mesh working and spare capacity relief and/or calculate detailed

➡ economic return for this placement trial; Store the benefit measure;

h) Remove the candidate ring and return the mesh capacities to the values before

➡ this trial} }

Place the ring with the largest positive economic return. Update the mesh capacity

➡ design with permanent adoption of the ring placed.}

Until (no positive economic return can be found).

In the event of a tie in economic return between ring candidates we place the ring which has the greater number of spans. Secondary ties are broken arbitrarily. The step of calculating the economic return is where an arbitrarily detailed cost assessment model, for both ring and mesh component, can be implemented in a network planning tool. This would include all specific costs for different types of transmission and DCS used in the mesh layer, span distances, and regenerator locations, existing fiber in ducts, types and sizes of rings available, and so on. As outlined in [Section 11.5.2](#) this can also include determination of all add/drop interface costs and the location of glass-through sites on each ring. One way to measure the effectiveness of the basic algorithm is, however, to use the same cost function as the objective

function in the Hyb-1 formulation ([Equation 11.3](#)). This permits comparison of results with the forcer clipping heuristics to corresponding optimal solutions, when the latter are possible. Accordingly, the heuristic here assesses economic return for its ring placement decisions as the difference in the value of [Equation 11.3](#) upon entry to the basic trial placement iteration, to its value after placing each candidate ring and updating the mesh SCA design. The heuristic can either solve the IP or LP relaxation of the mesh SCA subproblem. The ILP relaxation serves well as a fast surrogate for obtaining the redesigned mesh cost. Exact costs are not needed in that context, just accurate relative costs. Although we accept the relaxation in assessing trial placements, the assessment of final design costs from the heuristic against the optimal solutions is based on a true integer solution for the mesh SCA redesign. On the other hand, depending on problem size, the ILP for SCA may itself solve quickly enough that there is no real reason not to use the ILP version in the inner loop as well.

11.6.3 Methodology for Tests of the Forcer Clipping Heuristics

A set of results for the optimal method (Hyb-1) and for both heuristics was attempted on six test network designs. Net 1 is a U.S. metropolitan area network model from [\[YaHa88\]](#). It has a fairly high average nodal degree of 4.2. Net 2 was created by eliminating spans 2, 14, and 22 from Net 1 to create a related test case with a lower nodal degree ($d = 3.6$) than its parent. The purpose was to test the idea that if the demand matrix and most of the topology were otherwise the same, the savings from forcer-clipping rings should be *greater* in a network of lower nodal degree because this results in greater differences in forcer strength in the basic mesh design. Net 3 is another metropolitan interoffice network [\[BeFe98\]](#) with an average nodal degree of 3.7. The topology for Net 4 is based on that given by Lardies and Aguilar [\[LaAq98\]](#) representing a backbone fiber optic route structure between major European cities. Nets 5 and 6 are other wide-area regional topology models. [Table 11-3](#) summarizes the characteristics of each test network. The number of demand units each node pair exchanges is proportional to the product of the degrees of the two nodes and inversely proportional to the distance between the two nodes.

Table 11-3. Ring-Mesh Hybrid Test Networks

Net	# nodes	# spans	total working
1	11	23	1252
2	11	20	1035
3	15	28	1454
4	19	39	2075
5	16	29	2613
6	27	48	4242

Rather than use specific absolute cost data in these tests, we allow the relative cost of ring versus mesh transport to vary systematically, to see how the resulting hybrid designs respond. A ring-mesh relative cost parameter, F , represents the relative cost of a channel in a ring and the same function in the mesh transport environment including all nodal termination and core allocation costs in both cases. In a short haul metro environment there may be insignificant distance-related span costs, in which case the lower costs of ADM terminals compared to DCS nodes and port costs may mean the average unit hop in a ring costs perhaps only 60% of the same realization in the mesh. In other words, $F=0.6$. As span distances increase, a larger line-related cost (for regenerators and more fiber kms) becomes common to each architecture but the terminal cost for ADMs always remain lower on a per-unit bandwidth basis than DCS termination and core costs.

The test case designs used three modular capacities with relative costs including nodal termination and allocated nodal core costs as shown in [Table 11-4](#). The capacity and cost are in arbitrary units and exhibit a "four-times capacity for two-times cost" economy of scale effect (a "4x2x" model in the language of [\[DoGr00\]](#)). In addition, the ring costs for capacity are expressed in a way that reminds us that ring working capacity is always associated with a matching amount of built-in protection capacity. In other words, the seemingly higher costs for the rings above are actually for twice as much real capacity. Finally, the ring costs also reflect that although two modular units must always be purchased to access one module of working capacity, the cost is $F=0.8$ times that of acquiring and terminating the same bulk total amount of transmission in the mesh environment. The parameter, F^* thus reflects the overall extent to which it is thought that terminating capacity on ring ADMs is less expensive than doing so on cross-connects.

Table 11-4. Modular Capacities and Costs for Ring and Mesh (arb. units)

m (index)	Z^m (module capacity)	$C_j^m + T^{mesh,m}$ mesh cost for capacity	$2 \cdot (C_j^m + T^{ring,m})$ ring cost for capacity (F=0.8)
1	12	16.9	27.16
2	24	24	38.4
3	48	33.94	54.30

11.6.4 Results and Discussion

Under these capacity versus cost assumptions, results were obtained for both heuristics on the six test networks. Where possible, corresponding solutions to the optimal design model Hyb-1 were also obtained. The results are presented in [Table 11-5](#).

Table 11-5. Results with Forcer Clipping Hybrid Network Design Heuristics

Design	Net 1	Net 2	Net 3	Net 4	Net 5	Net 5
	11 nodes 23 spans	11 nodes 20 spans	15 nodes 28 spans	19 nodes 29 spans	16 nodes 29 spans	16 nodes 29 spans
Pure mesh	1877	1705	2211	3198	4606	7366
Heuristic 1	1750 30, 4 rings 7.3 min.	1504 30, 2 rings 1.7 min.	1968 50, 4 rings 1.5 hr.	2956 50, 4 rings 9.3 hr.	3497 50, 9 rings 33.3 min.	5717 50, 8 rings 3.1 hr.
Heuristic 2	1705 30, 5 rings 20.1 min.	1509 30, 2 rings 2.1 min.	1997 75, 4 rings 37.4 min.	2908 50, 5 rings 8.9 hr.	3563 50, 7 rings 25.3 min.	5890 50, 8 rings 1.4 hr.
Hyb-1 Design Model	1667 4 rings 36.9 min.	1487 3 rings 6.3 min.	2088 ^[a] 4 rings 25.3 hr.	-	-	-
LP Bound	1617	1437	1888	-	-	-

^[a] Best feasible solution with MIP gap = 10%

Each table body entry gives, in order: (1) the value of the hybrid network objective function at termination (i.e., total cost according to [Equation 11.3](#)), (2) the size of the working set used (for the heuristic runs only), (3) the total number of rings placed, and (4) the run time. An LP lower bound was also obtained from the optimal formulation with all capacity variables relaxed to real variables and this is shown for reference. Where the Hyb-1 solution method or its relaxation results are omitted, CPLEX had already run over three days on the problem without terminating or available memory was exceeded. In addition, the MIP results for Net 3 are the best feasible obtained after allowing CPLEX a MIP gap of 200 (absolute value), so that the run would terminate in roughly one day. The working set for all of the heuristic results comprised no more than 75 of the top ranked forcer clipping cycles as found by each heuristic.

The computational results show the extreme difficulty of solving the hybrid design problem to optimality but that the heuristics do fairly well relative to the cases where optimal results are found. In Nets 1 and 2, where the optimal solutions are available, the gaps are from nearly zero to only 5% at most. More importantly the hybrid designs show reductions in total network cost relative to the pure mesh reference designs, ranging from 7% to 25% under F=0.8. In a separate set of results with F=0.6, relative cost reductions up to 39% are observed in

Net 6, in a hybrid design involving 11 ring placement decisions. The heuristics typically ran to completion in under two hours, with the exception of Net 4.^[4]

^[4] Execution was on a SUN Enterprise HPC450 with four 250 MHz Ultra Sparc II processors and 800 MB RAM.

Inspection of the rings placed shows that three- to five-span 48-channel rings were the most common, especially under $F=0.8$. 24 and 12-channel rings were placed with greater frequency at $F=0.6$. The most geographically extensive ring placed was a 10-span 12-channel ring which was the last ring to pay-in as part of the design of Net 2 with heuristic 2 at $F=0.6$. We interpret formation of such an extensive but small-capacity ring, especially near the end of the design, to portray how the forcer clipping heuristic was looking hard to exploit remaining groupings of small forcers. The ring with the largest capacity-distance product in all designs was a 14-span 48-channel ring placed by heuristic 1 in Network 6 under $F=0.6$. The high frequency of 48-channel rings in the results suggests that a 2x economy of scale in capacity is a significant help to the ability to prove-in rings by going to the larger capacities when possible. Although the effect is small, the results for Net 2 compared to its "parent" Net 1, are consistent with the expectation that removing three spans would increase the ruggedness of the forcer landscape and thus increase the forcer-clipping design benefits.

Based on results here and a larger unpublished sample of results, Heuristic 2 seems to be the most advantageous. It is faster and most often closer to optimal than Heuristic 1. The explanation seems to be that, although Heuristic 1 better characterizes the presence of strong forcers, Heuristic 2 is better at identifying cycles that contain many different forcers. It was also observed that the first heuristic tends not to admit cycles that had a high proportion of small forcers, or small cycles of any type. This is due to its nature of seeking the greatest (absolute) total forcer mass in the sense that it is seeking the highest total $F \cdot x$. A large ring accumulating many weak forcers can thus rank higher than a small ring that consists entirely of moderate forcers. It is, however, desirable to consider these smaller rings of proportionately many logical forcers, as Heuristic 2 does, since they evidently can have high individual economic return factors.

A further factor in favor of Heuristic 2 is that it is more likely to admit cycles that may embody hidden co-forcer effects. By this we mean a forcer may appear to be of relatively weak forcing strength because the next latent forcer is not far below the current forcer threshold. As a pair, however, the two could be considerably higher than the next further latent forcer. In cases where such an apparently weak forcer is clipped off together with the next latent forcer for the same other spans, their combined removal can release a more than expected amount of mesh spare capacity; more than their individual forcer magnitudes would suggest. Neither of the working set selection heuristics can explicitly detect such effects, but once ring candidates that would have this effect are present in the working set, the detailed SCA redesign of the residual mesh definitely detects and capitalizes on such co-forcer clipping effects. An option in a production system is of course to simply pool the working sets from both of these (and/or any other) heuristics, thus ensuring (possibly in return for a greater run time) that the best opportunities from either viewpoint should be captured.

Given that solutions to Hyb-1 (where obtainable) are optimal and embody no *a priori* notions such as forcer-clipping, they provide a scientific test of the forcer-clipping hypothesis. In all cases where the optimal results are available, post-design forcer analysis of the mesh residual does confirm forcer-leveling effects, but they are smaller than anticipated. And yet the heuristics, working on forcer-clipping do well. This was initially hard to explain. One reason that the forcer leveling effect is not even *more* evident is that if a ring is used in the solution, then it is used not *only* to remove forcer capacity from the mesh below, but is also used to scoop up the maximum w_i quantity out of the underlying mesh on each of its spans. Thus, the placed rings actually *remove more than just forcer capacity* from the mesh. This improves the economic benefit, but has the side effect of recreating a still fairly rough forcer landscape in the mesh residual, thus making it hard to purely isolate and observe only the forcer clipping effect of the ring. The mesh residual network would be expected to be much more leveled, in a forcer sense, if the rings were strictly used only to shave off forcer peaks. But this would be foregoing the maximum cost benefit of the ring.

Another validation exercise was to force the heuristics to place one more ring than they would otherwise have done. The heuristics normally stop when no further ring can be found that has a net payback. We wanted to validate that the total network cost did not just keep going down the more mesh capacity was replaced by rings. After all, from a purely "bulk capacity" view, rings are given the advantage of being 20% to 40% cheaper than the same capacity realized in the mesh (i.e., $F=0.8$ means that the ring—including its protection capacity—costs 20% less than the same total capacity in the mesh). So one can ask: why does the outcome not just slide to an all-ring solution? The reason (as this test confirmed) is that the bulk cost of capacity is only part of the issue: To use a ring on desired spans, one must somehow close the ring over other spans, and pay for those spans, too. Thus, a ring can only be effective if circumstances coincide fairly well with the cyclical closure of the ring in some way over network graph as a whole. If there are two spans that would do well with an overlying ring, but the topology requires another six spans just to form a ring, then its overall economics are hampered. At the same time as the mesh residual becomes more forcer-leveled through ring placement, its efficiency goes up until some reasonably leveled forcer structure attains most to the achievable benefit in the mesh. Thereafter, rings are less able to pay for themselves by enhancing the mesh residual efficiency. Thus, there are two architectural phenomena working in the direction of retaining a certain mesh residual even though ring transport is unconditionally cheaper on a point-to-point bulk capacity basis.

In the experiments that compel the heuristics to go on placing the next best ring, even after the normal stopping point the total cost did go *up* as a result in all cases. Total design cost increased as ring placements went on past the normal stopping point. This confirms that there really is a cross-architectural trade-off happening in the ring-mesh hybrids. It also confirms how and why the heuristic works: As the first

rings are added, they are well loaded, they have good forcer clipping effects that reduce the residual mesh sparing, and they are on cycles that can group together several forcers into one clipping ring. But as the residual mesh gets an increasingly leveled forcer landscape, the amount of working capacity that is better off left in the mesh goes up, the strength of remaining forcers is weakened, and it becomes harder to find a ring that can efficiently group together numbers of forcers to fill itself. Part of the effect also is that the placed rings

consume all or part of the original W_i quantities thus reducing the achievable fill in subsequent rings. The understanding of these architectural interactions is sufficient to say that in general a hybrid construction could be lower in cost than either a pure mesh or pure ring network over a fairly wide range of relative costs.

[\[Team LiB \]](#)

◀ PREVIOUS NEXT ▶

11.7 Ring to Mesh Evolution via "Ring Mining"

While still on a theme of ring-mesh hybridization, let us now consider a kind of ring-mesh hybrid strategy in which the aim is to achieve the most cost-effective evolution from a ring-based status quo to a future mesh-based network.^[5] In the 1990s many network operators deployed extensive ring-based networks, reaching a culmination with OC-192 SONET BLSR rings in regional and long haul networks and extensively deployed OC-48 SONET BLSR and UPSR rings in metro areas, as well as 1:1 DP APS systems^[6]. In that era rings were the most widely standardized and vendor-supported solutions for practical deployment. By 2001, DWDM long haul rings (OSPRs) and metro optical ring analogs to UPSRs (OPPRs) were also beginning to be deployed. About the same time, however, standardization efforts to support mesh-based protection (by the ITU, IETF and OIF) began, and most operators quickly welcomed the practical options this would provide them to consider the mesh alternative. Thus, many operators see "mesh as the way to go in future" but currently operate extensive ring-based networks. So the problem is one of "how do we get there from here?"

^[5] The "ring mining" idea was initially undertaken as a class project in the author's graduate course. This was further developed and presented at [\[CIGr01\]](#). In collaboration with TELUS, ring mining has since been extended to consider p -cycles as the target architecture [\[KoGr03\]](#).

^[6] Although we don't specifically mention APS systems again in our treatment of ring-mining, 1:1 or 1+1 DP APS are equivalent to 2-node ring systems and it is implicit that they can be integrated into the same theory and methods—and related planning tools—for this category of ring-to-mesh evolution planning.

One option is to simply cap the existing rings and grow an all-new mesh network as a separate overlay. Eventually, under continued growth, the network will be almost all mesh-based and demands served in the residual rings could be rolled into the mesh at a suitable date in the future. We take this as the conventional reference strategy for a ring to mesh evolution. But a more interesting and possibly profitable strategy for operators to consider is the idea of "ring mining." The basic idea is conveyed by the question we first asked in [\[CIGr01\]](#): To what extent can an existing ring-based network "soak up" ongoing demand growth, simply by providing access to its raw span capacities and converting the mode of operation to a restorable mesh architecture? As stated this initial question implies pure ring mining where there would be *no* new capacity added at all while nonetheless sustaining ongoing growth solely by conversion to a mesh-restorable operation under the span capacities represented by the rings. The reason a network operator would be interested in considering this approach is the prospect of continuing service growth without adding new transmission equipment for a significant period. The strategy could be financially attractive if it meant an operator could serve growth, for possibly a year or more, while capping or deferring major capital additions for transport capacity.

The "pure" ring mining question above gives the basic idea but it is somewhat idealized because in practice, we expect there would be some kind of costs associated with converting ring ADMs for access by mesh-based cross-connects, changes in network element software, and network management changes. In addition, pure ring mining may not be the most advantageous approach for a practical evolution strategy. If relatively small new capacity additions were allowed, in conjunction with ring mining, there may be even greater potential to unlock stranded ring working capacity and access ring protection capacities during ring-to-mesh evolution. Accordingly, let us now look at theory and test case results to assess the potential extent and benefits of ring mining by studying the following three approaches to the idea:

- **Q1.** Without adding any new capacity, what is the largest scalar multiplier λ on the demand matrix that can be served assuming a span-restorable mesh that uses no more capacity on any span that is embodied in the initial ring set?
- **Q2.** If a total growth g is forecast to some date in the future, what is the minimum total capacity investment needed to meet this forecast using a combination of ring mining and mesh capacity additions?
- **Q3.** What is the minimum total cost strategy for growth to meet a future demand multiplier λ of involving ring-mining, selective new capacity additions, and allowing for selective conversions or reuse of ADMs with a cost for each node so adapted?

We recognize that from a network management standpoint it would probably be simplest to "cap the rings" and serve all new demand growth with a mesh overlay. This would be the most cost-effective strategy if operational costs and ring mining nodal conversion were high

enough to outweigh the ring mining advantages. We cannot assess such costs, however. What follows is therefore limited to addressing the *benefit* side of the cost/benefit question that is involved in adopting a ring mining evolution strategy. Individual network operators are in a better position to assess the cost side of accessing this opportunity given their own ring deployments and operational and planning capabilities. However, given the magnitude of the cost-savings and expense deferral we see in some cases, it seems possible that an operator who could manage the strategy would access a significant financial advantage over competitors during their period of ring mining.

11.7.1 Optimization Model for Pure Ring Mining

The potential for pure ring mining as a strategy to serve growth while evolving from ring to mesh operation can be studied with the ILP formulation below. This formulation determines the highest uniform multiplier l that can be applied to every element of the demand matrix while keeping the demand both routable and 100% mesh restorable under span restoration. This is a type of joint working and spare capacity optimization, as in the sense of JCA in [Chapter 5](#), but it is put into effect under span capacity limits set by the "broken up" rings. The main use of this formulation is to give a broad indication of the basic potential for ring mining.

- D = Set of O-D pairs or relations with non-zero demands.
- S = Set of spans.
- P_i = Set of eligible restoration routes for span i .
- Q^r = Set of eligible working routes for r^{th} O-D pair.
- U_j^m = Number of rings of module size m on span j in the ring design.
- d^r = Initial demand served by the ring network for the r^{th} O-D pair.
- l = A maximum uniform growth multiplier (to be solved for).
- $g^{r,q}$ = Working capacity required on the q^{th} eligible route to serve demand on relation r .
- f_i^p = Restoration flow through the p^{th} route for failure of span i (variable).
- $\delta_{i,j}^p = 1$ if the p^{th} restoration route for span i uses span j , 0 otherwise.
- $\zeta_j^{r,q} = 1$ if the q^{th} working route for the r^{th} O-D pair uses span j , 0 otherwise.
- M = A set of modular ring capacity sizes.
- Z^m = Number of working capacity units for the m^{th} module size.
- w_j = Mesh logical working capacity needed on span j to serve demands.
- s_j = Mesh logical spare capacity needed on span j to support restoration.

RM-1:

Maximize:

Equation 11.10

λ

Subject to:

1. The scaled-up demand matrix is fully routed:

Equation 11.11

$$\sum_{q \in Q} g^{r,q} = \lambda \cdot d^r \quad \forall r \in D$$

2. The mesh logical working capacity supports the routing of all scaled-up demands:

Equation 11.12

$$\sum_{r \in D} \sum_{q \in Q} g^{r,q} \cdot \zeta_j^{r,q} \leq w_j \quad \forall j \in S$$

3. Total of mesh restoration flows protect all mesh working span capacities:

Equation 11.13

$$\sum_{p \in P^i} f_i^p = w_i \quad \forall i \in S$$

4. Mesh restoration flows respect spare capacity on all spans:

Equation 11.14

$$\sum_{p \in P^i} f_i^p \cdot \delta_{i,j}^p \leq s_j \quad \forall i, j \in S^2$$

5. Total mesh working and spare logical capacity does not exceed preexisting total capacity of the modular rings overlying each span:

Equation 11.15

$$w_l + s_l \leq \sum_{m \in M} t_j^m \cdot 2 \cdot Z^m \quad \forall j \in S$$

[Equation 11.11](#) scales the demand served to be l times the original demand. In practice, the same growth multiplier will not necessarily arise for every individual demand pair, but for characterization of this strategy, finding the largest common demand multiplier that can be sustained gives a relative characterization of the potential for ring mining. [Equation 11.12](#) ensures that there is enough working capacity in the network to support the routing of all the demands. [Equation 11.13](#) ensures that the sum of restoration flows for each single-span cut is equal to the working capacity to be restored. [Equation 11.14](#) ensures that there is enough spare capacity on each span to support all the restoration flows that cross it in every span failure case. [Equation 11.15](#) ensures that the sum of working and spare capacity on each span in the logical mesh does not exceed the amount of capacity on that span as provided by the initial ring set, including the protection capacity of such rings. Z^m is defined here in the same way that it was for the ring-mesh hybrids in terms of the normally usable working capacity provided by a ring. Thus, "mining" a ring of the m^{th} module size provides $2 \cdot Z^m$ of capacity for use in a mesh.

11.7.2 Ring Network Designs for Tests of Ring Mining

For quantitative studies of the ring mining idea we used a supply of highly optimized pure ring network designs produced in doctoral studies of ring network design methods [\[Mor10\]](#). These research-based designs are probably more efficient and well-loaded than most real networks, which tends to work against the ring mining case. All the initial ring network designs fully serve the demand matrix for which they were designed and the same demand matrices are used as the starting point for ring mining studies. Six of the ring networks were produced by a "fixed charge and routing IP" (*fcrip*) formulation in which modular rings are selected jointly with the demand routing and ring loading decisions. These designs are capable of extremely high ring loading efficiencies and the benefits of span elimination [\[LeGr99\]\[Lee99\]](#). Another six designs are from a "span coverage IP" (*scip*) formulation in which demands are first shortest-path routed over the graph and then a minimum cost modular capacitated ring cover is found. *scip* designs are generally less efficient than *fcrip* and do not exhibit span eliminations. A final five designs were produced by a *Tabu Search* (TS) option embodied in RingBuilderTM [\[7\]](#) which is a ring network design system [\[MoGr01\]](#). The TS procedure calls RingBuilder for a starting design, and for subsequent search diversifying restarts. Between restarts it generates ring drop and ring add moves, with revised demand routing at each move, seeking a lower cost design that still serves all demands. The TS procedure can lead to some of the highest-efficiency designs possible for ring-based networks—packing the demands to be served into the fewest possible ring systems. Each method is described further in [\[MoGr99\]](#) or [\[Mor10\]](#).

^[7] RingBuilderTM is a TRILabs-developed design and optimization package for ring-based transport networks, licensed to VPIsystems (New Jersey) for further commercial development and distribution.

There are three different test network graphs for each design method and all designs for the same topology serve the same demand matrix. [Table 11-6](#) summarizes these and other attributes of the ring mining test cases. The designs on the (32,45) topologies were synthesized with OC-48 and OC-192 module options. The (15,28) and (20,31) test cases were synthesized with OC-12 and OC-48 module options. Module costs followed a moderate economy of scale model based on industry data (~3x2x). As evidenced in [Table 11-6](#) the resultant designs adopted a single most-economic module size so that for our tests, these all turn out to be single-modularity ring network designs. This is an effect arising from design for minimum cost, not necessarily minimum capacity. This is an effect that tends to favor ring mining. It is nonetheless consistent with the nature of real ring-based network designs that would be candidates for ring mining. On the other hand, the high efficiencies of these test-case designs works against ring mining—limiting the ultimate potential to support further growth because the initial capacity is well used to begin with.

Table 11-6. Properties of the Ring Networks for Testing the Ring Mining Strategies

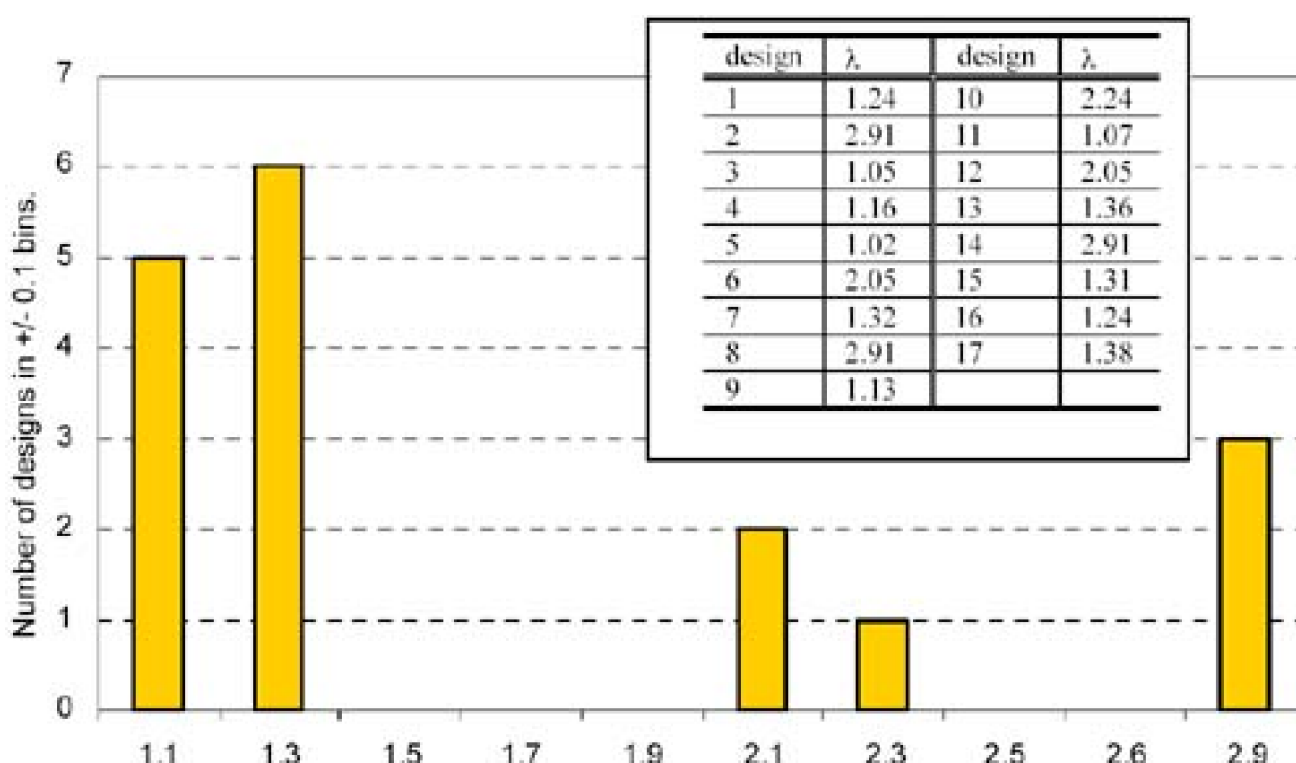
no.	(Nodes, spans)	# Rings	Design Method	Module Sizes	Total Capacity	# Span eliminations
1	15, 28	6	fcrip	OC-12	960	3
2	15, 28	5	fcrip	OC-48	2976	3
3	20, 31	14	fcrip	OC-12	1728	0
4	20, 31	6	fcrip	OC-48	2880	6
5	32, 45	12	fcrip	OC-48	8448	1

no.	(Nodes, spans)	# Rings	Design Method	Module Sizes	Total Capacity	# Span eliminations
6	32, 45	8	fcrip	OC-192	21504	3
7	15, 28	9	scip	OC-12	1200	0
8	15, 28	5	scip	OC-48	3264	0
9	20, 31	18	scip	OC-12	2136	0
10	20, 31	7	scip	OC-48	3840	0
11	32, 45	12	scip	OC-48	8544	0
12	32, 45	7	scip	OC-192	18048	0
13	15, 28	6	TS	OC-12	1944	3
14	15, 28	5	TS	OC-48	4032	2
15	20, 31	10	TS	OC-12	2232	0
16	32, 45	7	TS	OC-48	7296	3
17	32, 45	6	TS	OC-192	21504	5

11.7.3 Test Results for Pure Ring Mining

Figure 11-17 summarizes the results of solving RM-1 on the 17 test case ring networks above. The results show a uniform growth potential on the entire demand pattern that ranges from 10% to as much as a tripling in demand served. Over a third of the test cases could sustain a doubling in demand just by ring-to-mesh conversion.

Figure 11-17. Distribution of the maximum sustainable growth factor in pure ring mining.



λ. maximum sustainable growth multiplier

Why Ring Mining Works

No simple generalization seems warranted as to which designs will yield the greatest I in this pure ring mining sense. Not surprisingly the greater growth multipliers tend to arise in ring networks using the largest ring modular capacities but this is not always the case. One of the OC-48 designs shows only 1.07 sustainable growth factor while two other OC-48 designs are up at $I \approx 2.9$ times. Rather, the potential of the ring mining strategy seems to depend on the details of each network. But detailed inspection of the results does largely explain why such high multiples of additional demand growth are sustainable in some cases. Basically four forms of new efficiencies are being tapped when the architecture is converted to mesh. Although it is hard to quantitatively separate these effects for strict ranking of importance, our observations suggest that the following effects are all at work, in approximate order of importance:

1. Ring protection capacity is reclaimed for general use as mesh working and spare capacity. This is the greatest source of new "liquid" capacity because it is always a 100% set-aside matched to the working channel capacity available for provisioning of the prior rings.
2. Ring redundancy is reduced to mesh redundancy. There is greater spare capacity sharing in the mesh so the bulk fraction of overall capacity used for protection is lowered.
3. New (growth) demands in the mesh follow shortest-path routes over the facilities graph, not ring-constrained routes.
4. Ring "stranded capacity" is freed. The greater flexibility of the mesh in terms of both routing and the ability to decide flexibly in design, whether any given channel will be designated as working or spare, contributes to a greater ability to use all available capacity without working channels getting cut off from paths that could usefully lead up to and away from them for provisioning use.

The same detailed study of the ring mining solutions also lead us to hypothesize that some relatively small additions of new capacity might act like a catalyst to "unlock" significantly more of the ring capacity present. This is the key idea of the next strategy.

11.7.4 Ring Mining with Minimum Cost Capacity Additions

As explained, the pure ring mining growth potential is an indication of how much more demand can be served as a mesh, using only the capacity from an existing ring network design. Let us now consider scenarios in which we allow strategic additions of new capacity, in conjunction with ring mining. This is compared to an alternative strategy for ring to mesh evolution that consists of capping the existing ring network and building an entirely new mesh overlay on top of it to serve all new demands. For comparison to the present strategy, the "cap and grow" overlay strategy will be designed with the joint modular working and spare capacity optimization method (MJCA) from [Chapter 5](#).

The formulation RM-2 determines the minimum amount of added capacity that is needed, on top of available ring mining capacity, to sustain a growth of the demand matrix that is higher than the pure ring mining potential calculated with RM-1. With RM-2, the uniform demand multiplier I becomes an *input* to the problem and the number of mesh capacity modules to add to the network are new variables. The new parameters and variables that come into play are as follows, all others are as in RM-1:

- I = The total demand growth multiplier to be satisfied (a parameter).
- C_j^m = Cost of a module of the m^{th} capacity size if installed on span j (a parameter).
- N_j^m = Number of modules of the m^{th} size added on span j (integer variable).

RM-2:

Minimize:

Equation 11.16

$$\sum_{j \in S} \sum_{m \in M} n_j^m \cdot C_j^m$$

Subject to:

1. [Equation 11.11](#) through [Equation 11.14](#) inclusive, plus:
2. Total mesh working and spare logical capacity does not exceed preexisting total capacity of modular rings overlying each span plus new "mesh capacity" modular additions:

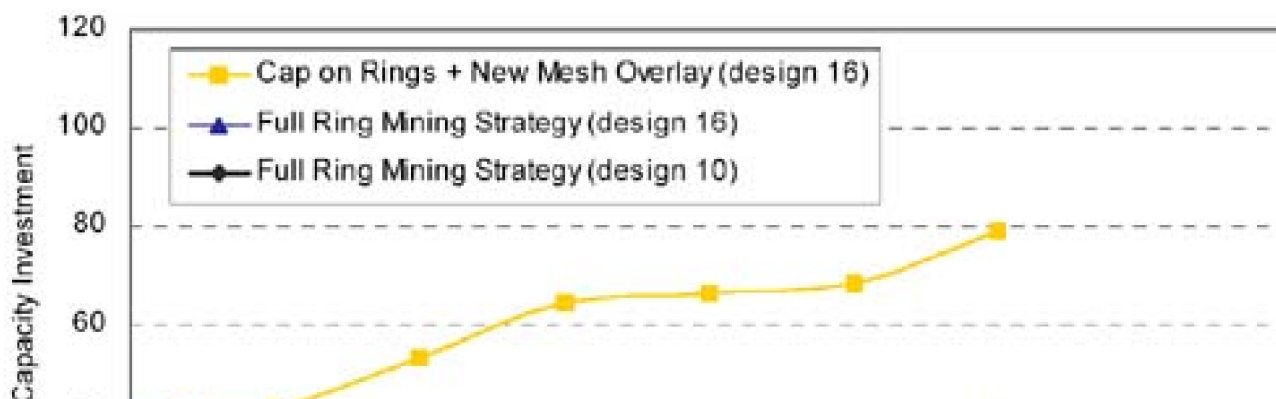
Equation 11.17

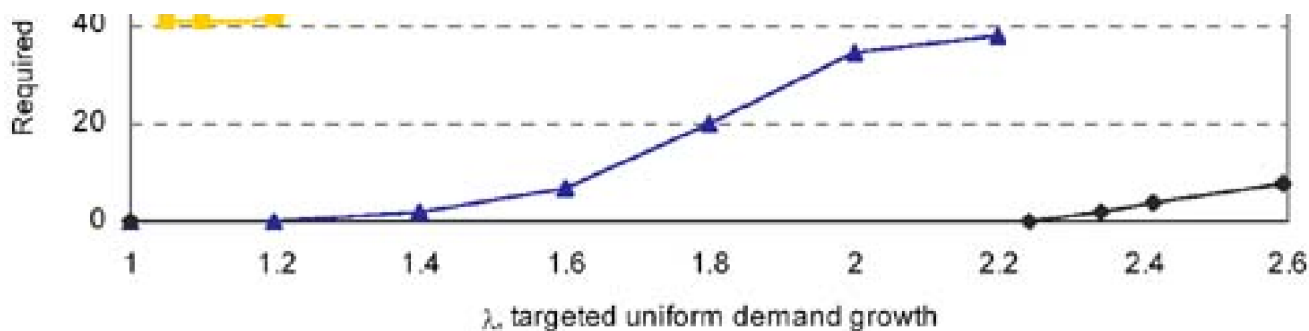
$$w_i + s_i \leq \sum_{m \in M} (2 \cdot t_j^m + n_j^m) \cdot Z^m \quad \forall j \in S$$

In [Equation 11.17](#), the available capacity on each span is now the sum of the capacity reclaimed from the ring design and any added mesh capacity that has to be paid for directly. The formulation minimizes the total capacity investment needed to meet a demand that is l times the original demand served by the ring design.

[Figure 11-18](#) shows results for two ring mining capacity profiles and the cap + grow strategy for one of them. The uppermost curve is for the "cap and grow" mesh capacity expenditure profile for ring network design 16. The lower design 16 curve shows the reduced and delayed expenditure profile required to meet the growth with ring mining. The curve for ring design 16 shows that compared to pure mesh growth for new demands, the overall investment profile to meet the next 220% growth is 50% lower with ring mining and most expenditure is deferred until after the first 40% of added growth. Design 16 was chosen because it is one of the cases showing a small potential for pure ring mining ($l = 1.24$) in [Figure 11-17](#) and is therefore a more challenging case. This ring design serves 354 demand units on 72 O-D pairs. The mesh growth model uses OC-48 and OC-192 module sizes with relative costs of 1 and 2, respectively. In comparison, Ring design 10, in the middle group on [Figure 11-17](#), shows a complete deferral of capacity growth costs until over 200% growth on the initial demand. The obviously desirable characteristics of deferral and reduction of new capacity investment are apparent relative to the baseline of capping the rings and putting all growth on mesh.

Figure 11-18. Serving growth with minimum cost capacity additions and ring mining.





11.7.5 Minimum Cost Evolutionary Strategy with Conversion Costs

We now extend the ring mining framework to provide a complete model for transitional growth planning to get from an existing ring set and demand matrix to a future demand growth multiplier of λ at minimum total cost. Here we allow a cost for ADM node *conversion* and we assume a small but non-zero cost (for example, for network management software changes) to permit *reuse* of an ADM as a chain element in the resulting logical mesh design. By conversion we mean any process that converts the ring line capacity into a form accessible by a co-located DCS (or OXC) that takes over operation of the same capacity in the mesh. Conversion could involve simple removal of the ADM, or the strategy above of using the ADM's add/drop and extra traffic capabilities to access its line capacity. Under reuse we consider leaving an existing ADM in place—operating as an ADM in a chain subnetwork of the new mesh.

For comparative results we also consider a reference strategy of capping the ring network and serving all new demands in a separate mesh network overlay. This ring mining model is more realistic in terms of considering the economic factors involved and can exactly specify at which nodes to break into the rings, where to add new capacity, which ADMs to reuse, and (at least in principle) which segments of ring capacity to actually abandon to avoid conversion costs if the overlying mesh can more efficiently carry the relevant demands. The latter may produce candidates for salvage value but we have not assumed any such benefit to the ring mining strategy. The general model allows each ADM to have a different conversion cost but we assume a generic conversion cost of c per ADM converted. $c = 1$ is defined as conversion costs that equal the cost of adding one line-rate unit of transmission capacity on the average length ring span. Only ADMs located in a geographic site of degree 3 or higher in the basic facilities topology graph are considered for conversion. All other ADMs are considered for simple reuse as degree 2 network elements for the mesh. ADMs have a *reuse* cost of $c/10$. The new parameters and variables at this stage are:

- R = The set of rings in the ring design at the start of the transition.
- x_l^m = Number of modules of the m^{th} size on ring l .
- A = Set of ADMs from the ring design, index k .
- E_k = Conversion cost for ADM k . Re-use cost is expressed as a factor $c \cdot E_k$; $c < 1$.
- $g_{j,l} = 1$ if span j is covered by ring l , 0 otherwise.
- $n_{k,l} = 1$ if ADM k is on ring l , 0 otherwise.
- $b_{j,k} = 1$ if span j is adjacent to ADM k , 0 otherwise.
- $d_{j,l} = 1$ if decision is made to use capacity from ring l on span j , 0 otherwise (a variable).
- $r_k = 1$ if decision is made to convert ADM k , 0 otherwise (variable).

RM-3:

Minimize:

Equation 11.18

$$\left\{ \sum_{j \in S} \sum_{m \in M} n_j^m \cdot C_j^m + \sum_{k \in A} \rho_k \cdot E_k \right\}$$

Subject to:

1. [Equation 11.11](#) through [Equation 11.14](#) inclusive, plus:
2. Total mesh working and spare logical capacity does not exceed preexisting total capacity of modular rings *that it has been decided to use through ADM node conversion, plus new "mesh capacity" modular additions:*

Equation 11.19

$$w_j + s_j \leq \left(\sum_{l \in R} \sum_{m \in M} \mu_{j,l} \cdot x_l^m + \sum_{m \in M} n_j^m \cdot Z^m \right) \quad \forall j \in S$$

3. Can only access capacity of ring l on spans it overlies:

Equation 11.20

$$\mu_{j,l} \leq \gamma_{j,l} \quad \forall j \in S \quad \forall l \in R$$

4. Can only access capacity of ring l if ADM conversion decision (and cost) is taken:

Equation 11.21

$$\rho_k \geq \mu_{j,l} \cdot \beta_{j,k} \cdot n_{k,l} \quad \forall k \in A \quad \forall l \in R \quad \forall j \in S$$

The first term in the objective function represents the cost for adding new capacity modules. The second term represents the cost of converting or reusing ADMs. [Equation 11.19](#) ensures that the sum of mesh working and spare capacity on each span does not exceed the amount of available capacity on that span. The available capacity in this case is the sum of the capacity reclaimed from some rings covering that span plus the sum of newly added modules. [Equation 11.20](#) ensures that the capacity from a ring is not reclaimed on a span that is not covered by that ring. Finally, [Equation 11.21](#) forces any ADM to be converted (if at a degree 3+ site) or reused (if at a degree 2 site) if the capacity of the ring it belongs to is reclaimed on one of its two adjacent spans.

[Figure 11-19](#) shows the time profile of expenditures to meet growth out to $I = 2.2$ with this strategy starting with ring design 16. Even with significant costs for ADM conversion, the ring mining strategy is less costly than the reference model of "cap and grow new mesh." Note that as expected when $E_k = 0$ the total strategy cost is equal to the pure capacity investment profile in [Figure 11-18](#), [Figure 11-20](#) and [Figure 11-21](#) show the effect of the conversion cost on the total evolution cost to a growth factor of 2. In [Figure 11-20](#) one can see that above $E_k = 0.8$ the cost of the ring mining strategy exceeds the baseline cap and grow strategy due to ADM conversion costs. Referring back to [Figure 11-18](#), $E_k = 0.8$ would be the value for which the curve for the selective mining and the one for "Cap on Rings + New Mesh" coincide at $I = 2$. Since the curves on [Figure 11-18](#) are roughly parallel, the conversion cost for which the minimum cost strategy changes will not depend greatly on the I considered in the decision.

Figure 11-19. Comparison of ring-mining costs with ADM conversion (and reuse).

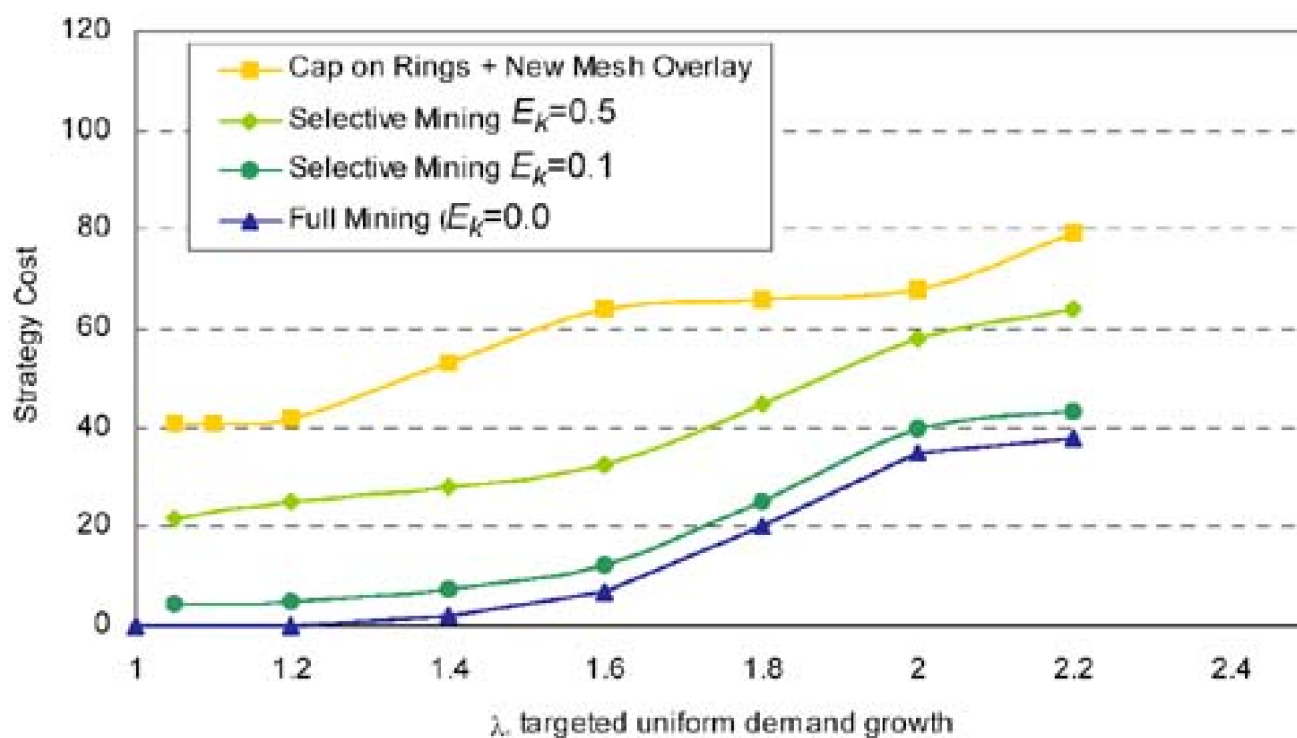


Figure 11-20. Effect of ADM conversion cost on total transition cost to $l=2$.

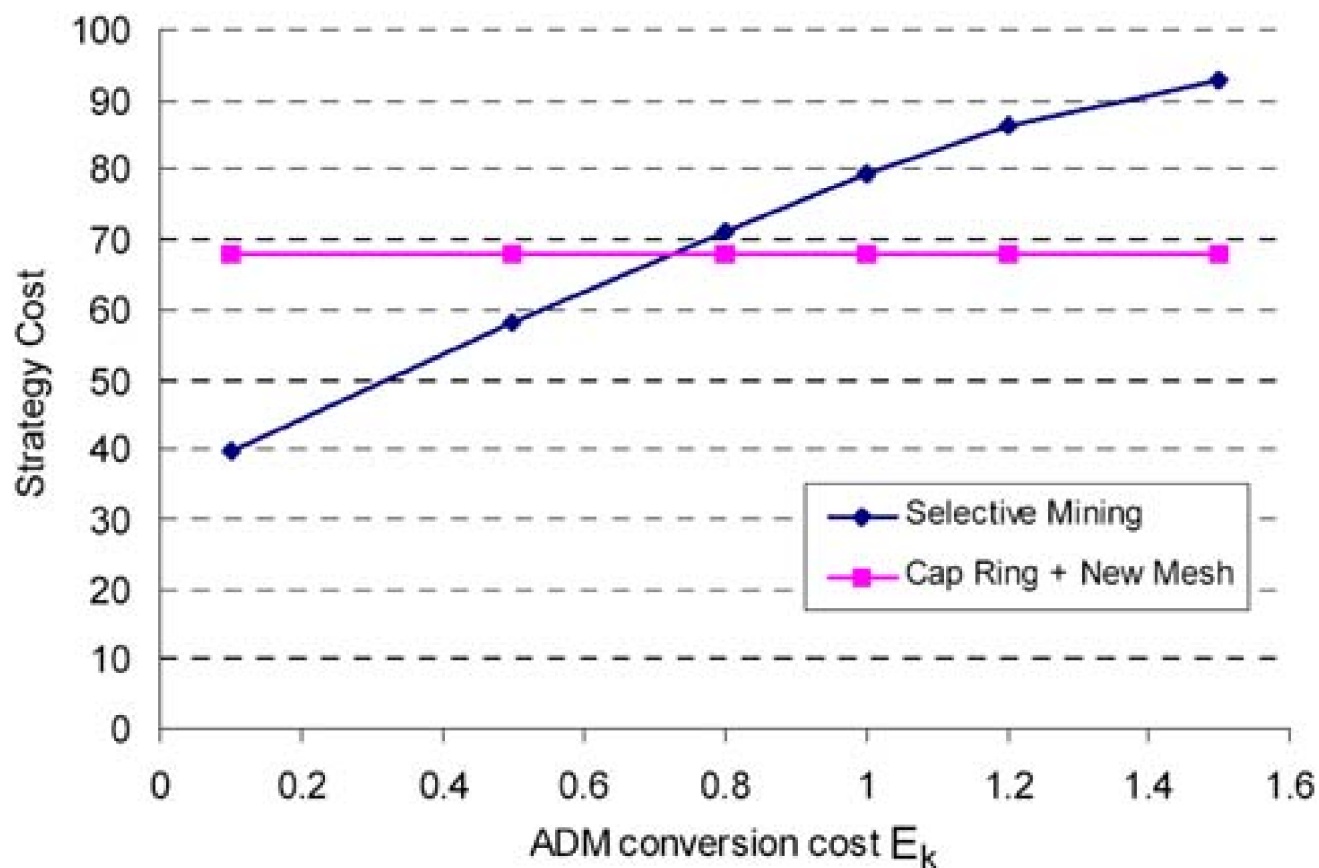


Figure 11-21. Effect of ADM reuse cost on the extent of ADM conversions for $l=2$.

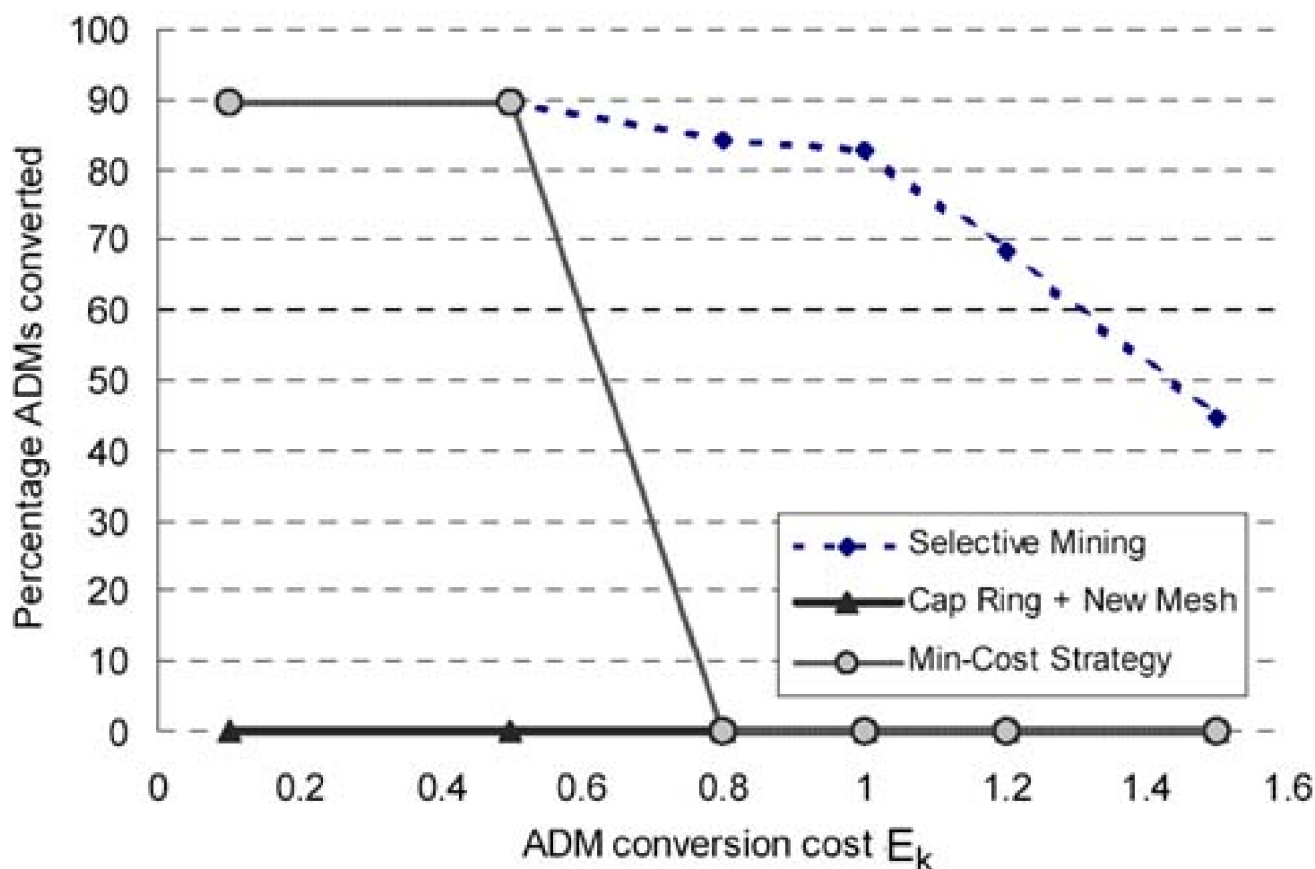
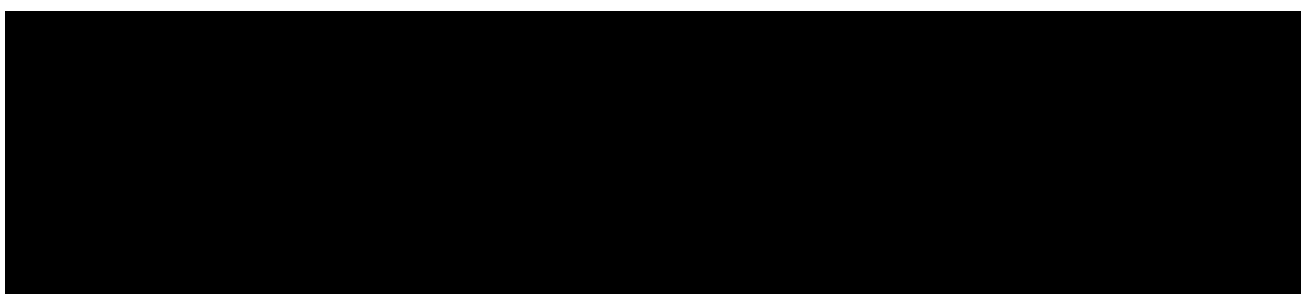
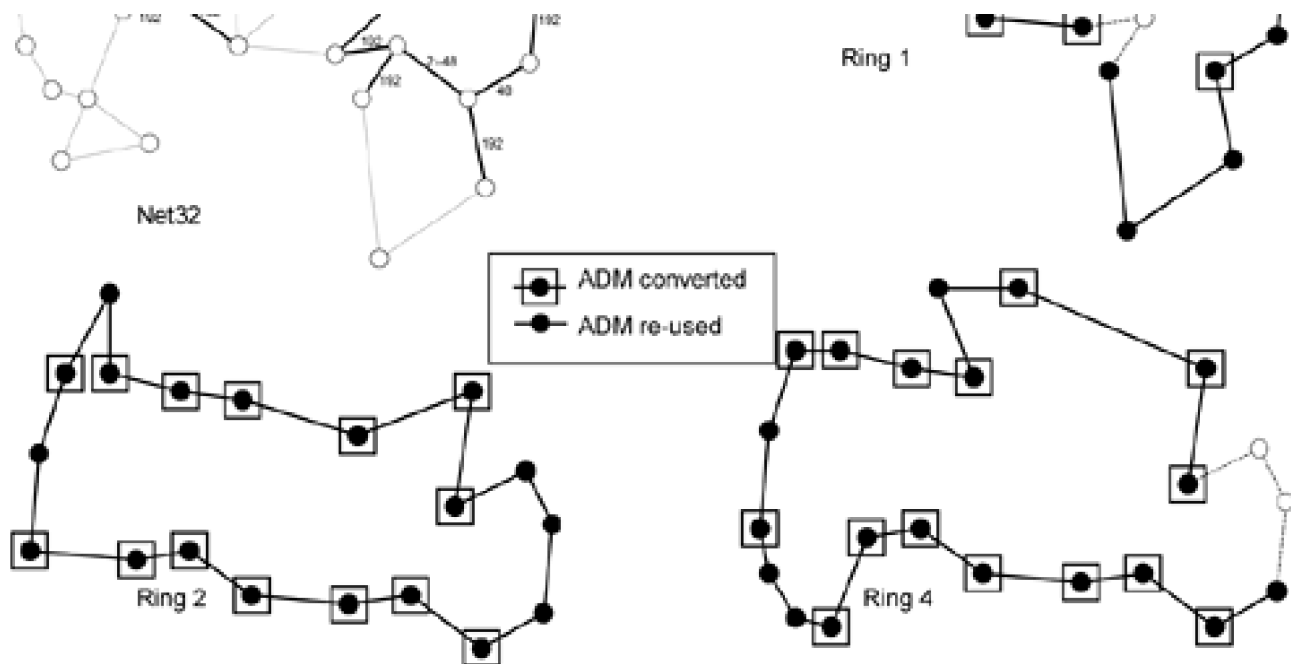


Figure 11-21 shows the percentage of eligible ADMs converted for each strategy depending on E_k . It is interesting to notice that even when E_k is low, the selective ring mining strategy only selects 90% of the ADMs for conversion. This shows an advantage of the third formulation in that it is able to identify the ADMs that are not worth converting or reusing. Figure 11-21 also shows that the pure minimum cost strategy jumps directly from 90% conversion to 0% conversion at a critical conversion cost between $E_k=0.6$ and $E_k=0.8$ in this network. At this critical point, even before the selective ring mining strategy starts reducing the number of converted ADMs, the reference strategy is more economical.

Figure 11-22 shows part of the detailed ring mining solution in the 32-node test network of ring design 16 assuming an ADM conversion cost $c=0.5$ and reuse cost of 0.05 growing out to a uniform doubling of demand. The facilities graph topology is shown in the upper left where added mesh capacity onlays are indicated in bold, annotated with the added module size. Other panels give isolated views of the three largest rings indicating which ADMs were converted for mesh access to their ring capacity and which ADMs are reused in the ring mining solution. Note that as expected all the conversions are at geographic sites with degree of 3 or more. It is at these sites that accessed ring capacity is being cross-connected for mesh routing and restoration efficiencies. Reused ADMs are those that play a cost-effective role in a chain of the resulting logical mesh. In this example 89% of all the eligible ADMs are converted and 92% of ADMs at degree 2 sites are reused.

Figure 11-22. Selective mining strategy result for the three largest rings in ring design 16.





11.7.6 Implementation of Ring Mining Strategies

Ring Types

One concern that seems reasonable at first is to assume that ring mining must depend rather intricately on the types of ring involved in the initial network. By type, we mean the logical protocol on which the ring is working, e.g., BLSR, UPSR, MS-SPRing, SNCP ring. However, it is really only the transmission line capacity of the ring, the fraction of the line capacity that is accessible for add/ drop at the ADMs, and the "extra traffic" accessibility of the protection channel (or redundant working tributary copies in a UPSR) that matters to ring mining. Although it may be desirable to keep certain ADMs in place in the transitional hybrid, the point is best made by considering simple removal of all ADMs. Clearly what is then left behind, regardless of any and all ring details, is a matching pair of point-to-point line transmission systems at the prior ring line-rate. The general point is that all rings are structurally 100% redundant regardless of their type. Every unit of useful working capacity is matched by an equal unit of protection or redundant working path capacity on the same span of the ring.

In the case of BLSR or MS-SPRings their line loopback restoration mechanism makes them especially amenable to ring-mining because at degree-2 locations they can remain in place functioning as part of a new mesh-restorable network. This is because the loopback reaction of shared protection ring ADMs retained within chain subnetworks of a mesh is identical to the reaction within the same chain of either a span or path-restorable network with respect to intra-chain flows.

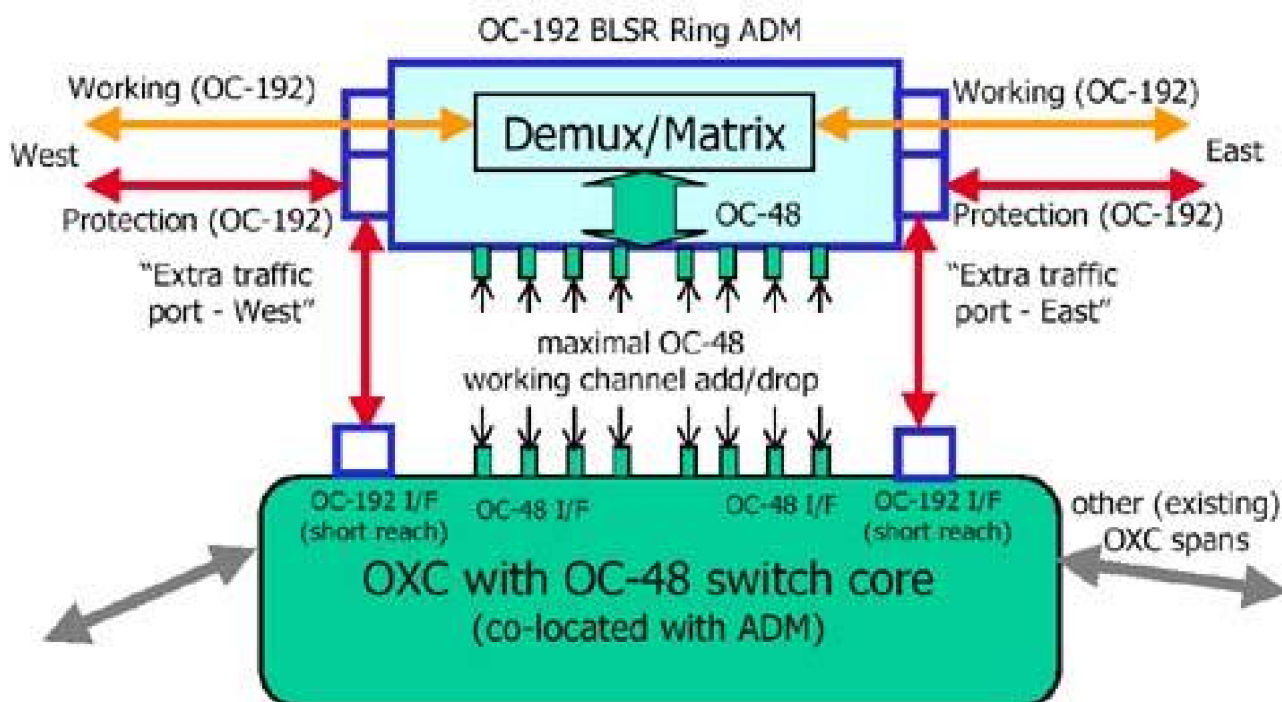
Accessing Ring Capacity for Ring Mining

Mesh-oriented access to the capacity of a ring may be through ADM nodes of the ring that have been kept in place and "converted" for ring mining or simply removed for salvage. There may be many different technical means of arranging such access but from a ring mining planning standpoint, the important characteristics are the total amount of ring capacity that can be made accessible to a co-located cross-connect, and the corresponding cost of making this capacity accessible. [Figure 11-23](#) is one example of how the entire working and protection line capacity of a BLSR-type ring might be accessed. The ADM is kept in place but converted effectively into a pair of back-to-back lightwave terminating equipment (LTE) elements. The protection channel is accessed directly at the line rate through the extra traffic feature of a BLSR ADM. The working capacity is accessed by programming the ADM into a 100% add/drop configuration and

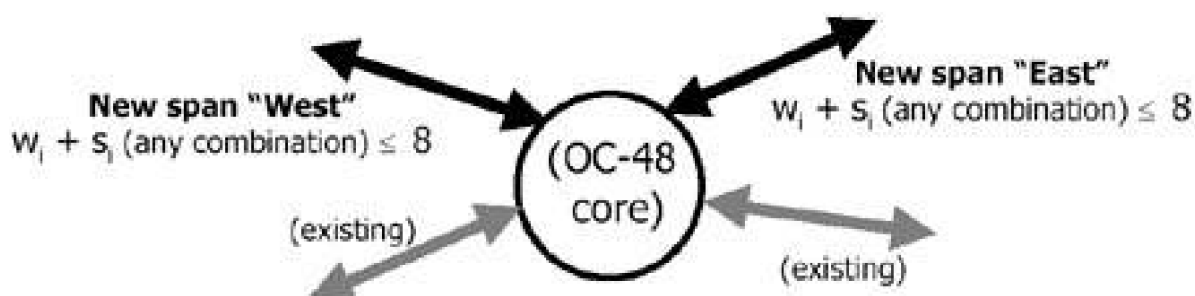
"freezing" it that way. In the example an OC-192 ADM is reused in this way providing two new spans of 8 OC-48 channels each for mesh-based. The interfaces between ADM and OXC for the OC-48 add/drops and the OC-192 extra traffic access are only short-reach (i.e., "cross-office") optical link implementations at these rates, which cost significantly less than the ADMs four existing OC-192 line-rate long-haul interfaces, which are re-used in this configuration. If the original ADM supports 100% add/drop, then the co-located cross-connects can access a total capacity that is twice the line-rate of the ring, on each of two logical spans from the mesh node. More generally if the ADM supports less than full add/drop, the converted ADM yields $(1+x)$ times the rings line-rate of accessible capacity where x is the maximum add/drop capability of the initial ADMs.

Figure 11-23. Example converting an OC-192 BLSR ADM to two new 8 x OC-48 mesh spans.

(a) Existing ADM converted to act as LTE providing two new spans of 8 x OC-48



(b) Functional view from mesh cross-connect standpoint:



This is just an example, however. The basic idea is to get line-rate or tributary-level access to both spans of the ring at converted nodes including its protection (or redundant working) channels. Other models are obviously possible depending on the ADM vendor, features, and so on. The "costs" for conversion may even be negative if there is a salvage value for removing the ADM entirely and terminating the fiber line systems directly on optically interfaced cross-connects. Perhaps the ideal ADM architecture for conversion to ring mining would be one where the optical line interface functions can be left in place and the ADM function salvaged (i.e., an ADM with separate OLTE). Since there is obviously a range of circumstances that each operator may find in this regard, we treat the cost of the technical means required to access the ring capacity as a parameter in our study. Costs for a cross-connect core at each site are assumed common to all strategies given the premise of the study that a mesh future is the common goal. Cross-connect cores are therefore present whether arrived at by ring mining or through a conventional cap-and-grow approach. Costs for incremental cross-connect terminations are meant to be part of the cost model for additional mesh capacity. Costs for cross-connect termination of ring-mined capacity are an assumed part of the ADM conversion cost.

11.8 Ring Mining to p -Cycles as the Target Architecture

In [Chapter 10](#) we introduced p -cycles. And now we are arguing strategy for network evolution from rings toward a span-restorable mesh. The reader might rightly have already put the two concepts together and said, "Wouldn't it be even more natural to somehow convert rings directly into p -cycles?" This a topic of research in the author's group that we now outline briefly. A more comprehensive discussion can be found in [\[KoGr03\]](#).

To motivate the idea, it is obvious that p -cycles and rings are topologically similar structures. They really only differ in two important ways: (1) p -cycles protect straddling spans, (2) p -cycles are spare-capacity-only structures (whereas rings define working and spare channels locked structurally together). The ADM-like p -cycle nodal device described in [Section 10.11](#) also suggests that perhaps the functionality of the so-called "capacity slice" p -cycle nodal element could be partly based on reuse of existing ring ADMs. One of the main advantages that p -cycles would offer over span-restorable mesh as the target architecture for ring mining is the ability to incorporate ADMs into the migrated network. In this closing section of the book we outline first at the nodal device level how existing ADMs can be converted to operate as p -cycle nodes, followed by a network-level demonstration of the overall strategy and effectiveness.

11.8.1 Converting Rings into Modular p -Cycles: Nodal View

Having looked in [Section 10.11](#) at a nodal device structure that is the p -cycle counterpart to the ring ADM node element, a BLSR-type ADM ring can be logically viewed as a p -cycle that has no straddling spans, or alternately as an incomplete part of a whole p -cycle, i.e., just the substructure that provided the circumferential protection cycle and the mechanism to cope with "on-cycle" failures. By again making use of the extra traffic feature of BLSR ring systems, we can add the missing functionality related to straddling spans and effectively reuse the ring ADM as part of a p -cycle node. [Figure 11-24](#) shows a generic ADM or OADM as part of a ring configuration. [Figure 11-25](#) shows the same ADM coupled to a new device that supports p -cycle networking access to the prior conventional ring. The only point of physical interface between the new device and the existing ring is that the new device is attached to the extra traffic ports of the ring. When the straddling span interface unit (SSIU) is attached to the ring's extra traffic ports at the co-located ADM, the normal (non-failure) protection channel continuity is then provided by the SSIU, through itself.

Figure 11-24. Conventional ADM in a (BLSR-type) ring.

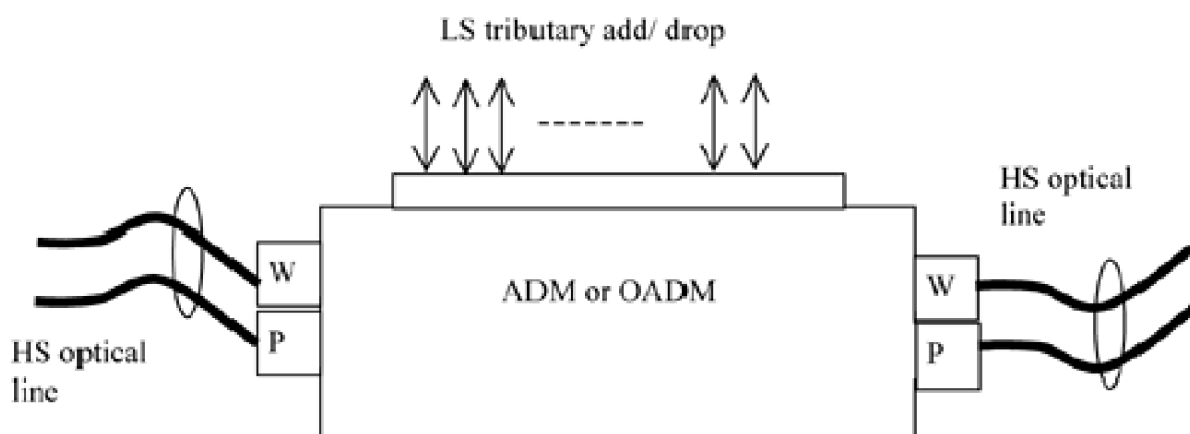
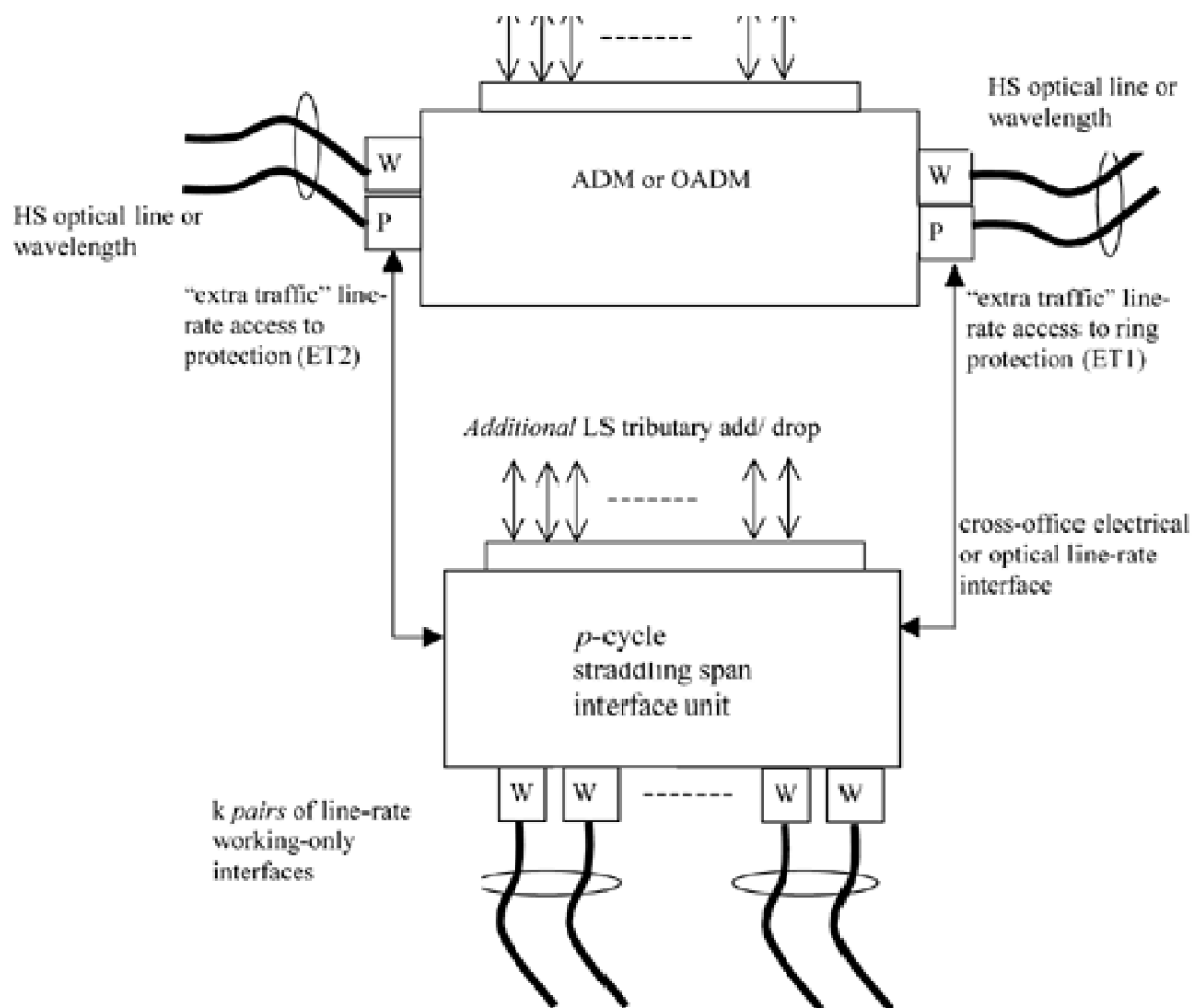


Figure 11-25. Addition of a straddling span interface unit to access the protection channel on a ring, effectively converting the ring to a p -cycle.





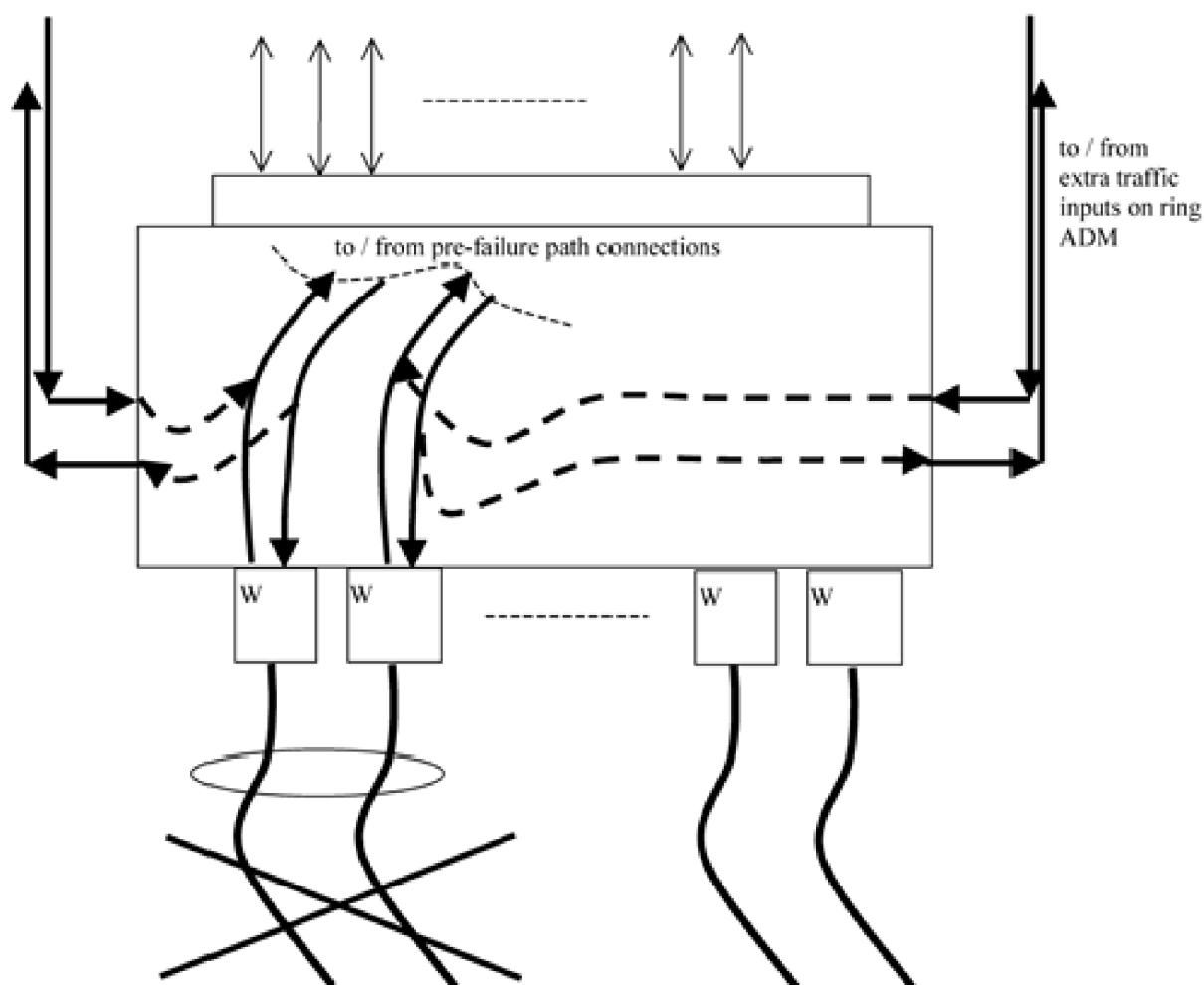
To first order, the ring need not know that the straddling span unit is anything other than an apparent source/sink of some form of low priority traffic at its site. More pragmatically, however, an exchange of state information would be required so the SSIU knows when the protection ring is free in each direction, and for the ADMs to be put in protection lockout mode if the SSIU has accessed protection for a straddling failure. In the case of a SONET ring, or any future ring type where the full protection state and protection protocol is accessible in the line overhead bytes (K1, K2, etc., in SONET), however, cooperation of the SSIU with the ring could be entirely transparent to the existing ADM and made possible through the fact that the SSIU has access to the signaling protocol on the ring protection channel and could be given authority to source/sink protection protocol sequences as needed. Also, because the protection path continuity is through the SSIU, not the ADM, the SSIU can completely observe the status of the protection channel, observe ring switches, and effectively block out or *deny* ring switches when needed due to a prior SSIU switch.

More specifically, the functions of the p -cycle SSIU device are:

1. Normally connects ET1 through to ET2 so that the protection continuity of the ring is normally maintained.
2. Senses either idle pattern or traffic pattern on protection and/or passively monitors the existing ring signaling protocol so it knows the ring protection status. (In many actual product lines this would also connect the SSIU to the ring-wide internal supervisory LAN enabling almost any further exchange of control and status information and development of any new software upgrades to support SSIU-ring interaction). Equivalently this status could be directly provided through inter-element communication or by network management.
3. Upon failure in the preexisting ring (an "on-cycle" failure for the p -cycle) the p -cycle SSIU does nothing except maintain the continuity of the protection channel path through itself. It does, however, note the "in-use" status of the protection channel (as in 2).
4. Upon failure of a straddling span the SSIU interrupts the through-continuity of the protection path of the prior ring and performs BLSR-like loopback switching to substitute the failed working (bidirectional) signals into the ring protection channels. The switching internal to the SSIU on the assumption where both working line-rate systems on a straddling span are present and both fail is illustrated in [Figure 11-26](#). The SSIU uses the ring protection in *both* directions in these circumstances.

Figure 11-26. Illustrating how the straddling span interface unit switches two units of line-rate working

capacity into the ring protection channel, using it as a *p*-cycle.



Note that, in elaboration on (4), the SSIU actually has the further information to make intelligent partial use of the ring protection channel in the following more specific cases:

- A straddling span fails but the ring protection is already in use in both east and west directions from the site of the SSIU. This is an unprotectable dual failure situation. The SSIU leaves the ring protection in place and raises an alarm. A priority scheme could alternately allow the SSIU to override the existing protected signal.
- A straddling span fails but the protection channel is in use in one direction (only) from the SSIU site. Visibility to the signaling state on the non-busy direction lets the SSIU know if the protection channel is free all the way to its peer SSIU on the other end of the respective straddling span that has failed. If so, the SSIU can make use of the protection channel to recover one of its possibly two failed working straddling spans.

11.8.2 Network Level View of Evolution from Rings to *p*-Cycles

In this section we develop two examples to illustrate some of the principles and opportunities involved in ring to *p*-cycle network evolution. A main point is to show in general how we can grow our way toward an efficient mesh from a ring-based starting point, using *p*-cycle technology as the catalyst. The main opportunities for efficiencies arise from:

1. Readmission of "eliminated spans" from the ring design.
2. Elimination of drop and continue bandwidth usage where "matched node" rings meet.
3. Reclamation of high speed (line-rate) interfaces and capacity for protection where rings meet or intersect.

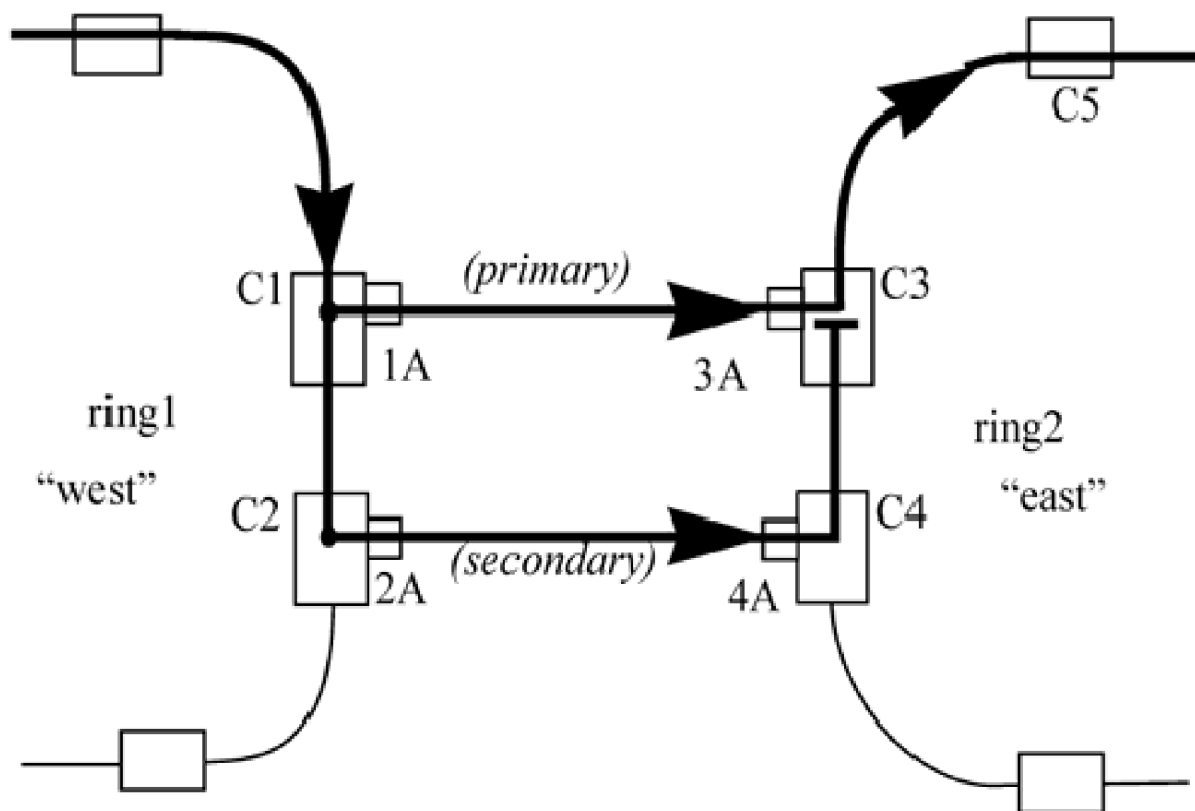
4. Opportunity to introduce new direct working-only transmission spans (or leased l's) as needed for growth, with no corresponding additions of protection capacity required.

The first extended example illustrates all these points. The second more specifically illustrates the reclamation of protection capacity where rings intersect and overlap. First a few words of explanation about some of the effects involved. To explain the "readmission of eliminated spans," let us refer back to [Figure 9-1](#) where we see a published example of a ring network design comprised of five rings, but planned on a richly connected facilities graph. The particular example is a rather extreme case of the span elimination effect in ring planning. Only a fraction of the physical facility graph edges on the left of the figure are actually employed in the most economic ring solution. All other spans are said to have been "eliminated"—not physically, but from the space of routing possibilities employed henceforth once the ring set is deployed. If the same five rings in the right hand side of [Figure 9-1](#) become p -cycles, then all such "eliminated" spans are readmitted for shortest path routing of working demands. From being disused assets bearing zero working capacity they become p -cycle straddling spans that can each bear twice as much working capacity as the corresponding rings previously had as protection capacity.

The second point above is an opportunity that arises when two rings that have an initially side-by-side or "tiling" relationship on the plane, such as in [Figure 10-5](#), are combined into a single p -cycle. One effect is the direct reclamation of the protection capacity to become straddling span working capacity. Part of this line-rate capacity may be used to form the p -cycle itself, however, at the location where the two rings are being joined. The example to follow shows this.

But in addition, coupled rings often employ the Dual Ring Interconnect (DRI) or "matched node" technique to protect transiting demands from cross-office wiring failures or isolated node failure at the crossing points. This requires "dropping" the respective arrangement which yields even greater advantage when converted to p -cycles as illustrated in [Figure 11-27](#) shows two rings A and B with matched node interconnection. The service path transfers redundantly via two nodes using the drop and continue feature of ADMs. In [Figure 11-27](#) a unidirectional signal flow is drawn, but a mirror image for the opposite signal direction is implied. C3 receives two copies of the transferring signal. In the event of failure affecting the primary add signal, it switches to the surviving signal copy arriving in the line signal for C4. This arrangement resists any single failure of the ADM cores or add/drop interfaces involved. But because of the way we convert to a p -cycle below, transiting demands such as in [Figure 11-27](#), retain similar single-node failure protection.

Figure 11-27. Drop and continue arrangement between "matched" ring nodes.



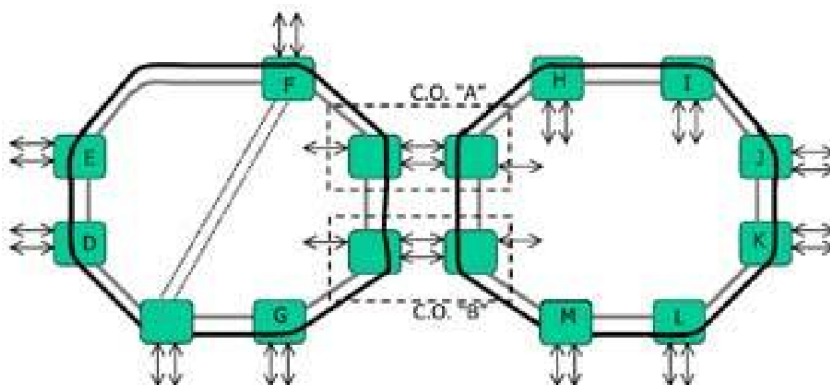
Let us now show a simple extended example of how graceful and effective the evolution from rings to p -cycles could be in many typical cases. We start by considering just two coupled rings from a legacy ring-based network and note that a span elimination was involved at the time these rings were designed. This is the initial situation in [Figure 11-28\(a\)](#). We assume that what might trigger the first planning action is imminent exhaust on the "facing" spans between buildings A and B due to drop and continue (D&C) capacity consumption. The initial pair of rings employs 15 line-rate optical interface pairs for working and for protection, giving an initial ratio of protection channels to

working channels of 100%. The first step in evolution is to form a p -cycle out of the two rings. This is done by eliminating two of the ADMs where the rings used to interface and adding SSIUs to the remaining two ADMs. Although not shown in detail the p -cycle is formed by using the existing ADM line rate interfaces that used to serve the D&C spans, to now complete the outer perimeter of the p -cycle.

Figure 11-28. A legacy pair of rings is converted to p -cycle and evolved with growth to an efficient mesh. No further protection capacity is added as growth is sustained.

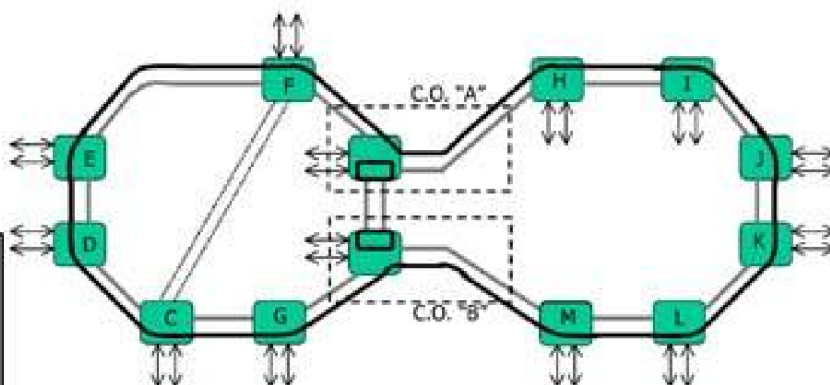
(a) Legacy Network Situation:

- Two coupled rings with one span elimination.
- Ring capacity exhaust imminent on D&C spans.
- Spare/Working ratio = $15/15 = 100\%$.
- 15 ADMs used.



(b) Growth Step 1:

- Convert to one p -cycle.
- Spare/Working ratio = $13/15 = 87\%$.
- Salvage 2 ADMs.
- Add 2 SSIUs.
- D&C congestion relieved.
- ADM line interfaces used to complete p -cycle.

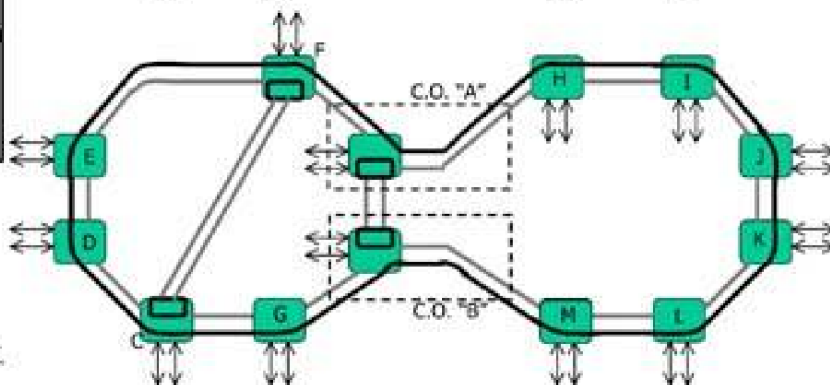


Legend

- ↔ add/drop trib. rate I/F
- working line rate I/F
- protection
- legacy span elimination
- ADM
- SSIU

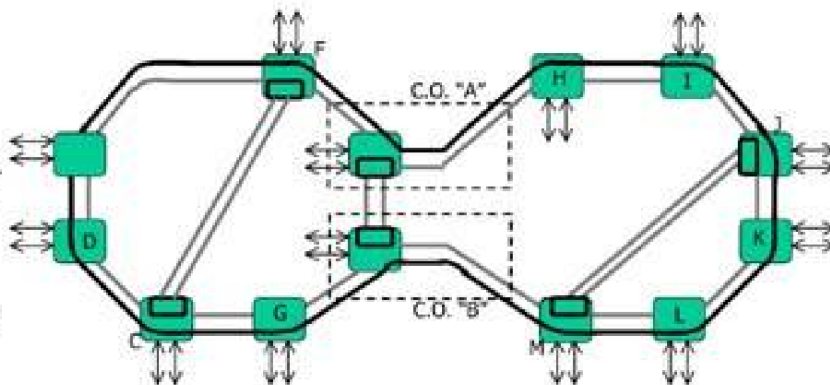
(c) Growth Step 2:

- Trigger: pending exhaust in regions of E D C G.
- Reinstate eliminated span.
- Spare/Working ratio = $13/17 = 76\%$.
- Add 2 more SSIUs at C and F.
- Return to shorter working routes in left-hand region.



(d) Growth Step 3:

- Trigger: pending exhaust in regions of M L K J.
- Action: Lease two new diverse λ paths to straddle M-J (working).
- Spare/Working ratio = $13/19 = 68.4\%$.
- Add 2 more SSIUs and release capacity by rerouting on shorter working routes in right-hand region.



The SSIUs support one straddling span of two working channels where the D&C spans used to be. Two p -cycle nodes replace the prior four matched-node ADMs. No new protection capacity is added and the spare to protection ratio drops to 87%. Although a failure of either of the p -cycle nodes will cause outage for demand flow that is routed over the new straddling spans, demands that previously transiting

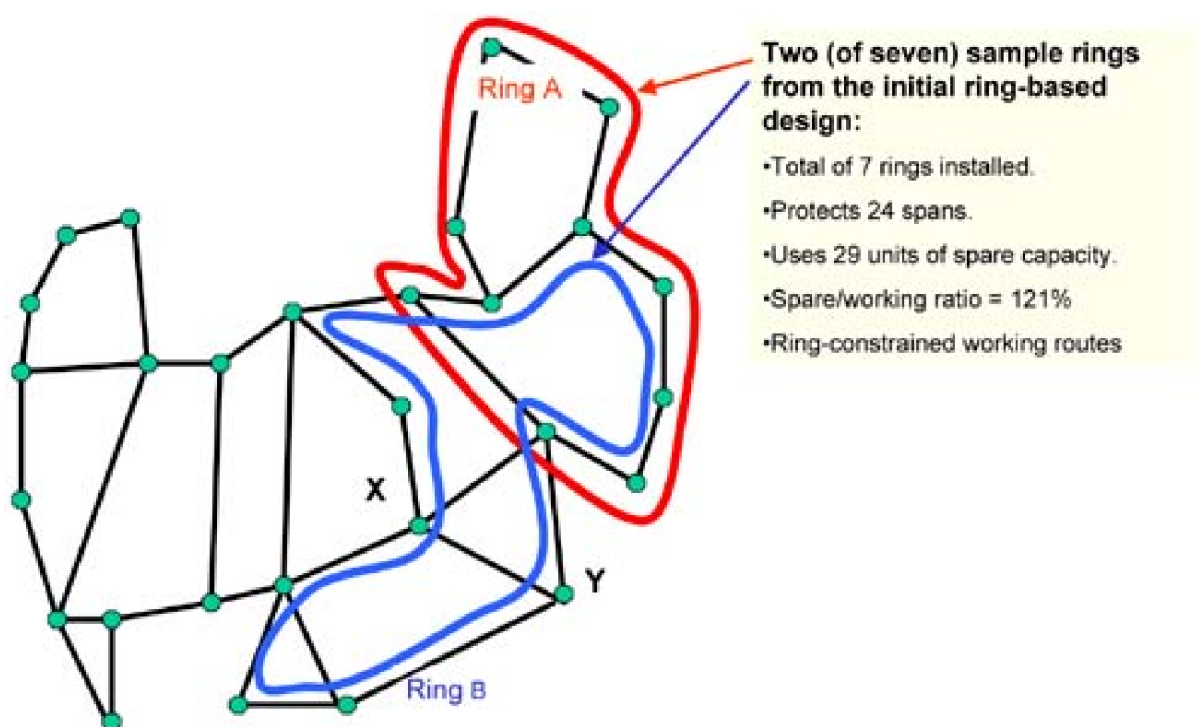
between rings via D&C arrangements at the same locations, have equivalent survivability against p -cycle node loss because they are routed on the (ring-like) perimeter of the p -cycle. So they enjoy the same transit-demand survivability against node-loss that a ring provides against a single failure of one of its ADM nodes.

Now suppose that some time later continued growth of demand routed through the vicinity of nodes D C G B threatens to exhaust the working capacity on one or more spans in that region. This could serve as the trigger for the next planning action—to reinstate the old eliminated span and commission two line-rate working channels on it. This relieves capacity on all the spans mentioned as well as shortening many working routes in the vicinity in general. The redundancy is now down to ~76%. Again, growth has been served with no further additional protection capacity. Investment in SSIUs and added working capacity is made, but this is purely to serve the growth in demand. The legacy protection capacity is being financially leveraged because the prior existing ring protect channels, used as a p -cycle, are being stretched to serve more efficiently. In the final step we postulate similar accumulating growth in the right-hand region of the example and show how the operator might again respond efficiently by leasing two new working-only lightpaths^[8] between nodes M and J, or more generally, acquiring their own new M-J span. At this stage we have grown the network into a fairly mesh-like redundancy of 68% and only spent money in this evolution on equipment and capacity directly needed to serve demand growth, and only when and where it actually materializes.

^[8] If leased from another carrier an assurance of physical diversity from route M-L-K-J is required in the example.

Let us now look at another example where the rings intersect in what is referred to as an "Olympic Rings" relationship, as opposed to abutting in a tiling relationship as above. As an example to work with [Figure 11-29](#) isolated two out of seven initial rings in one of the test case ring networks used for the ring mining studies above. These two rings have a total circumference of 29 hops but because of unavoidable span overlap effects, only 24 working channel-hops are protected. The overall redundancy (not counting partial fill or stranded capacity within working channels) is ~121%. In addition, although not quantified, the routing of all working demands is ring-constrained. For example, demand between nodes X-Y—more generally all network demands for which span X-Y is on the natural shortest path are detoured around Ring B and cannot take the direct hop from Y to X.

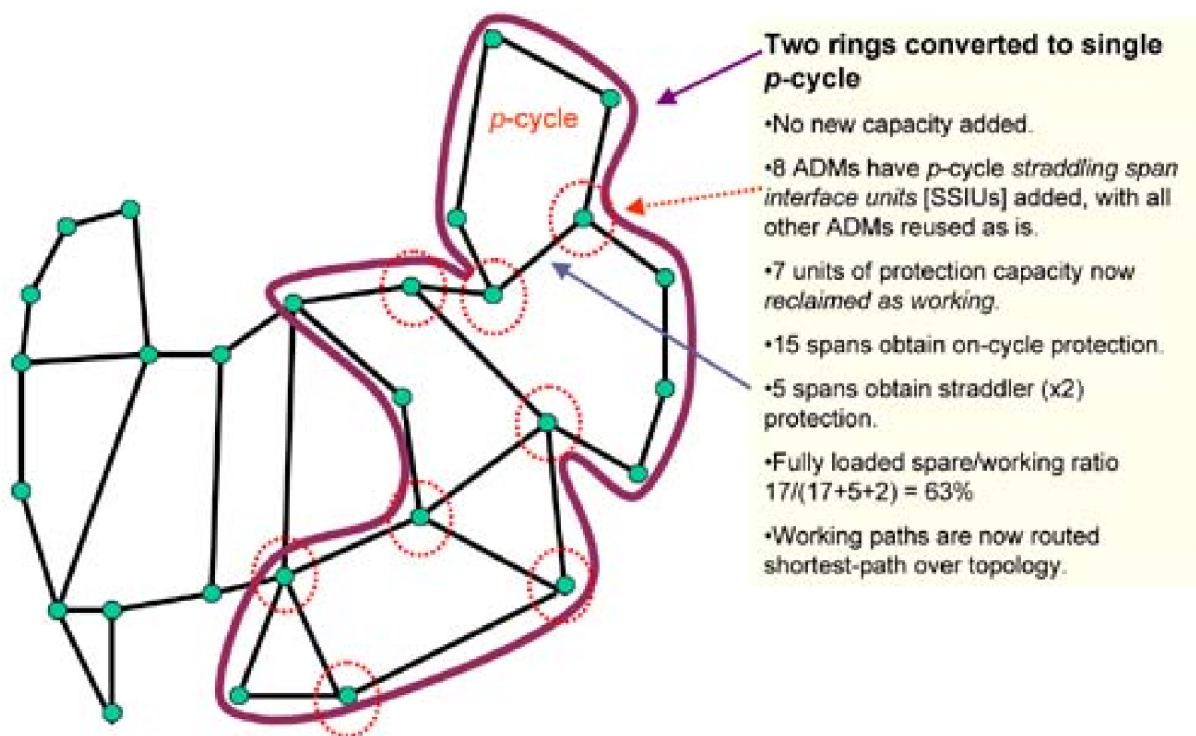
Figure 11-29. Two rings in an "Olympic Ring" intersection form of relationship.



[Figure 11-30](#) shows the same two rings converted into a p -cycle. The cost of the conversion is the deployment of SSIU devices at 8 nodes to convert the existing ADMs at those nodes to function as p -cycle nodes. This converts one span from each prior ring into new straddling spans and readmits three previously "eliminated" spans as straddling spans on the new p -cycle. All other ADMs are reused as is. (In fact such nodes need not strictly ever know they are not still parts of a ring.) In return for the eight SSIU devices, we recover two units of available working capacity from each of the two prior ring spans that were made into straddlers and on the five spans where the ring spans overlapped we may also be able to recoup two units of line capacity and salvage the ADMs as well. At the least, on such spans we can recoup one of the protection channels as this is not needed in the new single p -cycle. Finally, we can calculate that if the resulting p -cycle was ultimately fully loaded with working demands over all its straddler and on-cycle spans, the structure as a whole would be only 63%

redundant. That is a growth potential of 192% relative to the same amount of protection capacity investment in the two-rings case. On the other hand, to realize the potential on the three spans that were previously eliminated, we may have to add two units of new working capacity to realize this full build-out condition (assuming the eliminated spans were not equipped with available transmission channels). For a more comprehensive treatment of the evolution of ring-based networks into p -cycle networks, see [KoGr03].

Figure 11-30. Two intersecting rings converted to a single p -cycle yielding 192% increase in working channel capacity available for service provisioning.



A final point is to acknowledge that through the processes just exemplified, we could in principle go on to build such an efficient p -cycle structure that we exceed separate objectives from a dual failure availability standpoint. Such considerations will lead to practical limitations on the allowable circumference of p -cycles and/or the maximum number of straddling span relationships. In a sense, however, this is only "a good problem to have" in that we are capable of building such remarkably efficient survivable networks that we would have to invoke separate criteria to limit their effectiveness.

END

For much more on survivable networks please see the book's web site.

[[Team LiB](#)]

Bibliography

[Ada91] J.L. Adams, *Flying Buttresses, Entropy and O-rings: The World of an Engineer*, Harvard University Press, Cambridge Mass, 1991.

[Alle96] B. Allen, "SONET", Chapter 100 in *The Electronics Handbook*, J.C. Whitaker (editor in chief), CRC and IEEE Press, 1996, pp. 1524-1537.

[AnMi82] D. Anick, D. Mitra, M.M. Sondhi, "Stochastic Theory of a Data-Handling System with Multiple Sources," *Bell System Technical Journal*, vol.61, 1982, pp. 1871-94.

[AINa98] J.D. Allen, S. Nathan, J. Huang, "Rings in a highly-connected network—an economic comparison," *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, Brugge, Belgium, May 1998, paper 042.

[AN95a] ANSI, *Synchronous Optical Network (SONET) - Basic Description including Multiplex Structure, Rates and Formats*, ANSI T1.105-1995, 1995.

[AN95b] ANSI, *Synchronous Optical Network (SONET) - Automatic Protection Switching*, ANSI T1.105.01-1995, 1995.

[ArPa00] P. Arijs, W. Van Parys, P. Demeester, "Cost and availability comparison of WDM mesh and ring network architectures," *Proc. Second Int. Workshop on the Design of Reliable Comm. Networks, DRCN 2000*, April 2000, Munich, Germany, pp. 33-38.

[Ash91] G. Ash, J.-S. Chen, A. Frey, B. Huang, "Real-Time Network Routing in a Dynamic Class-of-service Network," *13th Intl Teletraffic Congress*, Copenhagen, June 1991, pp. 187-194.

[AsSh01] C. Assi, A. Shami, M.A. Ali, R. Kurtz, D. Guo, "Optical networking and real-time provisioning: an integrated vision for the next-generation Internet," *IEEE Network Magazine*, July/August 2001, pp. 36-45.

[Baas00] Baase, S., Gelder A. V., *Computer Algorithms: Introduction to Design and Analysis*, 3rd Edition, Addison Wesley Longman, 2000.

[BaGe00] S. Baase, A.V. Gelder, *Computer Algorithms: Introduction to Design and Analysis*, 3rd Edition, Addison Wesley Longman, 2000.

[Bake91] Baker, J. E., "A distributed link restoration algorithm with robust preplanning," *Proc. IEEE GLOBECOM '91*, Dec. 1991, pp. 306-311.

[BaPu92] M. Barezzani, S. Pupolin, M. Zorzi, "A new definition of transmission network availability with applications," *European Transactions on Telecommunications*, vol. 3, no. 4, Jul.-Aug. 1992, pp. 349-357.

[BeBl00] M. Beshai, F. J. Blouin, "Courteous Routing," in *Proc. Networks 2000-Toward Natural Networks: 9th International Telecommunication Network Planning Symposium*, Toronto, Canada, 10-15 Sept. 2000, paper #38.

[BeBu93] D. Beasley, D. R. Bull, R. R. Martin, "An Overview of Genetic Algorithms: Part 1- Fundamentals," URL - <ftp://alife.santefe.edu/pub/USER-AREA/EC/GA/papers/over93.ps.gz>, 1993.

[BeFe98] M. Bettin, G. Ferraris, G. Pignari, "Comparison of protection and restoration schemes for SDH networks," *Proc. 1st Int. Workshop Design Reliable Comm. Networks, DRCN'98*, Belgium, 1998.

[BeG92] D. Bertsekas, R. Gallager, *Data Networks*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 113.

[Bell93] *Digital Cross-connect systems in Transport Network Survivability*, Bellcore Special Report SR-NWT-002514, Issue 1, Jan. 1993.

[Bell88] Bellcore, *Digital Cross-Connect System (DCS) Requirements and Objectives*, TR-TSY-000170, Issue 1, November 1988.

[Bell95] Bellcore, *SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria*, GR-1400-Core, Issue 1,

Revision 1, October 1995.

[Be1195b] Bellcore, *SONET Bidirectional Line-Switched Ring Equipment Generic Criteria*, GR-1230-Core, Issue 2, November 1995.

[Bell99] Bell Labs, "Bell Labs Uses Ultra-Dense WDM to Transmit 1,022 Channels over Fiber," Bell Labs Press Release, Nov. 10, 1999.

[BeYa00] G. Bernstein, J. Yates, D. Saha, "IP-Centric control and management of optical transport networks," *IEEE Communications Magazine*, Oct. 2000, pp. 161-167.

[Bhan99] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, 1999.

[BiAl92] R. Billinton, R. N. Allan, *Reliability Evaluation of Engineering Systems*, 2nd Edition, Plenum Press, 1992.

[BoM76] J.A. Bondy, U.S.R. Murty, *Graph Theory with Applications*, North-Holland, New York, 1976.

[BoRo02] P. Bonenfant, A. Rodriguez-Moral, "Generic Framing Procedure (GFP): The Catalyst for Efficient Data over Transport," *IEEE Communications Magazine*, May 2002, pp. 72-79

[BrGr94] G.N. Brown, W.D. Grover, J.B. Slevinsky, M.H. MacGregor, "Mesh/Arc Networking: An Architecture for Efficient Survivable Self-Healing Networks," *Proc. of the IEEE Intl. Conf. on Comm (ICC '94)*, Vol. 1, May 1994, pp. 471-477.

[Brua92] R. Brualdi, *Introductory Combinatorics*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[Brun00] D.A. Brungard, "Draft ITU-T G.709 Network Node Interface for the OTN", *Contribution to T1 Standards Project T1X1.5/2000-091*, Feb. 2000.

[Cahn98] Robert S. Cahn, *Wide Area Network Design - Concepts and Tools for Optimization*, Morgan Kaufman Publishers, San Francisco, 1998, Chapters 8, 9.

[CaNa97] H. C. Cankaya, V. S.S. Nair, "Reliability and availability evaluation of self-healing SONET mesh networks", *Proc. IEEE Globecom 1997*, pp. 252-256.

[CaPa98] B. Van Caenegem, W. Van Parys, F. De Turck, P. M. Demeestere, "Dimensioning of Survivable WDM Networks", *IEEE Jour. Selected Areas in Communications*, Vol. 16, No. 7, pp. 1146-1157, September 1998.

[ChGe74] W. Chou, M. Gerla, H. Frank, J. Eckl, "A cut saturation algorithm for topological design of packet switched networks", *Proc. IEEE National Telecom. Conf.* pp. 1074-1085, Dec. 1974.

[ChKo91] T. Chujo, H. Komine, K. Miyazaki, T. Ogura, T. Soejima, "Distributed self-healing network and its optimum spare capacity assignment algorithm", *Electronics & Communications Japan*, part 1, vol. 74, no. 7, 1991, pp. 1-8.

[ChDo91] C.W. Chao, P. M. Dollard, J. E. Weythman, L. T. Nguyen, H. Eslambolchi, "FASTAR: a robust system for fast DS-3 restoration," *Proc. IEEE GLOBECOM '91*, pp. 39.1.1-39.1.5, 1991.

[CiHe00] T. Cinkler, T. Henk, G. Gordos, "Stochastic algorithms for design of thrifty single-failure-protected networks," *Proc. IEEE / VDE. Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, April 2000, pp. 298-303.

[Cis99] Cisco Systems, *Dynamic Packet Transport Solution*, GSR 12000 OC-12c/STM-4c Packet Ring Line Card, Data Sheet, San Jose, CA, 1999.

[Cisc99] Cisco Systems, *Dynamic Packet Transport Technology*, A Cisco Systems White Paper, 1999, 17 pp.

[ClGr00] M. Clouqueur, W. D. Grover, "Computational and Design Studies on the Unavailability of Mesh-restorable Networks", *Proc. IEEE/VDE Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, April 2000, pp. 181-186.

[ClGr01] M. Clouqueur, W. D. Grover, D. Leung, O. Shai, "Mining the Rings: Strategies for Ring-to-Mesh Evolution," *Proc. 3rd International Workshop on the Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, October 2001, pp. 113-120.

[ClGr02] M. Clouqueur, W.D. Grover, "Dual failure availability analysis of span-restorable mesh networks," *IEEE JSAC*, vol.20, no. 4, May 2002, pp. 810-821.

[ClGr02b] M. Clouqueur, W. D. Grover, "Mesh-restorable Networks with Complete Dual-failure Restorability and with Selectively Enhanced Dual-failure Restorability Properties," in *Proceedings of OptiComm 2002*, Boston, July 29-Aug. 2, 2002, pp. 1-12.

[ClGr03] M. Clouqueur, W.D. Grover, "Quantitative comparison of end-to-end availability of service in ring and mesh-restorable

networks," *Proceedings of NFOEC 2003*, Orlando, FL, USA, September 2003.

[Coan91] B.A. Coan, et al., "Using distributed topology updates and preplanned configurations to achieve trunk network survivability," *IEEE Transactions on Reliability*, vol. 40, no.4, 1991, pp. 404-416.

[Colb87] C.J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, Oxford and New York, 1987.

[CoMa02] *IEEE Communications Magazine*, Special Issue on Generic Framing Procedure and Data over SONET/SDH and OTN, vol. 40, no.5, May 2002, pp.60-112.

[Corte99] B. Cortez, "Integrated Rings in New Digital Cross Connect Systems," *Presentation at IEEE DCS Workshop*, June 23, 1999.

[CrMa98] F.R.B Cruz, J. MacGregor Smith, G.R. Mateus, "Solving to optimality the uncapacitated fixed-charge network flow problem", *Computers and Operation Research*, vol. 25, issue 1, 1998, pp. 67-81.

[CPL94] CPLEX Optimization, Inc., *CPLEX 3.0 User Documentation*, Suite 279, 930 Tahoe Blvd., Bldg. 802, Inline Village, NV USA, 1994.

[Craw93] D. Crawford, "Fiber Optic Cable Dig-ups" Causes and Cures," in *Network Reliability: A Report to the Nation - Compendium of Technical Papers*, National Engineering Consortium, Chicago, June 1993.

[CRLDP] Jamoussi, et. al., "Constraint-Based LSP Setup using LDP," RFC 3212, IETF, January 2002.

[DaRe00] B. Davie, Y. Rekhter, *MPLS Technology and Applications*, Morgan Kaufmann Publishers, San Francisco, 2000.

[Dem99] P. Demeester et al., "Resilience in Multilayer Networks," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 70-75.

[Deo84] N. Deo, C-Y. Pang, "Shortest-Path Algorithms: Taxonomy and Annotation," *Networks*, vol.14, 1984, pp. 275-323.

[Dijk59] E.W. Dijkstra, "A note on two problems in connection with graphs," *Number Math.*, Vol. 1, 1959, pp. 269-271.

[Dixi03] S. S. Dixit (editor), *IP over WDM: Building the Next Generation Optical Internet*, John Wiley and Sons, 2003.

[DoHa98] B. Doshi, P. Harshavardhana, "Broadband network infrastructure of the future: roles of network design tools in technology deployment strategies," *IEEE Communications Magazine*, vol. 36, no.5, May 1998, pp.61-71.

[DoWi94] R.D. Doverspike, B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques", *Journal of Networks and System Management*, Vol. 2, No. 2, pp. 95-123, 1994.

[DoYa01] R. Doverspike, J. Yates, "Challenges for MPLS in Optical Network Restoration," *IEEE Communications Magazine*, Feb. 2001, pp. 89-96

[DoGr00] J. Doucette, W. D. Grover, "Influence of Modularity and Economy of scale Effects on Design of Mesh-Restorable DWDM Networks," *IEEE JSAC*, vol.18, no.10, Oct. 2000, pp. 1912-1923

[DoGr01] J. Doucette, W. D. Grover, "Comparison of Mesh Protection and Restoration Schemes and the Dependency on Graph Connectivity", *Proc. 3rd International Workshop on the Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, October 2001.

[DoGr01a] J. Doucette, W. D. Grover, T. Bach, "Bi-criteria studies of mesh network restoration path-length versus capacity trade-offs," *Proc. OSA Optical Fiber Communications Conference (OFC 2001)*, Anaheim, Calif., March 17-22 2001, pp. TuG2-1 - TuG2-3.

[DoGr02] J.E. Doucette, W. D. Grover, "Maintenance-Immune Optical Mesh Network Design," *Proc. 18th National Fiber Optic Engineers Conference (NFOEC 2002)*, Dallas, Sept. 15-19, 2002, pp. 2049-2061.

[DoGr02b] J.E. Doucette, W. D. Grover, "Capacity Design Studies of Span-Restorable Mesh Networks with Shared-Risk Link Group (SRLG) Effects," in *Proc. OptiComm 2002*, Boston, July 29-Aug. 2, 2002, pp.25-38.

[DoCl03] J. Doucette, M. Clouqueur, W.D. Grover, "On the Service Availability and Respective Capacity Requirements of Shared Backup Path-Protected Mesh Networks," to appear in *Optical Networks Magazine* Special Issue on Engineering the Next Generation Optical Internet, vol. 4, no. 6, Nov/Dec 2003.

[DPW99] R.D. Doverspike, S. Phillips, J.R. Westbrook, "Future Transport Network Architectures," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 96-101.

[Dubh96] D. P. Dubhashi, "What can't you do with LP?," *Basic Research in Computer Science (BRICS) Lecture Series*, BRICS LS-96-5,

University of Aarhus, Denmark, December, 1996. Internet: www.brics.dk

[DuGr94] D.A. Dunn, W.D. Grover, M.H. MacGregor, "A comparison of k-shortest paths and maximum flow methods for network facility restoration", *IEEE JSAC*, Jan. 1994, vol. 12, no. 1, pp. 88-99.

[DWY99] P. Demeester, T.-H. Wu, N. Yoshikai, "Survivable Communication Networks," *IEEE Communications Magazine*, August 1999, pp. 40-42.

[EIBo03] G. Ellinas, E. Bouillet, R. Ramamurthy, J. Labourdette, S. Chaudhuri, K. Bald, "Routing and restoration architectures in mesh optical networks," *Optical Networks Magazine*, vol.4, no.1, Jan/Feb 2003, pp. 91-106.

[EIBu00] S.N. Elahmadi, P.A. Bullock, K. Nagaraj, T. Flanagan, "Telecommunications network having shared protect capacity architecture," *United States Patent* No. 6,154,296, Nov. 28, 2000.

[EIHa00] G. Ellinas, A. G. Hailemariam, T.E. Stern, "Protection Cycles in Mesh WDM Networks," *IEEE JSAC*, Oct.2000, pp. 1924-37.

[Ezem03] Chioma Ezema, *Topology Design of Mesh-Restorable Networks*, M.Sc. Thesis, University of Alberta, Spring 2003.

[Falc90] W.E. Falconer, "Service assurance in modern telecommunications networks," *IEEE Communications Magazine*, June 1990, pp. 32-39.

[Fee99] J.A. Fee, "Method and system for optical restoration tributary switching in a fiber network," *United States Patent* No. 5,884,017, March 16, 1999.

[Fish81] M.L. Fisher, "The Lagrangean relaxation method for solving integer programming problems," *Management Science*, 27(1):1-18, January 1981.

[Flan90] T. Flanagan, "Fiber Network Survivability," *IEEE Communications Magazine*, June 1990.

[FoFu62] L.R. Ford Jr., D.R. Fulkerson, *Flows in Networks*, Princeton University Press (1962).

[Fons98] P.F. Fonseca, "Pan-European multi-wavelength transport networks: network design, architecture, survivability and SDH networking," *Design of Reliable Comm. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 17-20, 1998, paper P3.

[FoGa93] R. Fourer, D. Gay, B. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Boyd & Fraser, 1993, pp 2-10.

[Fran97] A. Frangioni, *Dual-Ascent Methods and Multicommodity Flow Problems*, Ph.D. Thesis - TD 5/97, Dipartimento di Informatica, Università di Pisa-Genova-Udine, March 1997.

[Free96] R. L. Freeman, *Telecommunication System Engineering*, 3rd Ed. New York: Wiley, 1996, pp. 444-447.

[Free96b] R.L. Freeman, *Reference Manual for Telecommunications Engineering*, Updated 2/e, John Wiley & Sons, 1996, p. 2047.

[Free02] R. L. Freeman, *Fiber-Optic Systems for Telecommunications*, Wiley-Interscience, 2002.

[FrSo01] J. Frodsham, A. Solheim, "Next-generation Backbone Network Metrics," *National Fiber Optic Engineers Conference (NFOEC) 2001*. (see also www.innovance.com)

[GaJo79] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.

[Galv93] R. D. Galvao, "The use of Lagrangean relaxation in the solution of uncapacitated facility location problems," *Location Science*, vol.1, no.1, 1993, pp.57-79.

[GaTa92] R. Garnier, J. Taylor, *Discrete Mathematics for New Technology*, Adam Hilger Press, 1992, p. 468.

[Gavi91] B. Gavish, Topological design of telecommunication networks - Local access design methods, *Annals of Operations Research*, 33: (1991) 17-71.

[Gavi90] B. Gavish "Backbone network design tools with economic tradeoffs", *ORSA J. Computing*, 2: 236-252, 1990

[GeAn01] N. Geary, A. Antonopoulos, E. Drakopoulos, J. O'Reilly, J. Mitchell, "A Framework for Optical Network Planning under Traffic Uncertainty," *Design of Reliable Communication Networks (DRCN 2001)* Budapest, Hungary, October 2001.

[GeCr99] B. Gendron, T.G. Crainic, A. Frangioni, "Multicommodity capacitated network design," in *Telecommunications Network Planning*, eds. B. Sanso, P. Soriano, Kluwer Academic, 1999, pp. 1-19.

- [GeK97] O. Gerstel, S. Kutten, Dynamic Wavelength Allocation in All-Optical Ring Networks, *Proc. of ICC '97*, 1997, pp. 432-436.
- [Gers00] O. Gerstel, "Optical Layer Signaling: How Much is Really Needed?," *IEEE Communications Magazine*, Oct. 2000, pp. 154-160.
- [GhDe01] N. Ghani, D. Saha (editors) *IEEE Network Magazine: Special Issue on IP-Optical Integration*, July/August 2001
- [GHS94] L.M. Gardner, M. Heydari, J. Shah, I.H. Sudborough, I.G. Tollis, C. Xia, "Techniques for Finding Ring Covers in Survivable Networks," *Proc. of IEEE Globecom '94*, 1994, pp. 1862-1866.
- [GILa97] F. Glover, M Laguna, *Tabu Search*, Kluwer Academic Press, 1997.
- [GILa99] F. Glover, M. Laguna, Tabu search. in *The Handbook of Applied Optimization*, (P. M. Pardalos, M. G.C. Resende, editors), Oxford Academic Press, <http://bus.colorado.edu/Faculty/Laguna/Papers/ts2.pdf>, 1999.
- [GMP01] Berger, L., Ashwood-Smith, Peter, editors, "Generalized MPLS - Signaling Functional Description," Internet Draft, draft-ietf-mpls-generalized-signaling-02.txt, (work in progress), March 2001.
- [Gora01] W. Goralski, *Optical Networking & WDM*, Osborne/McGraw-Hill, Berkeley, 2001
- [GrBi91] W.D. Grover, T.D. Bilodeau, B.D. Venables, "Near Optimal Synthesis of a Mesh Restorable Network", *Proceedings of IEEE GLOBECOM'91*, Dec. 1991, pp. 2007-2012.
- [GrCl01] W.D. Grover, M. Clouqueur, T. Bach, "Quantifying and Managing the Influence of Maintenance Actions on the Survivability of Mesh-Restorable Networks," *Proc. 17th Annual National Fiber Optic Engineers Conference (NFOEC 2001)*, July 8-12, Baltimore, 2001, pp. 1514-1525.
- [GrCl02] W.D. Grover, M. Clouqueur, "Span-Restorable Mesh Network Design to Support Multiple Quality of Protection (QoP) Service-Classes," *1st Int'l Conf. Optical Commun. & Networks (ICOCN'02)*, Singapore, Nov. 11-14, 2002, pp. 321-323.
- [GrDo01] W.D. Grover, J. Doucette, "Topological design of survivable mesh-based transport networks," *Annals of Operations Research: Topological Network Design in Telecommunication Systems*, P. Kubat, J. MacGregor Smith (Editors), P.L. Hammer (Editor-in-Chief), Kluwer Academic, v.106, September 2001, pp.79-125.
- [GrDo02] W. D. Grover, J. Doucette, "Design of a Meta-Mesh of Chain Sub-Networks: Enhancing the Attractiveness of Mesh-Restorable WDM Networking on Low Connectivity Graphs," *IEEE JSAC*, vol. 20, no. 1, pp. 47-61, January 2002.
- [GrDo02b] W. D. Grover, J. E. Doucette, "Advances in optical network design with p -cycles: Joint optimization and pre-selection of candidate p -cycles," *Proceedings of the IEEE-LEOS Summer Topical Meeting on All Optical Networking*, Mont Tremblant, Quebec, July 15-17, 2002.
- [Grlr99] W.D. Grover, R.R. Iraschko, Y. Zheng, "Comparative Methods and Issues in Design of Mesh-Restorable STM and ATM Networks," in *Telecommunication Network Planning*, B.Sanso, P. Soriano (editors), Kluwer Academic Publishers, 1999, pp. 169-200.
- [GrLi99] W.D. Grover, D.Y. Li, "The Forcer Concept and Express Route Planning in Mesh Survivable Networks," *Journal of Network and Systems Management*, vol.7, No.2, Feb-Mar'99, pp. 199-223.
- [GrMa94] W. D. Grover, M. H. MacGregor, "Potential for spare capacity preconnections to reduce crossconnection workloads in mesh-restorable networks," *Electronics Letters*, Vol. 30, No. 3, February 3, 1994, pp. 194-195.
- [Grov89] W. D. Grover, *Selfhealing Networks - A Distributed Algorithm for k-shortest link-disjoint paths in a multigraph with applications in realtime network restoration*, Ph.D. Dissertation, Univ. of Alberta, Fall, 1989.
- [Grov92] W.D. Grover, "Case Studies of Survivable Ring, Mesh and Mesh-Arc Hybrid Networks," in *Proc. IEEE Globecom '92*, Vol. 1, pp. 633-638.
- [Grov97] W. D. Grover, "Self-organizing Broad-band Transport Networks," *Proceedings of the IEEE: Special Issue on Communications in the 21st Century*, vol. 85, no.10, pp. 1582-1611, October 1997.
- [GrRa97] W.D. Grover, V. Rawat, M. MacGregor, "A Fast Heuristic Principle for Spare Capacity Placement in Mesh-Restorable SONET/SDH Transport Networks," *Electronics Letters*, vol.33, no.3, Jan. 30, 1997, pp.195-196.
- [GrVe91] W. D. Grover, B. D. Venables, M. H. MacGregor, J. H. Sandham, "Development and performance verification of a distributed asynchronous protocol for real-time network restoration," *IEEE J. Select. Areas Comm.*, vol. 9, no. 1, pp. 112-125, January 1991.

- [GrKe91] G. Grover, A. Kershenbaum, P. Kermani, "MENTOR: an algorithm for mesh network topological optimization and routing," *IEEE Trans. Comm*, 39: 503-513, 1991
- [GrMa00] W. D. Grover, R. G. Martens, "Optimized design of ring-mesh hybrid networks," in *Proc. IEEE / VDE. Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, April 2000, pp. 291-297.
- [Grov87] W. D. Grover, "The selfhealing network: A fast distributed restoration technique for networks using digital cross-connect machines," *Proc. IEEE Globecom'87*, Tokyo, 1987, pp. 1090-1095.
- [Grov99] W. D. Grover, "High availability path design in ring-based optical networks," *IEEE/ACM Trans. on Networking*, vol.7, no.4, pp. 558-574, Aug. 1999.
- [Grov99b] W. D. Grover, "Resource management for fault tolerant paths in SONET ring networks," *Journal of Networks and Systems Management* (Plenum Publishing), vol.7, no.4, pp. 373-394, Dec. 1999.
- [Grov94] W.D. Grover, "Distributed Restoration of the Transport Network," Chapter 11 in *Network Management into the 21st Century*, (T. Plevyak, S. Aidarous, editors), IEEE / IEE Press co-publication, 1994, pp. 337-417.
- [GrSt98] W.D. Grover, D. Stamatelakis, "Cycle-oriented distributed pre-configuration: ring-like speed with mesh-like capacity for self-planning network restoration," *Proc. IEEE International Conf. Commun. (ICC '98)*, Atlanta, June 8-11 pp. 537-543.
- [GrSt98b] W. D. Grover, D. Stamatelakis, "Self-organizing closed-path configuration of spare capacity in broadband mesh transport networks," *Proc. Canadian Conference on Broadband Research (CCBR'98)*, Ottawa, June 12-24, 1998, pp.145-156.
- [GrSt00] W. D. Grover, D Stamatelakis, "Bridging the ring-mesh dichotomy with p -cycles," *Proc. Design of Reliable Networks Conference (DRCN 2000)*, Munich, April 2000, pp. 92-104
- [Grub01] Claus G. Gruber, "Deployment of p -Cycles in WDM Networks," *Diploma Thesis, Institute of Communication Networks*, Technical University of Munich, August 20, 2001.
- [GrZh98] W. D. Grover, Y. Zheng, "VP-based ATM network design with controlled over-subscription of restoration capacity," *Proc. 1st Int'l Workshop on Design of Reliable Communication Networks (DRCN'98)*, Brugge, Belgium, May 17-20, 1998, paper O.33.
- [GSM95] W.D. Grover, J.B. Slevinsky, M.H. MacGregor, "Optimized design of ring-based survivable networks," *Canadian J. Elect. & Comp. Eng.*, Vol. 20, No. 3, 1995.
- [GuAh91] R. Guerin, H. Ahmadi, M. Naghshineh, "Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks", *IEEE JSAC*, November, 1991, pp. 968-81.
- [GuPr03] K.P. Gummadi, M.J. Pradeep, M.J. C.S.R. Murthy, "An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks," *IEEE/ACM Transactions on Networking*, Vol. 11 no. 1, Feb 2003, pp 81 -94.
- [HaKr95] D.D.Harms, M. Kraetl, C.J. Colbourn, J.S. Devitt, *Network Reliability: Experiments with a Symbolic Algebra Environment*, CRC Press, 1995.
- [HeBy94] M. Herzberg, S. Bye, "An optimal spare-capacity assignment model for survivable networks with hop limits," *Proc. IEEE GLOBECOM '94*, pp. 1601-1607, 1994.
- [HeBy95] M. Herzberg, S. Bye, A. Utano, "The hop-limit approach for spare-capacity assignment in survivable networks," *IEEE/ACM Trans. Networking*, vol.3, Dec. 1995, pp. 775-784.
- [Heit93] J. Heitkoetter, *The Hitch-Hiker's Guide to Evolutionary Computation* (1993), FAQ document in Internet newsgroup: comp.ai.genetic.
- [HoKr56] A.J. Hoffman, J. B. Kruskal, Integral boundary points of convex polyhedra, in *Linear Inequalities and Related Systems* (H. W. Kuhn, A. W. Tucker, eds.), Princeton University Press, Princeton, New Jersey, 1956, pp.223-246.
- [Holl75] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI. 1975.
- [HoMo02] P. Ho, H. T. Mouftah, "A novel distributed control protocol in dynamic wavelength-routed optical networks," *IEEE Communications Magazine*, vol. 40, no.11, Nov. 2002, pp.38-45.
- [HoMo02b] P. Ho, H. T. Mouftah, "A framework for service guaranteed shared protection in WDM mesh networks," *IEEE Communications Magazine*, vol. 40, no.2, Feb. 2002, pp.97-103..

- [Hu69] T.C. Hu, *Integer Programming and Network Flows*, Addison-Wesley, Reading, Mass., 1969, pp. 131-142.
- [HuCo02] H. Huang, J.A. Copeland, "A series of Hamiltonian cycle-based solutions to provide simple and scalable mesh optical network resilience," *IEEE Communications Magazine*, vol. 40, no.11, Nov. 2000, pp.47-51.
- [Huit95] C. Huitema, *Routing in the Internet*, Prentice-Hall PTR, 1995
- [IEEE95] *IEEE Communications Magazine*, Special Issue on "Dynamic Routing in Telecommunications Networks," vol.33, no.7, July 1995
- [Ing03] L. Ingber, *Adaptive Simulated Annealing* (web pages and software), Internet:<http://ideas.repec.org/p/lei/ingber/96as.html>, May 2003.
- [Iras96] R. R. Iraschko, *Path-Restorable Networks*, Ph.D. Dissertation, University of Alberta, Fall, 1996.
- [IrGr00] R.R. Iraschko, W.D. Grover, "A highly efficient path-restoration protocol for management of optical network transport integrity," *IEEE JSAC*, vol.18, no.5, May 2000, pp. 779- 793.
- [IrMa96] R. R. Iraschko, M. H. MacGregor, W. D. Grover, "Optimal Capacity Placement for Path Restoration in Mesh Survivable Networks", *Proc. IEEE ICC '96*, pp. 1 568 - 1 574, 1996.
- [IrMa98] R. R. Iraschko, M. H. MacGregor, W. D. Grover, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 3, pp. 325-336, June 1998.
- [ITU96] ITU-T, *Network Node interfaces for the Synchronous Digital Hierarchy (SDH)*, ITU-T Recommendation G.707, 1993, 1996.
- [Isel00] A. Iselt, "Terminal pair availability calculation algorithms for communication networks," *Proc. Design of Reliable Comm. Networks, DRCN 2000*, April 2000, Munich, Germany, pp. 233-247.
- [John75] D.B. Johnson, "Finding All the Elementary Circuits of a Directed Graph," *SIAM J. on Computing*, Vol. 4, 1975, pp. 77-84.
- [JoSa98] Y.T. Joens, B. Sales, H. De Neve, P. Van Mieghem, "Fault tolerance in a hierarchically structured dynamic routing environment," *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 1998, paper O17.
- [KaO99] R. Kawamura, H. Ohta, "Architectures for ATM network survivability and their field deployment," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 88-94.
- [KaSa94] R. Kawamura, K. Sato, I. Tokizawa, "Self-healing ATM Networks Based on Virtual Path Concept", *IEEE Journal on Selected Areas in Communications*. Vol. 12, No. 1, January 1994, pp. 120-127.
- [KaTo95] R. Kawamura, Ikuo Tokizawa, "Self-healing virtual path architecture in ATM networks," *IEEE Communications Magazine*, 9 1995.
- [Kar86] R.M. Karp, "Combinatorics, Complexity, and Randomness," *Communications of the ACM*, Vol. 29, No. 2, February 1986, pp. 98-117.
- [Kart00] S.V. Kartalopoulos, *Introduction to DWDM Technology: Data in a Rainbow*, SPIE and IEEE Press Publishers, 2000.
- [KeOl03] J. Kennington, E. Olinick, K. Lewis, A Orzynski, G. Spiride, "Robust Solutions for the DWDM Routing and Provisioning Problem: Models and Algorithms," *Optical Networks Magazine*, vol. 4, no. 2, March/April 2003.
- [KeLe98] J. L. Kennington, M.W. Lewis, *The Path Restoration Version of the Spare Capacity Allocation Problem with Modularity Restrictions: Models, Algorithms, and an Empirical Analysis*, Technical Report 98-CSE-13, Dept. Computer Science And Engineering, Southern Methodist University, Dallas, December 1998.
- [KeNa97] J.L. Kennington, V.S.S. Nair, M.H. Rahman, *Optimization Based Algorithms for Finding Minimal Cost Ring Covers in Survivable Networks*, Technical Report 97-CSE -12, Southern Methodist University, Dallas, Texas, August 1997, pp. 1-26.
- [Kers93] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill, New York, NY, (1993).
- [KiKo01] S. Kini, M. Kodialam, T. V. Laksham, S. Sengupta, C. Villamizar, "Shared backup Label Switched Path restoration," IETF Internet Draft, draft-kini-restoration-shared-backup-01.txt, May 2001.
- [KiGe83] S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, May 13 1983, pp. 671-680.

- [KKG91] A. Kershenbaum, P. Kermani, G. Grover, "MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing," *IEEE Trans. on Comm.*, No. 4, 1991, pp. 503 - 513.
- [Know89] T.W. Knowles, *Management Science: Building and Using Models*, Irwin, 1989.
- [KoCh90] Komine, H., Chujo, T., Ogura, T., Miyazaki, K., Soejima, T., "A distributed restoration algorithm for multiple-link and node failures of transport networks," *Proc. IEEE Globecom '90*, Dec. 1990, pp. 459 - 463.
- [KoGr03] A. Kodian, W.D. Grover, J. Slevinsky, D. Moore, "Ring-Mining to p -Cycles as a Target Architecture: Riding Demand Growth into Network Efficiency," to appear in *Proc. National Fiber Optic Engineers Conference (NFOEC 2003)*, Orlando, Florida, Sept. 7-11, 2003.
- [KoRe00] Kompella, K., Rekhter, Y., Banerjee, A., et al, "OSPF Extensions in Support of Generalized MPLS," Internet Draft, draft-kompella-ospf-extensions-00.txt, (work in progress), July 2000.
- [KoCh90] H. Komine, T. Chujo, T. Ogura, K. Miyazaki, T. Soejima, "A distributed restoration algorithm for multiple-link and node failures of transport networks," *Proc. IEEE GLOBECOM'90*, 1990, pp. 459-463.
- [KrPr00] G.P. Krishna, M.J. Pradeep, C.S.R. Murthy, "A segmented backup scheme for dependable real-time communication in multihop networks." *8th Int'l Workshop on Parallel and Distributed Real-Time Systems (WPRDRTS)*, Springer-Verlag vol. 1800, May 2000, pp. 768-684
- [Kuhn97] D.R. Kuhn, "Sources of failure in the public switched telephone network," *IEEE Computer Magazine*, vol.30, no.4, pp. 31-36.
- [LaAg98] A. Lardies, A. Aguilar, "Planning Methodology for SDH + Optical Networks", *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 1998.
- [LaMi01] Lang, J., Mitra, K., Drake, J., Kompella, K., Rekhter, Y., Berger, L., Saha, D., Basak, D., Sandick, H., Zinin, A, "Link Management Protocol (LMP)", Internet Draft, draft-ietf-ccamp-lmp-02.txt, (work in progress), July 2001.
- [LaTa01] P. Laborczy, J. Tapolcai, P-H. Ho, T. Cinkler, A. Recski, H.T. Mouftah, "Algorithms for Asymmetrically Weighted Pair of Disjoint Paths in Survivable Networks," *Proc. Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, October 2001. pp. 220-227.
- [Level3] Level3 Communications, Network Maps, www.level3.com/userimages/dotcom/en_US/images/dotcom_us.gif, May 2003.
- [LiTi01] Y. Liu, D. Tipper, P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *Proceeding of IEEE INFOCOM*, pages 699-708, Anchorage, AL, April 24-28 2001
- [LiTi01b] Y. Liu, D. Tipper, "Spare Capacity Allocation for Non-Linear Cost and Failure-Dependent Path Restoration," *Proceedings Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, October 2001, pp.243-250.
- [Lee99] C.Y. Lee, *Heuristic Methods for Sub-Graph Topology Enhancement in Ring-Based Transport Network Design*, M.Sc. Dissertation, University of Alberta, Canada, Fall 1999.
- [LeGr99] C.Y. Lee, W.D. Grover, G.D. Morley, "Heuristic Methods for the 'Span Elimination' Problem in Ring-Based Transport Network Design," *Proc. IEEE Canadian Conf. on Elec. and Comp. Eng. '99*, Edmonton, May 1999, pp. 232-237.
- [LeGr02] D. Leung, W. D. Grover, "Comparative ability of span-restorable and path-protected network designs to withstand uncertainty in the demand forecast," *Proc. 18th National Fiber Optic Engineers Conference (NFOEC 2002)*, Dallas, Sept. 15-19, 2002., pp.1450-1461.
- [LeSo98] H. Lee, H-G. Song, "Integrated VP/VC rerouting model for multi-layer restoration in ATM networks," *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 1998, paper O36.
- [Lin98] L. Y. Lin, E. L. Goldstein, R. W. Tkach, "Free-space micromachined optical switches with sub-millisecond switching time for large-scale optical cross-connects," *IEEE Photonics Technology Letters*, April 1998, pp. 525-528.
- [Liu01] Yu Liu, *Spare Capacity Allocation Model, Analysis and Algorithm*, Ph.D. Dissertation, School of Information Sciences, University of Pittsburgh, 2001
- [LiYa02] G. Li, J. Yates, D. Wang, C. Kalmanek, "Control Plane Design for Reliable Optical Networks," *IEEE Communications Magazine*, vol. 40, no. 2, Feb. 2002, pp.90-96.
- [LuMe00] S. Lumetta, M. Médard, Y.-C. Tseng, "Capacity versus robustness: a trade-off for link restoration in mesh networks," *Journal of*

Lightwave Technology, vol. 18, no. 12, Dec. 2000, pp.1765-1775.

[MacG91] M. MacGregor, *The Self Traffic-Engineering Network*, Ph.D. Thesis, University of Alberta, 1991.

[MaCo03] S. DeMaesschalek, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, J. Derkacz, "Pan-European Optical Transport Networks: An Availability-based Comparison," *Photonic Network Communications*, vol.3, no.5, May 2003, pp. 203-225.

[MaDe76] P. Matei, N. Deo, "On algorithms for enumerating all circuits of a graph," *SIAM J. Comput.*, vol. 5, no. 1, Mar. 1976, pp. 90-99.

[MaGr94] M.H. MacGregor, W.D. Grover, "Optimized k -shortest paths algorithm for facility restoration," *Software - Practise & Experience*, Vol. 24(9), Wiley & Sons, Sept. 1994, pp.823-834.

[MaGr95] M.H. MacGregor, W.D. Grover, "Investigation of a cut-tree approach to network restoration from node loss," *Proceedings IEEE ICC '95*, June 18-22, Seattle, 1995, vol.3, pp.1530-1535.

[MaGr97]M. H. MacGregor, W. D. Grover, K. Ryhorchuk,"Optimal Spare Capacity Preconfiguration for Faster Restoration of Mesh Networks," *Journ. of Network and Systems Management*, Vol. 5, No. 2, 1997, pp. 159-171.

[MaGr97b] M.H. MacGregor, W.D. Grover, "Distributed Partial-Express Routing of Broadband Transport Network Demands," *IEEE / ACM Transactions on Networking*, vol.5, no.6, Dec. 1997, pp. 981-988.

[MaSh00] Makam, V., Sharma, V., Huang, C., Owens, K., Mack-Crane, B., et al,"A Framework for MPLS-based Recovery", Work in Progress, Internet Draft <draft-ietf-mpls-recovery-frmrwk-00.txt>, September 2000.

[McBo98] A. McGuire, P. Bonenfant,"Standards: The Blueprint for Optical Networking," *IEEE Communications Magazine*, Vol. 32, No. 2, 1998, pp. 68-78.

[McDo94] J.C. McDonald, "Public Network Integrity - Avoiding a crisis in trust", *IEEE Journal on Selected Areas in Communications*, vol.12, no.1, January 1994, pp. 5-12.

[MeBa02] M. Medard, R.A. Barry, S.G. Finn, W. He, S.S.Lumetta,"Generalized loop-back recovery in optical mesh networks," *IEEE Trans. Networking*, vol.10, no. 1, Feb.2002, pp. 153-164.

[Mino81] M. Minoux, "Optimum synthesis of a network with non-simultaneous multicommodity flow requirements," *Annals of Discrete Mathematics*, vol. 11, pp. 269-277.

[MiSa98] Y. Miyao, H. Saito,"Optimal Design and Evaluation of Survivable WDM Transport Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 7, pp. 1190-1198, September 1998.

[MoGr01] D. Morley, W.D. Grover, "Tabu Search Optimization of Optical Ring Transport Networks," *Proceedings IEEE Globecom 2001*, San Antonio, Texas, USA, Nov. 25-29, 2001, 5 pages.

[MoGr99] D. Morley, W.D. Grover, "Mathematical Programming Approaches to Optical Ring Network Design," *Proc. 3rd Can. Conf. On Broadband Research (CCBR'99)* Nov. 7, 9, Ottawa, 1999.

[MoLi01] E. Modiano, P.J.Lin, "Traffic Grooming in WDM Networks," *IEEE Communications Magazine*, July 2001, pp. 124-129.

[Moy98] J.T. Moy, *OSPF: Anatomy of an Internet Protocol*, Addison-Wesley, 1998.

[Morl01] D. Morley, *Analysis and Design of Ring-based Transport Networks*, Ph.D. Dissertation, University of Alberta, Canada, Spring 2001.

[MoSh89] C.L. Monma, D.F. Shallcross, "Methods for Designing Communication Networks with Certain Two-Connected Survivability Constraints," *Operations Research*, vol. 37, No. 4, July - August 1989, pg. 531 - 541.

[Mukh00] B. Mukherjee, "WDM optical communication networks: progress and challenges," *IEEE Journal Selected Areas in Communications*, vol.18, no.10, Oct. 2000, pp. 1810-1824

[OCon91] P. O'Connor, *Practical Reliability Engineering*, 3rd Edition, John Wiley & Sons, 1991.

[OePu98] M. Oellrich, J. von Puttkamer, "Reliable links in transmission networks," *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 1998, paper O22.

[OkNo00] E. Oki, N. Matsuura, K. Shiimoto, N. Yamanaka,"A disjoint path selection scheme with shared risk link groups in GMPLS

networks," *IEEE Communications Letters*, Volume 6, Issue 9, Sept. 2002, pp 406-408.

[Onvu94] R.O. Onvural, *Asynchronous Transfer Mode Networks Performance Issues*, Artech House, 1994.

[OPN02] OPNET, "Flow Analysis Module," http://www.opnet.com/products/modules/flow_analysis.html, 27 September 2002.

[PeEl96] H.G. Perros, K.M.Elsayed, "Call Admission Control Schemes: A Review", *IEEE Communication Magazine*, November, 1996, pp. 82-91.

[Perl92] R. Perlman, *Interconnections Bridges and Routers*, Addison-Wesley Professional Computing series, 1992.

[Petr85] H. Petrosky, *To Engineer is Human*, St. Martins Press, New York, 1985

[PiDe00] M. Pickavet, P. Demeester, "A Zoom-In approach to design SDH mesh-restorable networks," *Journal of Heuristics Special Edition on Heuristic Approaches for Telecommunications Network Management, Planning and Expansion*, vol.6, no.1, April 2000, pp. 103- 126.

[PiGa97] M. Pióro, P. Gajowniczek, "Solving Multicommodity Integral Flow Problems by Simulated Allocation," *Telecommunication Systems*, Baltzer Science Publishers, Vol.7, Nos.1-3, pp.17-28,1997.

[Pioro97] Michal Pióro, "Robust design problems in telecommunication networks," *Proceedings Fifteenth International Teletraffic Conference (ITC 15)*, Washington DC, June 1997.

[RaAg90] S. Rai, D. Agrawal, *Distributed Computing Network Reliability*, IEEE Computer Society Press, 1990.

[RaDa99] B. Ramamurthy, D. Datta, H. Feng, J. P. Heritage, B. Mukherjee, "Transparent vs. opaque vs. translucent wavelength-routed optical networks," *Proceedings of OFC'99*, vol. 1, pp. 59-61.

[RaLa03] R. Ramamurthy, J-F. Labourdette, A. Akyamac, S. Chaudhuri, "Limiting sharing on protection channels in mesh optical networks," *Proc. OSA Optical Fiber Communications Conference (OFC 2003)*, Atlanta, March 2003, vol.1, paper Tu3, pp.204-205.

[Rama01] B. Ramamurthy, *Design of Optical WDM Networks: LAN, MAN and WAN Architectures*, Kluwer Academic, 2001.

[RaSa97] R. Ramaswami, G. H. Sasaki, "Multi-wavelength optical networks with limited wavelength conversion," in *Proc. of IEEE Infocom '97*, 1997, Vol. 2, pp. 489-498.

[RaSi95] R. Ramaswami, K.N. Sivarajan, "Routing and Wavelength Assignment in All-Optical Networks," *IEEE/ACM Trans. on Networking*, Vol. 3, 1995, pp. 489-500.

[RaSi98] R. Ramaswami, K.N. Sivarajan, *Optical Networks: A Practical Perspective*, Academic Press, 1998.

[Rawl91] G. J. E. Rawlins (editor), *Introduction to Foundations of Genetic Algorithms*, San Mateo, CA, Morgan Kaufmann Publishers, 1991, pp. 1-10.

[RiHa01] C. C. Ribeiro, P. Hansen (Editors), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers; 1st edition, October 2001.

[Robe99] T.G. Robertazzi, *Planning Telecommunication Networks*, IEEE Press, 1999, pp. 66-72.

[SaAl98] J.M. Santos, T. Almeida, P. Fonseca, "The use of distributed restoration on WDM networks," *Proc. Design of Reliable Commun. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 1998, paper O10.

[SaGr03] A. Sack, W. D. Grover, "Hamiltonian p -Cycles for Fiber-Level Protection in Homogeneous and Semi-Homogeneous Optical Networks," to appear in *IEEE Network, Special Issue on Protection, Restoration, and Disaster Recovery*, (scheduled for Nov/Dec. 2003).

[SaOk92] H. Sakauchi, Y. Okanou, S. Hasegawa, "Spare-channel design schemes for self-healing networks," *IEICE Trans. Comm.*, vol. E75-B, no.7, pp. 624-633, July 1992.

[SaNi90] H. Sakauchi, Y. Nishimura, S. Hasegawa, "A self-healing network with an economical spare channel assignment," *Proc. IEEE GLOBECOM'90*, 1990, pp. 438-443.

[SaMu02] C.V. Saradhi, C.S.R. Murthy, "Dynamic establishment of segmented protection paths in single and multi-fiber WDM mesh networks," *Proc. of Opticomm 2002*, Boston, July 2002, SPIE vol. 4874, pp. 211-222.

[Sato96] Ken-ichi Sato, *Advances in Transport Network Technologies: Photonic networks, ATM, and SDH*, Artech House, Norwood MA, 1996.

- [ScSc03] D.A. Schupke, M.C. Scheffel, W.D. Grover, "Configuration of p -Cycles in WDM Networks with Partial Wavelength Conversion," to appear in *Photonic Network Communications*, (accepted June 2003).
- [Schal01] J. Schallenburg, "Is 50 ms Restoration Necessary?" Presentation to the *IEEE Bandwidth Management workshop IX*, Montebello, Quebec, June 2001.
- [ScGr02] D.A. Schupke, C. Gruber, A. Autenrieth, "Optimal Configuration of p -Cycles in WDM Networks," *Proc. of IEEE ICC*, 2002.
- [SeRe92] M. Sexton, A. Reid, *Transmission Networking: SONET and the Synchronous Digital Hierarchy*, Artech House, Norwood MA, 1992.
- [SeYa01] P. Sebos, J. Yates, G. Hjálmsón, A. Greenberg, "Auto-discovery of Shared Risk Link Groups," *Optical Fiber Commun. Conf.*, March 2001.
- [ShBo00] G. Shen, S. K. Bose, T. H. Cheng, C. Lu, T. K. Chai, "Efficient wavelength assignments for lightpaths in WDM optical networks with/without wavelength conversion," *Photonic Network Communications*, vol. 2, no. 4, pp. 349-360, November 2000.
- [ShGr03] G. Shen, W.D. Grover, "Capacity Requirements for Node Failure Recovery with Dynamic Path Restoration," *Proc. OSA OFC 2003*, Atlanta, March 28, 2003, vol.2, paper FQ3, pp.775-777.
- [ShGr03b] G. Shen, W.D. Grover, "Extending the p -Cycle Concept to Path-Segment Protection," *Proc. IEEE International Conference on Communications (ICC 2003)*, Anchorage, AK, USA, May 11-15, 2003, Session ON3.
- [ShGr03c] G. Shen, W.D. Grover, "Exploiting Forcer Structure to Serve Uncertain Demands and Minimize Redundancy of p -Cycle Networks," to appear in *Proc. OptiComm 2003*, Dallas, Texas, October 13-17, 2003.
- [Shei91] D.R. Shier, *Network Reliability and Algebraic Structures*, Clarendon Press, Oxford, 1991
- [Shei76] D.R. Sheir, "Iterative methods for determining the k-shortest paths in a network," *Networks*, vol.6, 1976, pp. 205-229.
- [Sill96] C.A. Siller, M. Shafi (editors), *SONET/SDH: A Sourcebook of Synchronous Networking*, IEEE Press, 1996, pp. 211-217.
- [SiSu00] K.M. Sivalingam, S. Subramaniam (editors), *Optical WDM Networks: Principles and Practice*, Kluwer Academic, (2nd Printing) 2001.
- [Sosn94] Sosnosky, J., "Service applications for SONET DCS distributed restoration," *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 1, pp. 59-68, January 1994.
- [SrSo02] M. Sridharan, A.K. Somani, "Design for upgradability in mesh-restorable optical networks," *Optical Networks Magazine*, vol. 3, no. 3, May/June 2002, pp. 77-87.
- [StGr00b] D. Stamatelakis, W.D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles (" p -cycles")," *IEEE Transactions on Communications*, Vol. 48, No. 8, pp. 1262-1265.
- [Stam97] D. Stamatelakis, *Theory and Algorithms for Pre-configuration of Spare Capacity in Mesh Restorable Networks*, M. Sc. Thesis, University of Alberta, Spring 1997.
- [Stav01] A.A. Stavdas (editor), *New Trends in Optical Network Design and Modeling*, Kluwer Academic, 2001.
- [StBa99] T.E. Stern, K. Bala, *Multiwavelength Optical Networks: A Layered Approach*, Addison-Wesley, 1999.
- [StCh01] J. Strand, A. Chiu, R. Tkach, "Issues for routing in the optical layer," *IEEE Communications Magazine*, February 2001, pp. 81-87.
- [Stev94] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley Professional Computing Series, 1994
- [StDo00] J. Strand, R. Doverspike, G. Li, "Importance Of Wavelength Conversion In An Optical Network," *Optical Networks Magazine*, May/June 2000, Vol 2 (3), pp. 33-44.
- [StGr98] D. Stamatelakis, W.D. Grover, "OPNET Simulation of Self-organizing Restorable SONET Mesh Transport Networks," *Proc. OPNETWORKS '98 (CD-ROM)*, Wash. D.C., April 1998, paper 04.
- [StGr99] D. Stamatelakis, W.D. Grover, *Network Restorability Design Using Pre-configured Trees, Cycles, and Mixtures of Pattern Types*, TR Labs Technical Report TR-1999-05, Issue 1.0, October 30 2000.

- [StGr99b] D. Stamatelakis, W.D. Grover, "Rapid Restoration of Internet Protocol Networks using Pre-configured Protection Cycles," *Proc. 3rd Can. Conf. Broadband Research (CCBR'99)*, Nov. 7-9, Ottawa, 1999, pp.33-44.
- [StGr00] D. Stamatelakis, W. D. Grover, "IP Layer Restoration and Network Planning Based on Virtual Protection Cycles," *IEEE JSAC Special Issue on Protocols and Architectures for Next Generation Optical WDM Networks*, vol.18, no.10, October, 2000, pp. 1938 - 1949.
- [SuAz96] S. Subramaniam, M. Azizoglu, A. K. Somani,"All-optical networks with sparse wavelength conversion," *IEEE Transaction on Networking*, vol. 4, no. 4, pp. 544-557, August 1996.
- [SuAz98] S. Subramaniam, M. Azizoglu, A. K. Somani,"On the optimal placement of wavelength converters in wavelength routed networks," *Proceedings of IEEE INFOCOM'98*, pp. 902-909.
- [Surb74] J.W. Surballe,"Disjoint Paths in a Network," *Networks*, vol.4, 1974, pp. 125-145.
- [SuTa84] J.W. Surballe, R.E. Tarjan,"A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, vol.14, 1984, pp. 325-336.
- [T1A193] T1A1.2 Working Group on Network Survivability Performance, Report No. 24A *Technical Report on Network Survivability Performance*, November 1993.
- [Taha87] H.A. Taha, *Operations Research: An Introduction*, Fourth Ed., MacMillian Publishing, New York, 1987.
- [Telc00] GR-253-CORE, "Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria," Telcordia Technologies, Issue 3, September 2000
- [ThSw92] K. Thulasiraman, M.N.S. Swamy, *Graphs: Theory and Algorithms*, John Wiley & Sons, 1992
- [ToNe94] M. To, P. Neusy,"Unavailability analysis of long-haul networks," *IEEE J. on Sel. Areas in Comm.*, vol. 12, no. 1, January 1994, pp100-109.
- [Topk88] D. M. Topkis, "A k -shortest path algorithm for adaptive routing in communications networks,"*IEEE Trans. Comm.*, vol. 36, no.7, July 1988, pp. 855-859.
- [Vau96] D. Vaughan, *The Challenger Launch Decision*, University of Chicago Press, 1996.
- [VeGr93] B. D. Venables, W. Grover, M. H. MacGregor,"Two strategies for spare capacity placement (SCP) in mesh restorable networks," *Proc. IEEE ICC'93*, Geneva, May 1993, pp. 267-271.
- [Vena92] B.D. Venables, *Algorithms for the Spare Capacity Design of Mesh Restorable Networks*, M.Sc. Thesis, University of Alberta, Edmonton. (1992)
- [VePo02] Presentation by A. J. Vernon, J. D. Portier,"Protection of Optical Channels in All-Optical Networks," *18th Annual National Fiber Optic Engineers Conference (NFOEC 2002)*, pp. 1695-1706, Dallas, TX, Sept. 2002.
- [Wang98] Y. Wang, *Modeling and Solving Single and Multiple Facility Restoration Problems*, Ph.D. dissertation, Sloan School of Management, MIT, June 1998., pp.32-33.
- [WoCh99] E.W.M. Wong, A.K.M. Chan, T.-S.P. Yum,"A Taxonomy of Rerouting in Circuit-Switched Networks," *IEEE Communications Magazine*, Vol. 37, No. 11, 1999, pp. 116-122.
- [Wins94] W. L. Winston, *Operations Research: Applications and Algorithms*, 3rd ed. (Duxbury, 1994) pp. 809-815.
- [WiSh97] M.Willebeek-LeMair, P. Shahabuddin,"Approximating dependability measures of computer networks: An FDDI case study," *IEEE/ACM Transactions on Networking*, vol.5, no.2., April 1997, pp. 311-327.
- [Wols98] L.A. Wolsey, *Integer Programming*, John Wiley & Sons, 1998.
- [Wyna01] C. Wynants, *Network Synthesis Problems (Combinatorial Optimization Series)*, Kluwer Academic Publishers 2001.
- [Wu92] T.-H. Wu, *Fiber Network Service Survivability*, Artech House, 1992
- [Wu99] T.-H. Wu, "Emerging Technologies for Fiber Network Survivability," *IEEE Communications Magazine*, Vol. 33, No. 2, 1999, pp. 70-75.
- [WuNo97] T.-H. Wu, N. Yoshika, *ATM Transport and Network Integrity*, Academic Press, 1997.

[WuKo93] T.-H. Wu, H. Kobrinski, D. Ghosal, T.V. Lakshman, "A service restoration time study for distributed control SONET digital cross-connect system self-healing networks," *Proc. ICC'93*, pp. 893-899, IEEE, 1993.

[XiMa99] Y. Xiong; L.G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks" *IEEE/ACM Transactions on Networking*, Volume: 7 Issue: 1, Feb. 1999, pp. 98 -110.

[Yage71] B. Yaged, Minimum cost routing for static network models, *Networks*, vol.1, 1971, pp. 139-172.

[YaHa88] C. Han Yang, S. Hasegawa, "FITNESS: A failure immunization technology for network service survivability," *Proc. IEEE GLOBECOM'88*, 1988, pp. 1549-1544.

[YaKa98] T. Yahara, R. Kawamura, S.Ohta, "Multi-reliability self-healing scheme that guarantees minimum cell rate," *Design of Reliable Comm. Networks (DRCN'98)*, IMEC, U.Ghent, Brugge, Belgium, May 17-20, 1998, paper O14.

[YaLa96] J. Yates, J. Lacey, D. Everitt, M. Summerfield, "Limited-range wavelength translation in all-optical networks," *Proceedings of IEEE INFOCOM'96*, pp. 954-961.

[ZaJu00] H. Zang, J.P. Jue, B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Optical Networks Magazine*, vol.1, Jan. 2000, pp. 47-60.

[ZeCa96] E. W. Zegura, K.L. Calvert, S. Bhattacharjee, "How to model an Internetwork," *IEEE INFOCOM, 1996*, vol. 2, pg. 594 - 602.

[ZeCa97] E.W. Zegura, K.L. Calvert, M.J. Donahoo, "A Quantitative Comparison of Graph-Based models for Internet Topology," *IEEE/ACM Transactions on Networking*, 1997, vol. 5, no. 6, pg. 770 - 783.

[ZhBi03] K. Zhu, B. Mukherjee, "A Review of Traffic Grooming in WDM Optical Networks: Architectures and Challenges," *Optical Networks Magazine*, vol. 4, no. 2, March/April 2003.

[Zhen97] Y. Zheng, *Spare Capacity Design of ATM VP-based Restorable Networks*, M.Sc. Thesis, University of Alberta, Fall 1997.

[ZhGr97] Y. Zheng, W.D. Grover, M. MacGregor, "Broadband Network Design with Controlled Exploitation of Flow Convergence Overloads in ATM VP-based Restoration", *Proceedings of the Canadian Conference on Broadband Research (CCBR '97)*, April 16-17, 1997, Ottawa, Canada.

[ZhGr97b] Y. Zheng, W.D. Grover, M.H. MacGregor, "Dependence of Network Capacity Requirements on the allowable flow convergence overloads in ATM Backup VP Restoration," *Electronics Letters*, vol. 33, no.5, Feb. 27, 1997, pp 362-363.

[ZhMu02] K. Zhu, B. Mukherjee, "Traffic grooming in an optical WDM mesh network," *IEEE JSAC*, vol. 20, no.1, Jan. 2002, pp. 122-133.

[Zorp89] G. Zorpette, "Keeping the phone lines open," *IEEE Spectrum*, June 1989, pp.32-36.

[\[Team LiB \]](#)

◀ PREVIOUS

[Team LiB]

[SYMBOL]

[Team LiB]

[\[Team LiB \]](#)

[\[SYMBOL\]](#)

[_Ref517594407](#)

11427

RF

[24. S. Subramaniam, M. Azizoglu, and A. K. Somani, All-optical networks with sparse wavelength conversion](#)

19227

Reference

[\[7\] Robert S. Cahn, Wide Area Network Design - Concepts and Tools for Optimization, Morgan Kaufman P](#)

20834

RF

[13. W. D. Grover, J. Doucette, Design of a Meta-Mesh of Chain Sub-Networks: Enhancing the Attractiveness of](#)

23308

RF

[16. Ryutaro Kawamura and Ikuo Tokizawa. Self-healing virtual path architecture in ATM networks. IEE](#)

26509

RF

[3. B. Gavish, Topological design of telecommunication networks - Local access design methods, Annals](#)

28317

Reference

[\[13\] B. Gendron, T.G. Crainic, A. Frangioni, Multicommodity capacitated network design, in: Telecomm](#)

56550

Reference

[\[12\] B. Yaged, Minimum cost routing for static network models, Networks, vol.1, 1971, pp. 139-172.](#)

57953

RF

[37. D. Morley, Analysis and Design of Ring-based Transport Networks, Ph.D. Dissertation, University of Alb](#)

61036

RF

[21. \[28\] D.A. Dunn, W.D. Grover, M.H. MacGregor, A comparison of k-shortest paths and maximum flow methods](#)

65174

Reference

[\[37\] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows : Theory, Algorithms and Applications, Pre](#)

65492

RF

[19. B. Ramamurthy, D. Datta, H. Feng, J. P. Heritage and B. Mukherjee, Transparent vs. opaque vs. transp](#)

73462

Reference

[\[38\] Y. Wang, Modelling and solving single and multiple facility restoration problems, Ph.D. dissert](#)

78855

RF

[35. J. L. Kennington, M.W. Lewis, The Path Restoration Version of the Spare Capacity Allocation Prob](#)

86434

RF

[34. Y. Liu, D. Tipper, and P. Siripongwutikorn. Approximating optimal spare capacity allocation by s](#)

89761

Reference

[\[17\] Venables, B.D. Algorithms for the Spare Capacity Design of Mesh Restorable Networks. M.Sc. Thes](#)

90193

RF

[9. T.G. Robertazzi, Planning Telecommunication Networks, IEEE Press, 1999, pp. 66-72.](#)

92597

RF

[5. Makam, V., Sharma, V., Huang, C., Owens, K., Mack-Crane, B., et al, AFrameworkforMPLS-basedRecov](#)

95415

RF

[11. J. Doucette, W. D. Grover, ComparisonofMeshProtectionandRestorationSchemesandtheDependencyon](#)

96996

RF

[28. Christelle Wynants, Network Synthesis Problems \(Combinatorial Optimization Series\), Kluwer Acade](#)

[\[Team LiB \]](#)

Brought to You by



Like the book? Buy it!