Martin Grötschel
László Lovász
Alexander Schrijver

Geometric Algorithms and Combinatorial Optimization

Springer-Verlag Berlin Heidelberg NewYork
London Paris Tokyo

Martin Grötschel
Institute of Mathematics
University of Augsburg
Memminger Straße 6
D-8900 Augsburg
Fed. Rep. of Germany

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest
Múzeum krt. 6–8
Hungary H-1088

Alexander Schrijver
Department of Econometrics
Tilburg University
P.O. Box 90153
NL-5000 LE Tilburg
The Netherlands

1980 Mathematics Subject Classification (1985 Revision):
primary 05-02, 11Hxx, 52-02, 90Cxx; secondary 05Cxx, 11H06, 11H55, 11J13, 52A43, 68Q25, 90C05, 90C10, 90C25, 90C27

ISBN 3-540-13624-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-13624-X Springer-Verlag New York Berlin Heidelberg

With 23 Figures

Library of Congress Cataloging-in-Publication Data
Grötschel, Martin.
Geometric algorithms and combinatorial optimization.
(Algorithms and combinatorics ; 2)
Bibliography: p. Includes indexes.
1. Combinatorial geometry. 2. Geometry of numbers. 3. Mathematical optimization. 4. Programming (Mathematics) I. Lovász, László, 1948–. II. Schrijver, A. III. Title. IV. Series. QA167.G76 1988 511'.6 87-36923
ISBN 0-387-13624-X (U.S.)

Printed in Germany

Media conversion, printing and binding: Universitätsdruckerei H. Stürtz AG, Würzburg
2141/3140-543210

Preface

Historically, there is a close connection between geometry and optimization. This is illustrated by methods like the gradient method and the simplex method, which are associated with clear geometric pictures. In combinatorial optimization, however, many of the strongest and most frequently used algorithms are based on the discrete structure of the problems: the greedy algorithm, shortest path and alternating path methods, branch-and-bound, etc. In the last several years geometric methods, in particular polyhedral combinatorics, have played a more and more profound role in combinatorial optimization as well.

Our book discusses two recent geometric algorithms that have turned out to have particularly interesting consequences in combinatorial optimization, at least from a theoretical point of view. These algorithms are able to utilize the rich body of results in polyhedral combinatorics.

The first of these algorithms is the **ellipsoid method**, developed for nonlinear programming by N. Z. Shor, D. B. Yudin, and A. S. Nemirovskiĭ. It was a great surprise when L. G. Khachiyan showed that this method can be adapted to solve linear programs in polynomial time, thus solving an important open theoretical problem. While the ellipsoid method has not proved to be competitive with the simplex method in practice, it does have some features which make it particularly suited for the purposes of combinatorial optimization.

The second algorithm we discuss finds its roots in the classical "geometry of numbers", developed by Minkowski. This method has had traditionally deep applications in number theory, in particular in diophantine approximation. Methods from the geometry of numbers were introduced in integer programming by H. W. Lenstra. An important element of his technique, called **basis reduction**, goes in fact back to Hermite. An efficient version of basis reduction yields a polynomial time algorithm useful not only in combinatorial optimization, but also in fields like number theory, algebra, and cryptography.

A combination of these two methods results in a powerful tool for combinatorial optimization. It yields a theoretical framework in which the polynomial time solvability of a large number of combinatorial optimization problems can be shown quite easily. It establishes the algorithmic equivalence of problems which are "dual" in various senses.

Being this general, this method cannot be expected to give running times comparable with special-purpose algorithms. Our policy in this book is, therefore, not to attempt to obtain the best possible running times; rather, it is to derive just the polynomial time solvability of the problems as quickly and painlessly as

possible. Thus, our results are best conceived as "almost pure" existence results for polynomial time algorithms for certain problems and classes of problems.

Nevertheless, we could not get around quite a number of tedious technical details. We did try to outline the essential ideas in certain sections, which should give an outline of the underlying geometric and combinatorial ideas. Those sections which contain the technical details are marked by an asterisk in the list of contents. We therefore recommend, for a first reading, to skip these sections.

The central result proved and applied in this book is, roughly, the following. If K is a convex set, and if we can decide in polynomial time whether a given vector belongs to K, then we can optimize any linear objective function over K in polynomial time. This assertion is, however, not valid without a number of conditions and restrictions, and even to state these we have to go through many technical details. The most important of these is that the optimization can be carried out in an approximate sense only (as small compensation, we only need to test for membership in K in an approximate sense).

Due to the rather wide spread of topics and methods treated in this book, it seems worth while to outline its structure here.

Chapters 0 and 1 contain mathematical preliminaries. Of these, Chapter 1 discusses some non-standard material on the complexity of problems, efficiency of algorithms and the notion of oracles.

The main result, and its many versions and ramifications, are obtained by the ellipsoid method. Chapter 2 develops the framework necessary for the formulation of algorithmic problems on convex sets and the design of algorithms to solve these. A list of the main problems introduced in Chapter 2 can be found on the inner side of the back cover. Chapter 3 contains the description of (two versions of) the ellipsoid method. The statement of what exactly is achieved by this method is rather complicated, and the applications and specializations collected in Chapter 4 are, perhaps, more interesting. These range from the main result mentioned above to results about computing the diameter, width, volume, and other geometric parameters of convex sets. All these algorithms provide, however, only approximations.

Polyhedra encountered in combinatorial optimization have, typically, vertices with small integral entries and facets with small integral coefficients. For such polyhedra, the optimization problem (and many other algorithmic problems) can be solved in the exact sense, by rounding an approximate solution appropriately. While for many applications a standard rounding to some number of digits is sufficient, to obtain results in full generality we will have to use the sophisticated rounding technique of diophantine approximation. The basis reduction algorithm for lattices, which is the main ingredient of this technique, is treated in Chapter 5, along with several applications. Chapter 6 contains the main applications of diophantine approximation techniques. Besides strong versions of the main result, somewhat different combinations of the ellipsoid method with basis reduction give the strongly polynomial time solvability of several combinatorial optimization problems, and the polynomial time solvability of integer linear programming in fixed dimension, remarkable results of É. Tardos and H. W. Lenstra, respectively.

Chapters 7 to 10 contain the applications of the results obtained in the previous chapters to combinatorial optimization. Chapter 7 is an easy-to-read introduction to these applications. In Chapter 8 we give an in-depth survey of combinatorial optimization problems solvable in polynomial time with the methods of Chapter 6. Chapters 9 and 10 treat two specific areas where the ellipsoid method has resolved important algorithmic questions that so far have resisted direct combinatorial approaches: perfect graphs and submodular functions.

We are grateful to several colleagues for many discussions on the topic and text of this book, in particular to Bob Bixby, András Frank, Michael Jünger, Gerhard Reinelt, Éva Tardos, Klaus Truemper, Yoshiko Wakabayashi, and Zaw Win. We mention at this point that the technique of applying the ellipsoid method to combinatorial optimization problems was also discovered by R. M. Karp, C. H. Papadimitriou, M. W. Padberg, and M. R. Rao.

We have worked on this book over a long period at various institutions. We acknowledge, in particular, the support of the joint research project of the German Research Association (DFG) and the Hungarian Academy of Sciences (MTA), the Universities of Amsterdam, Augsburg, Bonn, Szeged, and Tilburg, Cornell University (Ithaca), Eötvös Loránd University (Budapest), and the Mathematical Centre (Amsterdam).

Our special thanks are due to Frau Theodora Konnerth for the efficient and careful typing and patient retyping of the text in TEX.

March 1987

Martin Grötschel
László Lovász
Alexander Schrijver

Table of Contents

The sections and chapters marked with * are technical. We recommend that the reader skip these on the first reading.

Five Basic Problems (see inner side of the back cover)

Chapter 0

Mathematical Preliminaries

This chapter summarizes mathematical background material from linear algebra, linear programming, and graph theory used in this book. We expect the reader to be familiar with the concepts treated here. We do not recommend to go thoroughly through all the definitions and results listed in the sequel – they are mainly meant for reference.

0.1 Linear Algebra and Linear Programming

In this section we survey notions and well-known facts from linear algebra, linear programming, polyhedral theory, and related fields that will be employed frequently in subsequent chapters. We have also included a number of useful inequalities and estimates. The material covered here is standard and can be found in several textbooks. As references for linear algebra we mention FADDEEV and FADDEEVA (1963), GANTMACHER (1959), LANCASTER and TISMENETSKY (1985), MARCUS and MINC (1964), STRANG (1980). For information on linear programming and polyhedral theory see for instance CHVÁTAL (1983), DANTZIG (1963), GASS (1984), GRÜNBAUM (1967), ROCKAFELLAR (1970), SCHRIJVER (1986), STOER and WITZGALL (1970).

Basic Notation

By $\mathbb{R}$ ($\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{C}$) we denote the set of real (rational, integral, natural, complex) numbers. The set $\mathbb{N}$ of natural numbers does not contain zero. $\mathbb{R}_+$ ($\mathbb{Q}_+$, $\mathbb{Z}_+$) denotes the nonnegative real (rational, integral) numbers. For $n \in \mathbb{N}$, the symbol $\mathbb{R}^n$ ($\mathbb{Q}^n$, $\mathbb{Z}^n$, $\mathbb{N}^n$, $\mathbb{C}^n$) denotes the set of vectors with n components (or n-tuples or n-vectors) with entries in $\mathbb{R}$ ($\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{C}$). If E and R are sets, then R^E is the set of mappings of E to R. If E is finite, it is very convenient to consider the elements of R^E as $|E|$-vectors where each component of a vector $x \in R^E$ is indexed by an element of E, i. e., $x = (x_e)_{e \in E}$. For $F \subseteq E$, the vector $\chi^F \in \mathbb{R}^E$ defined by $\chi^F_e = 1$ if $e \in F$ and $\chi^F_e = 0$ if $e \in E \setminus F$ is called the **incidence vector** of F.

Addition of vectors and multiplication of vectors with scalars are as usual. With these operations, $\mathbb{R}^n$ and $\mathbb{Q}^n$ are vector spaces over the fields $\mathbb{R}$ and $\mathbb{Q}$, respectively, while $\mathbb{Z}^n$ is a module over the ring $\mathbb{Z}$. A vector is always considered as a **column vector**, unless otherwise stated. The superscript "T"

denotes **transposition**. So, for $x \in \mathbb{R}^n$, x^T is a row vector, unless otherwise stated. $\mathbb{R}^n$ is endowed with a **(Euclidean) inner product** defined as follows:

$$x^T y := \sum_{i=1}^{n} x_i y_i \quad \text{for } x, y \in \mathbb{R}^n.$$

For a real number α, the symbol $\lfloor \alpha \rfloor$ denotes the largest integer not larger than α (the **floor** or **lower integer part** of α), $\lceil \alpha \rceil$ denotes the smallest integer not smaller than α (the **ceiling** or **upper integer part** of α) and $\lceil \alpha \rfloor := \lceil \alpha - \frac{1}{2} \rceil$ denotes the integer nearest to α. If $a = (a_1, \ldots, a_n)^T$ and $b = (b_1, \ldots, b_n)^T$ are vectors, we write $a \leq b$ if $a_i \leq b_i$ for $i = 1, \ldots, n$.

For two sets M and N, the expression $M \subseteq N$ means that M is a subset of N, while $M \subset N$ denotes strict containment, i. e., $M \subseteq N$ and $M \neq N$. We write $M \setminus N$ for the set-theoretical difference $\{x \in M \mid x \notin N\}$, $M \triangle N$ for the **symmetric difference** $(M \setminus N) \cup (N \setminus M)$, and 2^M for the set of all subsets of M, the so-called **power set** of M. For $M, N \subseteq \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, we use the following standard terminology for set operations: $M + N := \{x + y \mid x \in M, y \in N\}$, $\alpha M := \{\alpha x \mid x \in M\}$, $-M := \{-x \mid x \in M\}$, $M - N := M + (-N)$.

For any set R, $R^{m \times n}$ denotes the set of **$m \times n$-matrices** with entries in R. For a matrix $A \in R^{m \times n}$, we usually assume that the row index set of A is $\{1, \ldots, m\}$ and that the column index set is $\{1, \ldots, n\}$. Unless specified otherwise, the elements or entries of $A \in R^{m \times n}$ are denoted by a_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$; we write $A = (a_{ij})$. Vectors with n components are also considered as $n \times 1$-matrices.

If I is a subset of the row index set M of a matrix A and J a subset of the column index set N of A, then A_{IJ} denotes the submatrix of A induced by those rows and columns of A whose indices belong to I and J, respectively. Instead of A_{MJ} (A_{IN} resp.) we frequently write $A_{.J}$ ($A_{I.}$ resp.). A submatrix A of the form A_{II} is called a **principal submatrix** of A. If $K = \{1, \ldots, k\}$ then A_{KK} is called the k-th **leading principal submatrix** of A. $A_{i.}$ is the i-th row of A (so it is a row vector), and $A_{.j}$ is the j-th column of A.

Whenever we do not explicitly state whether a number, vector, or matrix is integral, rational, or complex it is implicitly assumed to be real. Moreover, we often do not specify the dimensions of vectors and matrices explicitly. When operating with them, we always assume that their dimensions are compatible.

The **identity matrix** is denoted by I or, if we want to stress its dimension, by I_n. The symbol 0 stands for any appropriately sized matrix which has all entries equal to zero, and similarly for any zero vector. The symbol $\mathbb{1}$ denotes a vector which has all components equal to one. The **j-th unit vector** in $\mathbb{R}^n$, whose j-th component is one while all other components are zero, is denoted by e_j. If $x = (x_1, \ldots, x_n)^T$ is a vector then the $n \times n$-matrix with the entries $x_1, \ldots, x_n$ on the main diagonal and zeros outside the main diagonal is denoted by $\operatorname{diag}(x)$. If $A \in R^{m \times p}$ and $B \in R^{m \times q}$ then (A, B) (or just $(A\ B)$ if this does not lead to confusion) denotes the matrix in $R^{m \times (p+q)}$ whose first p columns are the columns of A and whose other q columns are those of B.

The **determinant** of an $n \times n$-matrix A is denoted by $\det(A)$. The **trace** of an $n \times n$-matrix A, denoted by $\operatorname{tr}(A)$, is the sum of the elements of the main diagonal of the matrix A.

When using functions like det, tr, or diag we often omit the brackets if there is no danger of confusion, i. e., we frequently write $\det A$ instead of $\det(A)$ etc.

The inverse matrix of an $n\times n$-matrix A is denoted by A^{-1}. If a matrix has an inverse matrix then it is called **nonsingular**, and otherwise **singular**. An $n\times n$-matrix A is nonsigular if and only if $\det A \neq 0$.

Hulls, Independence, Dimension

A vector $x \in \mathbb{R}^n$ is called a **linear combination** of the vectors $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ if, for some $\lambda \in \mathbb{R}^k$,

$$x = \sum_{i=1}^{k} \lambda_i x_i .$$

If, in addition,

$$\left.\begin{array}{l}\lambda \geq 0\\ \lambda^T \mathbb{1} = 1\\ \lambda \geq 0,\ \lambda^T \mathbb{1} = 1\end{array}\right\} \text{ we call } x \text{ a } \left\{\begin{array}{l}\textbf{conic}\\ \textbf{affine}\\ \textbf{convex}\end{array}\right\} \textbf{ combination}$$

of the vectors $x_1, x_2, \ldots, x_k$. These combinations are called **proper** if neither $\lambda = 0$ nor $\lambda = e_j$ for some $j \in \{1, 2, \ldots, k\}$. For a nonempty subset $S \subseteq \mathbb{R}^n$, we denote by

$$\left.\begin{array}{l}\operatorname{lin}(S)\\ \operatorname{cone}(S)\\ \operatorname{aff}(S)\\ \operatorname{conv}(S)\end{array}\right\} \text{ the } \left\{\begin{array}{l}\textbf{linear}\\ \textbf{conic}\\ \textbf{affine}\\ \textbf{convex}\end{array}\right\} \textbf{ hull of the elements of } S,$$

that is, the set of all vectors that are linear (conic, affine, convex) combinations of finitely many vectors of S. For the empty set, we define $\operatorname{lin}(\emptyset) := \operatorname{cone}(\emptyset) := \{0\}$ and $\operatorname{aff}(\emptyset) := \operatorname{conv}(\emptyset) := \emptyset$.

A subset $S \subseteq \mathbb{R}^n$ is called

$$\left\{\begin{array}{l}\text{a } \textbf{linear subspace}\\ \text{a } \textbf{cone}\\ \text{an } \textbf{affine subspace}\\ \text{a } \textbf{convex set}\end{array}\right\} \text{ if } \left\{\begin{array}{l}S = \operatorname{lin}(S)\\ S = \operatorname{cone}(S)\\ S = \operatorname{aff}(S)\\ S = \operatorname{conv}(S)\end{array}\right\}.$$

A subset $S \subseteq \mathbb{R}^n$ is called **linearly (affinely) independent** if none of its members is a proper linear (affine) combination of elements of S; otherwise S is called **linearly (affinely) dependent**. It is well-known that a linearly (affinely) independent subset of $\mathbb{R}^n$ contains at most n elements ($n+1$ elements). For any set $S \subseteq \mathbb{R}^n$, the **rank** of S (**affine rank** of S) denoted by $\operatorname{rank}(S)$ ($\operatorname{arank}(S)$), is the cardinality of the largest linearly (affinely) independent subset of S. For any subset $S \subseteq \mathbb{R}^n$, the **dimension** of S, denoted by $\dim(S)$, is the cardinality of a largest affinely independent subset of S minus one, i. e., $\dim(S) = \operatorname{arank}(S) - 1$. A set $S \subseteq \mathbb{R}^n$ with $\dim(S) = n$ is called **full-dimensional**.

The **rank of a matrix** A (notation: $\operatorname{rank}(A)$), is the rank of the set of its column vectors. This is known to be equal to the rank of the set of its row vectors. An $m\times n$-matrix A is said to have **full row rank** (**full column rank**) if $\operatorname{rank} A = m$ ($\operatorname{rank} A = n$).

Eigenvalues, Positive Definite Matrices

If A is an $n\times n$-matrix, then every complex number λ with the property that there is a nonzero vector $u \in \mathbb{C}^n$ such that $Au = \lambda u$ is called an **eigenvalue** of A. The vector u is called an **eigenvector** of A associated with λ. The function $f(\lambda) := \det(\lambda I_n - A)$ is a polynomial of degree n, called the **characteristic polynomial** of A. Thus the equation

$$\det(\lambda I_n - A) = 0$$

has n (complex) roots (multiple roots counted with their multiplicity). These roots are the (not necessarily distinct) n eigenvalues of A.

We will often consider **symmetric matrices** (i. e., $n\times n$-matrices $A = (a_{ij})$ with $a_{ij} = a_{ji}$, $1 \le i \le j \le n$). It is easy to see that all eigenvalues of real symmetric matrices are real numbers.

There are useful relations between the eigenvalues $\lambda_1, \ldots, \lambda_n$ of a matrix A, its determinant and its trace, namely

$$\det A = \prod_{i=1}^{n} \lambda_i, \tag{0.1.1}$$

$$\operatorname{tr} A = \sum_{i=1}^{n} \lambda_i. \tag{0.1.2}$$

An $n \times n$-matrix A is called **positive definite** (**positive semidefinite**) if A is symmetric and if $x^T Ax > 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$ ($x^T Ax \ge 0$ for all $x \in \mathbb{R}^n$). If A is positive definite then A is nonsingular and its inverse is also positive definite. In fact, for a symmetric $n\times n$-matrix A the following conditions are equivalent:

(0.1.3)
(i) A is positive definite.
(ii) A^{-1} is positive definite.
(iii) All eigenvalues of A are positive real numbers.
(iv) $A = B^T B$ for some nonsingular matrix B.
(v) $\det A_k > 0$ for $k = 1, \ldots, n$, where A_k is the k-th leading principal submatrix of A.

It is well known that for any positive definite matrix A, there is exactly one matrix among the matrices B satisfying (0.1.3) (iv) that is itself positive definite. This matrix is called the **(square) root** of A and is denoted by $A^{1/2}$.

Positive semidefinite matrices can be characterized in a similar way, namely, for a symmetric $n\times n$-matrix A the following conditions are equivalent:

(0.1.4)
(i) A is positive semidefinite.
(ii) All eigenvalues of A are nonnegative real numbers.
(iii) $A = B^T B$ for some matrix B.
(iv) $\det A_{II} \ge 0$ for all principal submatrices A_{II} of A.
(v) There is a positive definite principal submatrix A_{II} of A with $|I| = \operatorname{rank} A$.

Vector Norms, Balls

A function $N : \mathbb{R}^n \to \mathbb{R}$ is called a **norm** if the following three conditions are satisfied:

(0.1.5) (i) $N(x) \geq 0$ for $x \in \mathbb{R}^n$, $N(x) = 0$ if and only if $x = 0$,
(ii) $N(\alpha x) = |\alpha| N(x)$ for all $x \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$,
(iii) $N(x + y) \leq N(x) + N(y)$ for all $x, y \in \mathbb{R}^n$ (**triangle inequality**).

Every norm N on $\mathbb{R}^n$ induces a **distance** d_N defined by

$$d_N(x, y) := N(x - y) \quad \text{for } x, y \in \mathbb{R}^n.$$

For our purposes four norms will be especially important:

$$\|x\| := \sqrt{x^T x} = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2} \quad \text{(the } l_2\text{- or } \textbf{Euclidean norm}\text{)}.$$

(This norm induces the **Euclidean distance** $d(x, y) = \|x - y\|$. Usually, the Euclidean norm is denoted by $\|\cdot\|_2$. But we use it so often that we write simply $\|\cdot\|$.)

$$\|x\|_1 := \sum_{i=1}^{n} |x_i| \qquad \text{(the } l_1\text{- or } \textbf{1-norm}\text{)},$$

$$\|x\|_\infty := \max_{1 \leq i \leq n} |x_i| \qquad \text{(the } l_\infty\text{- or } \textbf{maximum norm}\text{)},$$

$$\|x\|_A := \sqrt{x^T A^{-1} x},$$

where A is a positive definite $n \times n$-matrix. Recall that A induces an inner product $x^T A^{-1} y$ on $\mathbb{R}^n$. Norms of type $\|\cdot\|_A$ are sometimes called **general Euclidean** or **ellipsoidal norms**. We always consider the space $\mathbb{R}^n$ as a Euclidean space endowed with the Euclidean norm $\|\cdot\|$, unless otherwise specified. So all notions related to distances and the like are defined via the Euclidean norm.

For all $x \in \mathbb{R}^n$, the following relations hold between the norms introduced above:

(0.1.6) $$\|x\| \leq \|x\|_1 \leq \sqrt{n}\|x\|,$$

(0.1.7) $$\|x\|_\infty \leq \|x\| \leq \sqrt{n}\|x\|_\infty,$$

(0.1.8) $$\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty .$$

If A is a positive definite $n \times n$-matrix with smallest eigenvalue λ and largest eigenvalue Λ then

(0.1.9) $$\sqrt{\lambda}\|x\| \leq \|x\|_{A^{-1}} \leq \sqrt{\Lambda}\|x\| \quad \text{for all } x \in \mathbb{R}^n.$$

The **diameter** of a set $K \subseteq \mathbb{R}^n$, denoted by $\operatorname{diam}(K)$, is the largest distance between two points of K; more exactly:

$$\operatorname{diam}(K) := \sup\{\|x-y\| \mid x, y \in K\}.$$

The **width** of K is the minimum distance of two parallel hyperplanes with K between them; that is,

$$\operatorname{width}(K) := \inf_{c \in \mathbb{R}^n,\, \|c\|=1} \{\sup\{c^T x \mid x \in K\} - \inf\{c^T x \mid x \in K\}\}.$$

For any set $K \subseteq \mathbb{R}^n$ and any positive real number ε, the set

$$S(K,\varepsilon) := \{x \in \mathbb{R}^n \mid \|x-y\| \leq \varepsilon \text{ for some } y \in K\} \tag{0.1.10}$$

is called the **ball of radius** ε **around** K (with respect to the Euclidean norm). For $K = \{a\}$ we set

$$S(a,\varepsilon) := S(\{a\},\varepsilon)$$

and call $S(a,\varepsilon)$ the **ball of radius** ε **with center** a. $S(0,1)$ is the **unit ball** around zero.

The "unit ball around zero" with respect to the maximum norm is the hypercube $\{x \in \mathbb{R}^n \mid -1 \leq x_i \leq 1, i = 1, \ldots, n\}$. For any positive definite matrix A, the "unit ball around zero" with respect to $\|\cdot\|_A$ is the ellipsoid $\{x \in \mathbb{R}^n \mid x^T A^{-1} x \leq 1\}$ – see Section 3.1 for further details. We shall frequently use the **interior** ε**-ball of** K defined by

$$S(K,-\varepsilon) := \{x \in K \mid S(x,\varepsilon) \subseteq K\}. \tag{0.1.11}$$

The elements of $S(K,-\varepsilon)$ can be viewed as the points "deep inside of K". Note that if K is convex, then

$$S(S(K,\varepsilon),-\varepsilon) = K, \quad S(S(K,-\varepsilon),\varepsilon) \subseteq K, \tag{0.1.12}$$

and that

$$\begin{aligned} S(S(K,-\varepsilon_1),-\varepsilon_2) &= S(K,-\varepsilon_1-\varepsilon_2), \\ S(S(K,\varepsilon_1),\varepsilon_2) &= S(K,\varepsilon_1+\varepsilon_2). \end{aligned} \tag{0.1.13}$$

Equality does not always hold in the containment relation of (0.1.12). (Take, e. g., $K = S(0,\delta)$ where $0 < \delta < \varepsilon$.) But if $\operatorname{diam}(K) \leq 2R$ for some $R > 0$, and if we know that K contains a ball with radius r, then one can easily see that

$$K \subseteq S(S(K,-\frac{\varepsilon r}{2R}),\varepsilon) \tag{0.1.14}$$

for all $\varepsilon \in \mathbb{R}$ with $0 < \varepsilon < 2R$.

Matrix Norms

Any norm on the linear space $\mathbb{R}^{m\times n}$ of $m\times n$-matrices is called a **matrix norm**. If N_1 is a norm on $\mathbb{R}^n$ and N_2 a norm on $\mathbb{R}^m$, then a matrix norm M on $\mathbb{R}^{m\times n}$ is said to be **compatible** with N_1 and N_2 if

$$N_2(Ax) \leq M(A)\, N_1(x) \quad \text{for all } x \in \mathbb{R}^n, \text{ and all } A \in \mathbb{R}^{m\times n}.$$

If N is a vector norm on $\mathbb{R}^n$ and M a matrix norm on $\mathbb{R}^{n\times n}$ such that $N(Ax) \leq M(A)N(x)$ for all $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$, then we say that M is **compatible** with N. A matrix norm M on $\mathbb{R}^{n\times n}$ is called **submultiplicative** if

$$M(AB) \leq M(A)\, M(B) \quad \text{for all } A, B \in \mathbb{R}^{n\times n}.$$

Every vector norm N on $\mathbb{R}^n$ induces a matrix norm on $\mathbb{R}^{n\times n}$, called the norm **subordinate** to N and denoted by lub_N (**least upper bound**), as follows:

$$\text{lub}_N(A) := \max_{x\neq 0} \frac{N(Ax)}{N(x)} \; (= \max\{N(Ax) \mid N(x) = 1\}). \tag{0.1.15}$$

Clearly, lub_N is the smallest among the matrix norms compatible with N; and lub_N is submultiplicative. The norm subordinate to the Euclidean norm $\|\cdot\|$ is

$$\|A\| := \text{lub}_{\|\cdot\|}(A) = \max_{x\neq 0} \sqrt{\frac{x^T A^T A x}{x^T x}} = \sqrt{\Lambda(A^T A)} \tag{0.1.16}$$

where $\Lambda(A^T A)$ is the largest eigenvalue of $A^T A$. $\|A\|$ is called the **spectral norm** of A. If A is a symmetric $n\times n$-matrix then one can show

$$\|A\| = \max\{|\lambda| \mid \lambda \text{ eigenvalue of } A\} = \max\{|x^T A x| \mid \|x\| = 1\}. \tag{0.1.17}$$

Moreover, if A is symmetric and nonsingular then

$$\|A^{-1}\| = \max\{|\lambda|^{-1} \mid \lambda \text{ eigenvalue of } A\} = \max_{x\neq 0} \frac{x^T x}{x^T A x}. \tag{0.1.18}$$

The norm subordinate to the maximum norm $\|\cdot\|_\infty$ is

$$\|A\|_\infty := \text{lub}_{\|\cdot\|_\infty}(A) = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| \quad \textbf{(row sum norm)}. \tag{0.1.19}$$

The norm subordinate to the 1-norm $\|\cdot\|_1$ is

$$\|A\|_1 := \text{lub}_{\|\cdot\|_1}(A) = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| \quad \textbf{(column sum norm)}. \tag{0.1.20}$$

Further submultiplicative matrix norms are

$$\|A\|_2 := \sqrt{\sum_{i,j=1}^{n} a_{ij}^2} \quad \textbf{(Frobenius norm)} \tag{0.1.21}$$

and

$$\|A\|_{\max} := n \max \{|a_{ij}| \mid i,j = 1, \ldots, n\}. \tag{0.1.22}$$

The matrix norm $\|\cdot\|_{\max}$ is compatible with the vector norms $\|\cdot\|$, $\|\cdot\|_1$, and $\|\cdot\|_\infty$, and the matrix norm $\|\ \|_2$ is compatible with the Euclidean norm $\|\cdot\|$. The following inequalities hold:

$$\|A\| \leq \|A\|_{\max},\ \|A\|_\infty \leq \|A\|_{\max},\ \|A\|_1 \leq \|A\|_{\max} \quad \text{for } A \in \mathbb{R}^{n\times n}, \tag{0.1.23}$$

$$\frac{1}{\sqrt{n}}\|A\|_2 \leq \|A\| \leq \|A\|_2 \leq \|A\|_{\max} \quad \text{for } A \in \mathbb{R}^{n\times n}. \tag{0.1.24}$$

If M is a matrix norm on $\mathbb{R}^{n\times n}$ that is compatible to some vector norm on $\mathbb{R}^n$, then

$$|\lambda| \leq M(A) \quad \text{for each eigenvalue } \lambda \text{ of } A. \tag{0.1.25}$$

Some Inequalities

The following well-known inequality will be used frequently:

$$|x^T y| \leq \|x\|\,\|y\| \quad \text{for all } x, y \in \mathbb{R}^n \quad \textbf{(Cauchy-Schwarz inequality)}. \tag{0.1.26}$$

This inequality holds with equality if and only if x and y are linearly dependent.

The **parallelepiped** spanned by $a_1, \ldots, a_m \in \mathbb{R}^n$ is the convex hull of the zero vector and all vectors that are sums of different vectors of the set $\{a_1, \ldots, a_m\}$. Let A be the $n\times m$-matrix with columns $a_1, \ldots, a_m$. Then

$$\sqrt{\det(A^T A)}$$

is the **volume** of the parallelepiped spanned by $a_1, \ldots, a_m$. This observation implies

$$\sqrt{\det(A^T A)} \leq \prod_{i=1}^{m} \|a_i\| \quad \textbf{(Hadamard inequality)}. \tag{0.1.27}$$

Equality holds in (0.1.27) if and only if the vectors a_i are **orthogonal** (i. e., $a_i^T a_j = 0$ for $1 \leq i < j \leq m$), in other words, if and only if the parallelepiped is a rectangular box. In particular, for an $n\times n$-matrix A the Hadamard inequality reads

$$|\det A| \leq \prod_{i=1}^{n} \|a_i\|, \tag{0.1.28}$$

where $a_1, \ldots, a_n$ denote the columns of A.

Polyhedra, Inequality Systems

If A is a real $m \times n$-matrix and $b \in \mathbb{R}^m$, then $Ax \leq b$ is called a **system of (linear) inequalities**, and $Ax = b$ a **system of (linear) equations**. The solution set $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ of a system of inequalities is called a **polyhedron**. A polyhedron P that is **bounded** (i. e., $P \subseteq S(0, R)$ for some $R > 0$) is called a **polytope**. A **polyhedral cone** is a cone that is also a polyhedron.

If $a \in \mathbb{R}^n \setminus \{0\}$ and $a_0 \in \mathbb{R}$, then the polyhedron $\{x \in \mathbb{R}^n \mid a^T x \leq a_0\}$ is called a **halfspace**, and the polyhedron $\{x \in \mathbb{R}^n \mid a^T x = a_0\}$ a **hyperplane**. To shorten notation we shall sometimes speak of the hyperplane $a^T x = a_0$ and the halfspace $a^T x \leq a_0$. Every polyhedron is the intersection of finitely many halfspaces.

An inequality $a^T x \leq a_0$ is called **valid** with respect to a polyhedron P if $P \subseteq \{x \mid a^T x \leq a_0\}$. A set $F \subseteq P$ is called a **face** of P if there exists a valid inequality $a^T x \leq a_0$ for P such that $F = \{x \in P \mid a^T x = a_0\}$. We say that F is the **face defined** (or **induced**) by $a^T x \leq a_0$. If v is a point in a polyhedron P such that $\{v\}$ is a face of P, then v is called a **vertex** of P. A polyhedron is called **pointed** if it has a vertex. One can easily prove that a nonempty polyhedron $P = \{x \mid Ax \leq b\}$ is pointed if and only if A has full column rank. A **facet** of P is an inclusionwise maximal face F with $\emptyset \neq F \neq P$. Equivalently, a facet is a nonempty face of P of dimension $\dim(P) - 1$.

Note that the linear subspaces of $\mathbb{R}^n$ are the solution sets of homogeneous equation systems $Ax = 0$, and that the affine subspaces are the solution sets of equation systems $Ax = b$. Hence they are polyhedra.

Let us remark at this point that the description of a polyhedron as the solution set of linear inequalities is by no means unique. Sometimes it will be convenient to have a kind of "standard" representation of a polyhedron. If the polyhedron P is full-dimensional then it is well known that P has a representation $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ such that each of the inequalities $A_{i.}x \leq b_i$ defines a facet of P and such that each facet of P is defined by exactly one of these inequalities. If we normalize the inequalities so that $\|A_{i.}\|_\infty = 1$ for all rows $A_{i.}$ of A, then this representation of P is unique. We call this representation the **standard representation** of the full-dimensional polyhedron P.

If the polyhedron P is not full-dimensional, then one can find a system $Cx = d$ of linear equations whose solution set is the affine hull of P. We can choose – after permuting coordinates, if necessary – the matrix C and the right hand side d so that $C = (I, C')$ holds. Moreover, given such a matrix C, we can find an inequality system $Ax \leq b$ such that each of the inequalities $A_{i.}x \leq b_i$ defines a facet of P, each facet of P is defined by exactly one of these inequalities, each row of A is orthogonal to each row of C, and $\|A_{i.}\|_\infty = 1$ holds for each i. A representation of a polyhedron P of the form

$$P = \{x \in \mathbb{R}^n \mid Cx = d, Ax \leq b\},$$

where C, d, A, b satisfy the above requirements will be called a **standard representation** of P. This representation is unique up to the choice of the columns forming the identity matrix I.

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r. A submatrix B of A is called a **basis** of A if it is a nonsingular $r \times r$-matrix. Let $A \in \mathbb{R}^{m \times n}$ have full column rank, and let

$b \in \mathbb{R}^m$. For any basis $B = A_I$. of A, let b_I be the subvector of b corresponding to B. Then the vector $B^{-1}b_I$ is called a **basic solution** of $Ax \leq b$. Warning: $B^{-1}b_I$ need not satisfy $Ax \leq b$. If $B^{-1}b_I$ satisfies $Ax \leq b$ it is called a **basic feasible solution** of this inequality system. It is easy to see that a vector is a vertex of the polyhedron $P = \{x \mid Ax \leq b\}$, A with full column rank, if and only if it is a basic feasible solution of $Ax \leq b$ for some basis B. Note that this basis need not be unique.

There is another way of representing polyhedra, using the convex and conic hull operation, that is in some sense "polar" to the one given above. Polytopes are the convex hulls of finitely many points. In fact, every polytope is the convex hull of its vertices. Every polyhedron P has a representation of the form

$$P = \text{conv}(V) + \text{cone}(E),$$

where V and E are finite subsets of $\mathbb{R}^n$. Moreover, every point in a polytope $P \subseteq \mathbb{R}^n$ is a convex combination of at most $\dim(P) + 1$, and thus of at most $n+1$, affinely independent vertices of P (Carathéodory's theorem). The following type of polytopes will come up frequently. A set $\Sigma \subseteq \mathbb{R}^n$ is called a **d-simplex** (where $1 \leq d \leq n$) if $\Sigma = \text{conv}(V)$ and V is a set of $d+1$ affinely independent points in $\mathbb{R}^n$. Instead of n-simplex in $\mathbb{R}^n$ we often say just **simplex**.

For any nonempty set $S \subseteq \mathbb{R}^n$,

$$\text{rec}(S) := \{y \in \mathbb{R}^n \mid x + \lambda y \in S \text{ for all } x \in S \text{ and all } \lambda \geq 0\}$$

denotes the **recession cone** (or characteristic cone) of S. We set $\text{rec}(\emptyset) := \{0\}$. Intuitively, every vector of $\text{rec}(S)$ represents a "direction to infinity" in S. A more general version of Carathéodory's theorem states that every point in a d-dimensional polyhedron P can be represented as a convex combination of affinely independent points $v_1, \ldots, v_s, v_1 + e_1, \ldots, v_1 + e_t$, where $s + t \leq d + 1$, the points $v_1, \ldots, v_s$ are elements of minimal nonempty faces of P, and the points $e_1, \ldots, e_t$ are elements of minimal nonzero faces of the recession cone of P.

By

$$\text{lineal}(S) := \{y \in \text{rec}(S) \mid -y \in \text{rec}(S)\} = \text{rec}(S) \cap (-\text{rec}(S))$$

we denote the **lineality space** of a set $S \subseteq \mathbb{R}^n$. If $S \neq \emptyset$, the lineality space of S is the largest linear subspace L of $\mathbb{R}^n$ such that $x + L \subseteq S$ for all $x \in S$. The recession cone and the lineality space of polyhedra can be characterized nicely. Namely, if $P = \{x \mid Ax \leq b\}$ is nonempty, then

$$\text{rec}(P) = \{x \mid Ax \leq 0\},$$
$$\text{lineal}(P) = \{x \mid Ax = 0\}.$$

If $P = \text{conv}(V) + \text{cone}(E)$ and $V \neq \emptyset$, then

$$\text{rec}(P) = \text{cone}(E),$$
$$\text{lineal}(P) = \text{lin}(\{e \in E \mid -e \in \text{cone}(E)\}) = \text{cone}(E) \cap (\text{cone}(-E)).$$

A set $S \subseteq \mathbb{R}^n$ is called **up-monotone (down-monotone)** if for each $y \in S$ all vectors $x \in \mathbb{R}^n$ with $x \geq y$ ($x \leq y$) are in S. Sometimes we shall restrict our attention to subsets of a given set T (for instance $T = \mathbb{R}^n_+$ or $T = [0,1]^n$), in which case we call a set $S \subseteq T$ **up-monotone (down-monotone) in** T if, for each $y \in S$, all vectors $x \in T$ with $x \geq y$ ($x \leq y$) are in S.

The **dominant** of a set S is the smallest up-monotone convex set containing S, that is, the dominant is the set $\text{conv}(S) + \mathbb{R}^n_+$. Similarly the **antidominant** of S is the smallest down-monotone convex set containing S, i. e., $\text{conv}(S) - \mathbb{R}^n_+$. The **dominant** and **antidominant in a convex set** T are defined in the obvious way.

For any set $S \subseteq \mathbb{R}^n$,

$$S^\circ := \{y \in \mathbb{R}^n \mid y^T x \leq 0 \text{ for all } x \in S\}$$

is called the **polar cone** of S, and

$$S^* := \{y \in \mathbb{R}^n \mid y^T x \leq 1 \text{ for all } x \in S\}$$

is called the **polar** of S. S is a closed cone if and only if $(S^\circ)^\circ = S$. S is a closed convex set containing zero if and only if $(S^*)^* = S$. Moreover, if $P = \text{conv}(V) + \text{cone}(E)$ is a nonempty polyhedron then

$$P^\circ = \{x \mid y^T x \leq 0 \text{ for all } y \in V \cup E\},$$
$$P^* = \{x \mid v^T x \leq 1 \text{ for all } v \in V \text{ and } e^T x \leq 0 \text{ for all } e \in E\}.$$

There are two related operations which are important in combinatorial applications. For any set $S \subseteq \mathbb{R}^n_+$, its **blocker** is

$$\text{bl}(S) := \{y \in \mathbb{R}^n_+ \mid y^T x \geq 1 \text{ for all } x \in S\},$$

and its **antiblocker** is

$$\text{abl}(S) := \{y \in \mathbb{R}^n_+ \mid y^T x \leq 1 \text{ for all } x \in S\}.$$

It is well known that $\text{bl}(\text{bl}(S)) = S$ if and only if $S \subseteq \mathbb{R}^n_+$ and S is closed, convex, and up-monotone. Furthermore, $\text{abl}(\text{abl}(S)) = S$ if and only if $S \subseteq \mathbb{R}^n_+$ and S is nonempty, closed, convex, and down-monotone in $\mathbb{R}^n_+$.

Linear (Diophantine) Equations and Inequalities

There are some basic problems concerning linear spaces and polyhedra which will occur frequently in our book and which play an important role in applications.

Let an $m \times n$-matrix A and a vector $b \in \mathbb{R}^m$ be given. Consider the system of linear equations

$$Ax = b. \tag{0.1.29}$$

Then we can formulate the following four problems:

(0.1.30) Find a solution of (0.1.29).

(0.1.31) Find an integral solution of (0.1.29).

(0.1.32) Find a nonnegative solution of (0.1.29).

(0.1.33) Find a nonnegative integral solution of (0.1.29).

Borrowing a term from number theory, we could call problems (0.1.31) and (0.1.33) the *diophantine* versions of (0.1.30) and (0.1.32), respectively.

We can ask similar questions about the solvability of the system of linear inequalities

$$Ax \leq b, \tag{0.1.34}$$

namely:

(0.1.35) Find a solution of (0.1.34).

(0.1.36) Find an integral solution of (0.1.34).

(0.1.37) Find a nonnegative solution of (0.1.34).

(0.1.38) Find a nonnegative integral solution of (0.1.34).

Obviously, the nonnegativity conditions in (0.1.37) and (0.1.38) could be included in (0.1.34); hence, (0.1.37) is equivalent to (0.1.35), and (0.1.38) is equivalent to (0.1.36). Furthermore, it is rather easy to see that problem (0.1.35) is equivalent to (0.1.32); and if A and b are rational then (0.1.36) is equivalent to (0.1.33).

For problems (0.1.30), (0.1.31), and (0.1.32), there are classical necessary and sufficient conditions for their solvability.

(0.1.39) Theorem (Solvability of Linear Equations). *There exists a vector $x \in \mathbb{R}^n$ such that $Ax = b$ if and only if there does not exist a vector $y \in \mathbb{R}^n$ such that $y^T A = 0$ and $y^T b \neq 0$.* □

(0.1.40) Theorem (Integral Solvability of Linear Equations). *Assume A and b are rational. Then there exists a vector $x \in \mathbb{Z}^n$ such that $Ax = b$ if and only if there does not exist a vector $y \in \mathbb{R}^m$ such that $y^T A$ is integral and $y^T b$ is not integral.* □

(0.1.41) Theorem (Nonnegative Solvability of Linear Equations). *There exists a vector $x \in \mathbb{R}^n$ such that $Ax = b$, $x \geq 0$ if and only if there does not exist a vector $y \in \mathbb{R}^m$ such that $y^T A \geq 0$ and $y^T b < 0$.* □

Theorem (0.1.41) is known as the **Farkas lemma**. The Farkas lemma has the following equivalent version which gives a necessary and sufficient condition for (0.1.35).

(0.1.42) Theorem. *There exists a vector $x \in \mathbb{R}^n$ such that $Ax \leq b$ if and only if there does not exist a vector $y \in \mathbb{R}^m$ such that $y^T A = 0$, $y \geq 0$ and $y^T b < 0$.* □

We leave it to the reader as an exercise to formulate the version of the Farkas lemma characterizing the solvability of (0.1.37).

Problem (0.1.33) (equivalently (0.1.36) and (0.1.38)) is much more difficult than the other problems. A kind of characterization was obtained by CHVÁTAL (1973) and SCHRIJVER (1980a), based on work of GOMORY (1958, 1960). We shall formulate this criterion for problem (0.1.38) only, and leave its adaptation to problems (0.1.33) and (0.1.36) to the reader.

The essence of the solvability characterizations (0.1.39), (0.1.40), and (0.1.41) is that if $Ax = b$ does not have a solution of a certain type, then we can infer one single linear equation from $Ax = b$ for which it is obvious that it does not have a solution of this type. Here "infer" means that we take a linear combination of the given equations. The version (0.1.42) of the Farkas lemma characterizing the solvability of (0.1.35) may be viewed similarly, but here we use the inference rule that we can take a *nonnegative* linear combination of the given linear inequalities.

The solvability criterion for (0.1.38) can be formulated along the same lines, but we have to allow the following more complicated rules.

(0.1.43) Rules of Inference.

Rule 1. *Given inequalities $a_1^T x \leq \beta_1, \ldots, a_m^T x \leq \beta_m$ and $\lambda_1, \ldots, \lambda_m \geq 0$, infer the inequality $(\sum_{i=1}^m \lambda_i a_i^T)x \leq \sum_{i=1}^m \lambda_i \beta_i$.*

Rule 2. *Given an inequality $\alpha_1 x_1 + \ldots + \alpha_n x_n \leq \beta$, infer the inequality $\lfloor \alpha_1 \rfloor x_1 + \ldots + \lfloor \alpha_n \rfloor x \leq \lfloor \beta \rfloor$.* □

It is obvious that, if a nonnegative integral vector x satisfies the given inequalities in Rule 1 or Rule 2, it also satisfies the inferred inequality. This observation gives the trivial direction of the following theorem.

(0.1.44) Theorem (Nonnegative Integral Solvability of Linear Inequalities). *Assume that A and b are rational. Then there exists a vector $x \in \mathbb{Z}^n$, $x \geq 0$ such that $Ax \leq b$ if and only if we cannot infer from $Ax \leq b$ by a repeated application of Rule 1 and Rule 2 of* (0.1.43) *the inequality $0^T x \leq -1$.* □

It is important to note here that to derive the inequality $0^T x \leq -1$, it may be necessary to apply the Rules 1 and 2 of (0.1.43) a large number of times.

(0.1.45) Example. Consider the following inequality system in $\mathbb{R}^2$:

$$\begin{aligned} x + y &\leq 3.5, \\ x - y &\leq 0.5, \\ -x + y &\leq 0.5, \\ -x - y &\leq -2.5. \end{aligned}$$

This system does not have a nonnegative integral solution. First we apply Rule 2 to each of the given inequalities to obtain

$$\begin{aligned} x + y &\leq 3, \\ x - y &\leq 0, \\ -x + y &\leq 0, \\ -x - y &\leq -3. \end{aligned}$$

From the first two inequalities above we infer by Rule 1 that

$$x \leq 1.5.$$

Similarly, the last two inequalities yield, by Rule 1, that

$$-x \leq -1.5.$$

Now applying Rule 2 gives

$$\begin{aligned} x &\leq 1, \\ -x &\leq -2, \end{aligned}$$

adding these by Rule 1 gives

$$0x + 0y \leq -1.$$

The reader is invited to verify that, in this example, to infer $0x + 0y \leq -1$ from the given system, Rule 2 has to be applied more than once; in fact, Rule 2 has to be applied to an inequality that itself was obtained by using Rule 2 at some earlier step. □

It is also true that if we start with any system of inequalities $Ax \leq b$ with at least one nonnegative integral solution, we can derive, using Rules 1 and 2 a finite number of times, every inequality that is valid for all nonnegative integral solutions of $Ax \leq b$.

Linear Programming and Duality

One of the most important problems of mathematical programming, and in various ways the central subject of this book, is the following problem.

(0.1.46) Linear Programming Problem (LP). *Given an $m \times n$-matrix A, a vector $b \in \mathbb{R}^m$, and a vector $c \in \mathbb{R}^n$, find a vector $x^* \in P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ maximizing the linear function $c^T x$ over P.*

A linear programming problem (we also say just **linear program**) is usually written in one of the following short forms:

(0.1.47)
$$\begin{aligned} &\max\ c^T x \\ &Ax \leq b, \\ \text{or}\quad &\max\{c^T x \mid Ax \leq b\}, \\ \text{or}\quad &\max c^T x,\ Ax \leq b, \\ \text{or}\quad &\max c^T x,\ x \in P. \end{aligned}$$

As we will see in the sequel there are various other ways to present a linear program. A vector $\bar{x}$ satisfying $A\bar{x} \leq b$ is called a **feasible solution** of the linear program, and a feasible solution $\bar{x}$ is called an **optimal solution** if $c^T\bar{x} \geq c^T x$ for all feasible vectors x. The linear function $c^T x$ is called the **objective function** of the linear program. If we replace "maximizing" in (0.1.46) by "minimizing", the resulting problem is also called a linear program and the same terminology applies.

With every linear program $\max c^T x$, $Ax \leq b$, another program, called its **dual**, can be associated; this reads as follows:

$$\min y^T b,\ y^T A = c^T,\ y \geq 0, \tag{0.1.48}$$

where y is the variable vector. Using trivial transformations this program can be brought into the form of a linear program as defined above. The original program is sometimes referred to as the **primal** program. The following fundamental theorem establishes an important connection between a primal problem and its dual.

(0.1.49) Duality Theorem. *Let* (P) $\max c^T x$, $Ax \leq b$ *be a linear program and* (D) $\min y^T b$, $y^T A = c^T$, $y \geq 0$ *be its dual. If* (P) *and* (D) *both have feasible solutions then both problems have optimal solutions and the optimum values of the objective functions are equal.*

If one of the programs (P) *or* (D) *has no feasible solution, then the other is either unbounded or has no feasible solution. If one of the programs* (P) *or* (D) *is unbounded then the other has no feasible solution.* □

This theorem can be derived from, and is equivalent to, the Farkas lemma (0.1.41). A useful optimality condition for linear programming problems is the following result.

(0.1.50) Complementary Slackness Theorem. *Suppose* u *is a feasible solution to the primal linear programming problem* (P) $\max c^T x$, $Ax \leq b$ *and* v *is a feasible solution for the dual* (D) $\min y^T b$, $y^T A = c^T$, $y \geq 0$. *A necessary and sufficient condition for* u *and* v *to be optimal for* (P) *and* (D), *respectively, is that for all* i

$$v_i > 0 \quad \textit{implies} \quad A_i.u = b_i,$$
$$\textit{(equivalently, } A_i.u < b_i \quad \textit{implies} \quad v_i = 0\textit{)}.$$

□

It follows from the definition of optimality that the set of optimum solutions of a linear program over a polyhedron P is a face of the polyhedron P. If P is a pointed polyhedron, then every face contains at least one vertex. Hence, if P is pointed and the linear program $\max c^T x$, $x \in P$ is bounded then it has at least one optimum solution x^* that is a vertex of P. In particular, this implies that every linear program over a nonempty polytope has an optimum vertex solution.

There is a diophantine version of the linear programming problem, important in view of combinatorial applications.

(0.1.51) Integer Linear Programming Problem. *Given an $m \times n$-matrix A, vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, find an integral vector $x^* \in P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ maximizing the linear function $c^T x$ over the integral vectors in P.*

Almost every combinatorial optimization problem can be formulated as such an integer linear programming problem.

Given an integer linear program, the linear program which arises by dropping the integrality stipulation is called its **LP-relaxation**. We may also consider the dual of this LP-relaxation and the associated integer linear program. Then we have the following inequalities (provided that the optima involved exist):

$$\begin{array}{ccccccc} \max c^T x & \leq & \max c^T x & = & \min y^T b & \leq & \min y^T b \\ Ax \leq b & & Ax \leq b & & y^T A = c^T & & y^T A = c^T \\ x \text{ integral} & & & & y \geq 0 & & y \geq 0 \\ & & & & & & y \text{ integral} \end{array} \tag{0.1.52}$$

In general, one or both of these inequalities can be strict.

We say that the system $Ax \leq b$ is **totally primal integral (TPI)** if A and b are rational and the first inequality in (0.1.52) holds with equality for each integral vector c, for which the maxima are finite. The system $Ax \leq b$ is **totally dual integral (TDI)** if A and b are rational and the second inequality in (0.1.52) holds with equality for each integral vector c, for which the minima are finite. The following theorem due to EDMONDS and GILES (1977) relates these two concepts.

(0.1.53) Theorem. *If b is an integral vector and $Ax \leq b$ is totally dual integral, then it is also totally primal integral.* □

Note that total primal integrality and total dual integrality are not dual concepts (the roles of b and c are not symmetric); in fact, the converse of Theorem (0.1.53) does not hold. Observe that the condition that c is an integral vector can be dropped from the definition of total primal integrality without changing this notion. Geometrically, total primal integrality means that P is equal to the convex hull of the integral points contained in P – in particular, if P is pointed then all vertices of P are integral. A further property equivalent to total primal integrality is that for each integral vector c the optimum value of $\max c^T x$, $Ax \leq b$ is an integer (if it is finite).

0.2 Graph Theory

Unfortunately there is no standard terminology in graph theory. We shall use a mixture of BERGE (1973), BOLLOBÁS (1978), BONDY and MURTY (1976), and LAWLER (1976) but we shall also introduce some new symbols which are more appropriate for our purposes.

Graphs

A **graph** $G = (V, E)$ consists of a finite nonempty set V of **nodes** and a finite set E of **edges**. With every edge, an unordered pair of nodes, called its **endnodes**, is associated and we say that an edge is **incident** to its endnodes. Note that we usually assume that the two endnodes of an edge are distinct, i. e., we do not allow loops, unless specified otherwise. If there is no danger of confusion we denote an edge e with endnodes i and j by ij. Two edges are called **parallel** if they have the same endnodes. A graph without parallel edges is called **simple**. The number of nodes of G is called the **order** of G.

A node that is not incident to any edge is called **isolated**. Two nodes that are joined by an edge are called **adjacent** or **neighbors**. For a node set W, $\Gamma(W)$ denotes the set of neighbors of nodes in W. We write $\Gamma(v)$ for $\Gamma(\{v\})$. The set of edges having a node $v \in V$ as one of their endnodes is denoted by $\delta(v)$. The number $|\delta(v)|$ is the **degree** of node $v \in V$. More generally, if $W \subseteq V$, then $\delta(W)$ denotes the set of edges with one endnode in W and the other endnode in $V \setminus W$. Any edge set of the form $\delta(W)$, where $\emptyset \neq W \neq V$, is called a **cut**.

If s and t are two different nodes of G, then an edge set $F \subseteq E$ is called an **$[s,t]$-cut** if there exists a node set $W \subseteq V$ with $s \in W$, $t \notin W$ such that $F = \delta(W)$. (We shall often use the symbol $[.,.]$ to denote a pair of objects where the order of the objects does not play a role; here in particular, an $[s,t]$-cut is also a $[t,s]$-cut.)

If $W \subseteq V$ and $F \subseteq E$ then $E(W)$ denotes the set of edges in $G = (V, E)$ with both endnodes in W, and $V(F)$ denotes the nodes of G which occur as an endnode of at least one edge in F.

If W is a node set in $G = (V, E)$, then $G - W$ denotes the graph obtained by **removing** (or **deleting**) W, i. e., the node set of $G-W$ is $V \setminus W$ and $G-W$ contains all edges of G which are not incident to a node in W. By $G[W]$ we denote the subgraph of G **induced by** a node set $W \subseteq V$, i. e., $G[W] = G - (V \setminus W)$. For $F \subseteq E$, the graph $G - F := (V, E \setminus F)$ is called the graph obtained from G by **removing** (or **deleting**) F. For $v \in V$ and $e \in E$, we write $G - v$ and $G - e$ instead of $G - \{v\}$ and $G - \{e\}$.

For a node set $W \subseteq V$, the graph $G \cdot W$ denotes the graph obtained by **contracting** W, i. e., $G \cdot W$ contains all nodes $V \setminus W$ and a new node, say w, that replaces the node set W. All edges of G not incident to a node in W are kept, all edges of G with both endnodes in W are removed, and for all edges of G with exactly one endnode in W, this endnode is replaced by w (so parallel edges may result). If $e = uv \in E$ and G contains no edge parallel to e, then the contraction $G \cdot e$ of e is the graph $G \cdot \{u, v\}$. If G contains edges parallel to e, $G \cdot e$ is obtained by adding as many loops to $G \cdot \{v, w\}$ containing the new node w as there are edges parallel to e in G. Since here loops come up, we will be careful with this operation. The contraction of a loop results in the same graph as its deletion. The contraction $G \cdot F$ of an edge set F is the graph obtained by contracting the edges of F (in any order).

A **matching** (or **1-matching**) M in a graph $G = (V, E)$ is a set of edges such that no two edges of M have a common endnode. A matching M is called **perfect** if every node is contained in one edge of M.

A simple graph is called **complete** if every two of its nodes are joined by an edge. The (up to isomorphism unique) complete graph of order n is denoted by K_n. A graph G whose node set V can be partitioned into two nonempty disjoint sets V_1, V_2 with $V_1 \cup V_2 = V$ such that no two nodes in V_1 and no two nodes in V_2 are adjacent is called **bipartite**. The node sets V_1, V_2 are called **color classes**, a **2-coloring**, or a **bipartition** of V. If G is simple and bipartite, $|V_1| = m$, $|V_2| = n$, and every node in V_1 is adjacent to every node in V_2, then G is called **complete bipartite** and is denoted by $K_{m,n}$. The complete bipartite graph $K_{1,n}$ is called a **star**, and the star $K_{1,3}$ a **claw**.

If G is a graph, then the **complement** of G, denoted by $\overline{G}$, is the simple graph which has the same node set as G and in which two nodes are adjacent if and only if they are nonadjacent in G.

The **line graph** $L(G)$ of a graph G is the simple graph whose node set is the edge set of G and in which two nodes are adjacent if and only if the corresponding edges of G have a common endnode.

A **stable set** (**clique**) in a graph $G = (V, E)$ is a set of nodes any two of which are nonadjacent (adjacent). A **coloring** (**clique covering**) of a graph $G = (V, E)$ is a partition of V into disjoint stable sets (cliques).

Clearly, every graph $G = (V, E)$ can be drawn in the plane by representing nodes as points and edges as lines linking the two points which represent their endnodes. A graph is called **planar**, if it can be drawn in the plane in such a way that no two edges (i. e., the lines representing the edges) intersect, except possibly in their endpoints.

Digraphs

A **directed graph** (or **digraph**) $D = (V, A)$ consists of a finite nonempty set V of **nodes** and a set A of **arcs**. With every arc a, an ordered pair (u, v) of nodes, called its **endnodes**, is associated; u is the **initial endnode** (or **tail**) and v the **terminal endnode** (or **head**) of a. As in the undirected case, loops (u, u) will only be allowed if explicitly stated. If there is no danger of confusion we denote an arc a with tail u and head v by (u, v); we also say that a **goes from** u **to** v, that a is **incident from** u and **incident to** v, and that a **leaves** u and **enters** v. If there is an arc going from u to v, we say that u is a **predecessor** of v and that v is a **successor** of u.

If $D = (V, A)$ is a digraph and $W \subseteq V$, $B \subseteq A$, then $V(B)$ is the set of nodes occurring at least once as an endnode of an arc in B, and $A(W)$ is the set arcs with head and tail in W. Deletion and contraction of node or arc sets is defined in the same way as for undirected graphs.

If $D = (V, A)$ is a digraph, then the graph $G = (V, E)$ having an edge ij whenever $(i, j) \in A$ or $(j, i) \in A$ is called the **underlying graph** of D. A digraph has an "undirected property" whenever its underlying graph has this property. For example, a digraph is planar if its underlying graph is planar.

If $v \in V$ then the set of arcs having v as initial (terminal) node is denoted by $\delta^+(v)$ $(\delta^-(v))$; we set $\delta(v) := \delta^+(v) \cup \delta^-(v)$. The numbers $|\delta^+(v)|$, $|\delta^-(v)|$, and $|\delta(v)|$ are called the **outdegree**, **indegree**, and **degree** of v, respectively.

For any set $W \subseteq V$, $\emptyset \neq W \neq V$, we set $\delta^+(W) := \{(i,j) \in A \mid i \in W, j \notin W\}$, $\delta^-(W) := \delta^+(V \setminus W)$, and $\delta(W) := \delta^+(W) \cup \delta^-(W)$. If s, t are two different nodes

of a digraph $D = (V, A)$, then an arc set $F \subseteq A$ is called an (s,t)-**cut** ($[s,t]$-**cut**) in D if there is a node set W with $s \in W$ and $t \notin W$ such that $F = \delta^+(W)$ ($F = \delta(W)$). An arc set of the form $\delta^+(W)$, $\emptyset \neq W \neq V$, is called a **directed cut** or **dicut** if $\delta^-(W) = \emptyset$, i. e., $\delta(W) = \delta^+(W) = \delta^-(V \setminus W)$. If $r \in V$ then every arc set of the form $\delta^-(W)$, where $\emptyset \neq W \subseteq V \setminus \{r\}$ is called an r-**rooted cut** or just r-**cut**.

Walks, Paths, Circuits, Trees

In a graph or digraph, a **walk** is a finite sequence $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ ($k \geq 0$), beginning and ending with a node, in which nodes v_i and edges (arcs) e_j appear alternately, such that for $i = 1, 2, \ldots, k$ the endnodes of every edge (arc) e_i are the nodes v_{i-1}, v_i. The nodes v_0 and v_k are called the **origin** and the **terminus**, respectively, or the **endnodes** of W. The nodes $v_1, \ldots, v_{k-1}$ are called the **internal nodes** of W. The number k is the **length** of the walk. If (in a digraph) all arcs e_i are of the form (v_{i-1}, v_i) then W is called a **directed walk** or **diwalk**. An edge (arc) connecting two nodes of a walk but not contained in the walk is called a **chord** of the walk.

A walk in which all nodes (edges or arcs) are distinct is called a **path** (**trail**). A path in a digraph that is a diwalk is called a **directed path** or **dipath**. If a node s is the origin of a walk (diwalk) W and t the terminus of W, then W is called an $[s,t]$-**walk** ((s,t)-**diwalk**).

Two nodes s, t of a graph G are said to be **connected** if G contains an $[s,t]$-path. G is called **connected** if every two nodes of G are connected. A digraph D is called **strongly connected** (or **diconnected**) if for every two nodes s, t of D there are an (s,t)-dipath and a (t,s)-dipath in D.

A graph G (digraph D) is called k-**connected** (k-**diconnected**) if every pair s, t of nodes is connected by at least k $[s,t]$-paths ((s,t)-dipaths) whose sets of internal nodes are mutually disjoint. The **components** of a graph are the maximal connected subgraphs of the graph. An edge e of G is called a **bridge** (or **isthmus**) if $G - e$ has more components than G. A **block** of a graph is a node induced subgraph (W, F) such that either $F = \{f\}$ and f is a bridge, or (W, F) is 2-connected and maximal with respect to this property.

A walk is called **closed** if it has nonzero length and its origin and terminus are identical. A closed walk (diwalk) in which the origin and all internal nodes are different and all edges (arcs) are different is called a **circuit** (**dicycle** or **directed cycle**). A circuit (dicycle) of odd (even) length is called **odd** (**even**). A circuit of length three (five) is called a **triangle** (**pentagon**).

A walk (diwalk) that traverses every edge (arc) of a graph (digraph) exactly once is called an **Eulerian trail** (**Eulerian ditrail**). We refer to a closed Eulerian trail (ditrail) as an **Eulerian tour**. An **Eulerian graph** (**Eulerian digraph**) is a graph (digraph) containing an Eulerian tour.

A circuit of length n in a graph of order n is called a **Hamiltonian circuit**. A graph G that contains a Hamiltonian circuit is called **Hamiltonian**. Similarly, a digraph D is called **Hamiltonian** if it contains a Hamiltonian dicycle. Hamiltonian circuits or dicycles are often called (**Hamiltonian**) **tours**.

We shall also use the words "path, circuit, dipath, dicycle, Eulerian tour" to denote the edge or arc set of a path, circuit, dipath, dicycle, Eulerian tour. Thus, whenever we speak of the incidence vector of a circuit etc., we mean the incidence vector of the edge (arc) set of the circuit etc.

A **forest** is an edge set in a graph which does not contain a circuit. A connected forest is called a **tree**. A **spanning tree** of a graph is a tree containing all nodes of the graph. A digraph or arc set which does not contain a dicycle is called **acyclic**.

A **branching** B is an arc set in a digraph D that is a forest such that every node of D is the head of at most one arc of B. A branching that is a tree is called an **arborescence**. A branching that is a spanning tree of D is called a **spanning arborescence** of D. Clearly, in a spanning arborescence B of D every node of D is the head of one arc of B except for one node. This node is called the **root** of B. If r is the root of arborescence B we also say that B is **rooted at** r or r**-rooted**.

Chapter 1

Complexity, Oracles, and Numerical Computation

This chapter is still of a preliminary nature. It contains some basic notions of complexity theory and outlines some well-known algorithms. In addition, less standard concepts and results are described. Among others, we treat oracle algorithms, encoding lengths, and approximation and computation of numbers, and we analyse the running time of Gaussian elimination and related procedures. The notions introduced in this chapter constitute the framework in which algorithms are designed and analysed in this book. We intend to stay on a more or less informal level; nevertheless, all notions introduced here can be made completely precise – see for instance AHO, HOPCROFT and ULLMAN (1974), GAREY and JOHNSON (1979).

1.1 Complexity Theory: $\mathcal{P}$ and $\mathcal{NP}$

Problems

In mathematics (and elsewhere) the word "problem" is used with different meanings. For our purposes, a **problem** will be a general question to be answered which can have several parameters (or variables), whose values are left open. A problem is defined by giving a description of all its parameters and specifying what properties an (optimal) **solution** is required to satisfy. If all the parameters are set to certain values, we speak of an **instance** of the problem.

For example, the open parameters of the linear programming problem (0.1.46) are the $m \times n$-matrix A, and the vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$. If a particular matrix A, and particular vectors c, b are given, we have an instance of the linear programming problem. A solution of an instance of (0.1.46) is one of the following: the statement that $P = \{x \mid Ax \leq b\}$ is empty, the statement that $c^T x$ is unbounded over P, or a vector $x^* \in P$ maximizing $c^T x$ over P.

Two problems of a different sort are the following. Suppose a graph $G = (V, E)$ is given, and we ask whether G contains a circuit, or whether G contains a Hamiltonian circuit. The first problem is called the **circuit problem**, the second one the **Hamiltonian circuit problem**. In both cases the open parameter is the graph G, and the answer is not a vector or a statement as in the LP-problem, but just a "yes" or "no" decision.

These two problems have natural optimization versions. Namely, suppose in addition to the graph $G = (V, E)$ a "length" or "weight" $c_e \in \mathbb{Z}_+$ is given for every edge $e \in E$. The problem of finding a circuit such that the sum of the lengths of the edges of the circuit is as small as possible is called the **shortest circuit problem**. The problem of finding a Hamiltonian circuit of minimum total length is called the (symmetric) **traveling salesman problem** (on G).

To give a formal definition of "problem", we assume that we have an **encoding scheme** which represents each instance of the problem as well as each solution as a string of 0's and 1's. So mathematically, a problem is nothing else than a subset Π of $\{0,1\}^* \times \{0,1\}^*$, where $\{0,1\}^*$ denotes the set of all finite strings of 0's and 1's. Each string $\sigma \in \{0,1\}^*$ is called an **instance** or **input** of Π while a $\tau \in \{0,1\}^*$ with $(\sigma, \tau) \in \Pi$ is called a corresponding **solution** or **output** of Π. We shall assume that, for each instance $\sigma \in \{0,1\}^*$, there exists at least one solution $\tau \in \{0,1\}^*$ such that $(\sigma, \tau) \in \Pi$. (Informally, this only means that if, in its natural setting, an instance of a problem has no solution we declare "no solution" as the solution.)

Algorithms and Turing Machines

In many books or papers it suffices to define an **algorithm** as anything called an algorithm by the authors. In our case, however, algorithms using oracles will play a major role. Since this notion is less standard, we have to go into some details.

Informally, an algorithm is a program to solve a problem. It can have the shape of a sequence of instructions or of a computer program. Mathematically, an algorithm often is identified with some computer model. We want to describe one such model, the (t-**tape**) **Turing machine**. For the reader not familiar with Turing machines it helps to imagine a real-world computer.

The machine consists of a central unit, t tapes, and t heads, where t is some positive integer. Each head is able to read from and write on its tape. The central unit can be in a finite number of states. A tape is a 2-way infinite string of squares (or cells), and each square is either blank or has "0" or "1" written on it. At a particular moment, each head is reading a particular square of its own tape. Depending on what the heads read and on the state of the central unit, each head overwrites or erases the symbol it has read, possibly moves one square to the left or right, and the central unit possibly goes into another state.

At the beginning of the computation, the first tape contains the input string and the first head is reading the first symbol on the tape. The other tapes are blank and the central unit is in a special state B called "beginning state". The computation ends when the central unit reaches another special state E called the "end state". At this moment, the first tape contains the result of the computation (blanks are ignored). So a Turing machine can be described formally as a 6-tuple $T = (X, B, E, \Phi, \Psi, \Xi)$ where:

(i) *X is a finite set of* **states**;

(ii) *B, E are special elements of X, B is the* **beginning state** *and E the* **end state**;

(iii) $\Phi : X \times \{0, 1, *\}^t \to X$ *is the function describing the new state of the central unit;*

(iv) $\Psi : X \times \{0,1,*\}^t \to \{0,1,*\}^t$ *is the function determining the new symbols written on the tapes;*

(v) $\Xi : X \times \{0,1,*\}^t \to \{0,1,-1\}^t$ *is the function determining the movement of the heads.*

(Here "*" stands for blank.) Now an algorithm can be considered as nothing else than a Turing machine T. It **solves** problem $\Pi \subseteq \{0,1\}^* \times \{0,1\}^*$ if, for each string $\sigma \in \{0,1\}^*$, when we give strings $\sigma_1 := \sigma, \sigma_2 := \emptyset, \ldots, \sigma_t := \emptyset$ to T with beginning state B, then, after a finite number of moves of the read-write heads, it stops in state E, while "on tape 1" we have a string $\sigma_1 = \tau$ which is a solution of the problem, i. e., $(\sigma, \tau) \in \Pi$.

There are other computer models that can be used to define an algorithm (like the RAM (= random access machine) or the RASP (= random access stored program machine)), but for our purposes – deciding the polynomial-time solvability of problems – most of them are equivalent.

Encoding

It is obvious that, for almost any imaginable problem, the running time of an algorithm to solve a problem instance depends on the "size" of the instance. The concept of "size" can be formalized by considering an **encoding scheme** that maps problem instances into strings of symbols describing them. The **encoding length** or **input size** (or just **length** or **size**) of a problem instance is defined to be the length of this string of symbols.

Different encoding schemes may give rise to different encoding lengths. For our purposes, most of the standard encoding schemes are equivalent, and for the generality of our results it is not necessary to specify which encoding scheme is used. However, when using encoding lengths in calculations we have to fix one scheme. In these cases we will use the usual binary encoding – details will be given in Sections 1.3 and 2.1.

We would like to point out here that the publication of the ellipsoid method put into focus various controversies about which parameters should be counted in the encoding length. In particular, for an instance of a linear programming problem given by a matrix $A \in \mathbb{Q}^{m \times n}$ and vectors $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, the question is whether the number $n \cdot m$ (of variables times constraints) should be considered as the size of the instance, or whether, in addition, the space needed to encode A, b and c should be counted. Both points of view have their merit. They lead, however, to different notions of complexity.

Time and Space Complexity

Given an encoding scheme and an algorithmic model, the **time complexity function** or **running time function** $f : \mathbb{N} \to \mathbb{N}$ of an algorithm expresses the maximum time $f(n)$ needed to solve any problem instance of encoding length at most $n \in \mathbb{N}$. In the Turing machine model "time" means the number of steps in which the machine reaches the end state E from the beginning state B, for a certain input string σ.

Similarly, the **space complexity function** $g : \mathbb{N} \to \mathbb{N}$ of an algorithm expresses the maximum space $g(n)$ needed to solve any problem instance of encoding length at most $n \in \mathbb{N}$. In the t-tape Turing machine model "space" means the maximum length of strings occuring throughout executing the steps on tape i, summed over $i = 1$ to t.

A **polynomial time (space) algorithm** is an algorithm whose time (space) complexity function $f(n)$ satisfies $f(n) \leq p(n)$ for all $n \in \mathbb{N}$, for some polynomial p.

The main purpose of this book is to derive – with geometric methods – that many interesting problems can be solved by means of a polynomial time algorithm. There are, however, many problems for which such algorithms are not known to exist, and which appear to be much harder.

Decision Problems: The Classes $\mathscr{P}$ and $\mathscr{NP}$

In order to distinguish between "hard" and "easy" problems it is convenient to restrict the notion of problem, and to analyse decision problems only. **Decision Problems** are problems having merely two possible solutions, either "yes" or "no". Decision problems are for instance the circuit problem and the Hamiltonian circuit problem. The class of all those decision problems for which a polynomial time algorithm exists is called the class $\mathscr{P}$.

More exactly, a **decision problem** is a problem Π such that for each $\sigma \in \{0,1\}^*$, exactly one of $(\sigma,0)$, $(\sigma,1)$ belongs to Π, and if $(\sigma,\tau) \in \Pi$ then $\tau \in \{0,1\}$. (0 stands for "no", and 1 for "yes".) The class $\mathscr{P}$ consists of all decision problems that can be solved by a polynomial time algorithm.

An example of a problem in $\mathscr{P}$ is the circuit problem, since the existence of a circuit in a graph $G = (V,E)$ can be checked in $O(|E|)$ time by depth-first search. Recall that a function $f(n)$ is $O(g(n))$, in words "f is of order g", if there is a constant c such that $|f(n)| \leq c|g(n)|$ for all integers $n \geq 0$.

It is by no means the case that all combinatorial problems one encounters are known to be solvable in polynomial time. For example, no polynomial time algorithm is known for the Hamiltonian circuit problem. In fact, it is generally believed that no polynomial time algorithm exists for this problem. Similarly, many other combinatorial problems are expected not to be solvable in polynomial time. Most of these problems have been proved to be equivalent in the sense that if one of them is solvable in polynomial time then the others are as well. In order to sketch the theory behind this, we first describe a class of problems, most probably wider than $\mathscr{P}$, that contains many of the combinatorial problems one encounters.

Informally, this class of problems, denoted by $\mathscr{NP}$, can be defined as the class of decision problems Π with the following property:

If the answer to an instance of Π is in the affirmative, then this fact has a proof of polynomial length.

Note that this definition does not require any method for finding the short proof. For example, if a graph is Hamiltonian then somebody may find a Hamiltonian circuit (by pure luck, intuition, or supernatural power) and mark it with a red

pencil. This yields a proof that the graph is Hamiltonian, and even if we write out all steps of this proof right from the axioms of set theory, its length is polynomial in the size of the graph.

The definition of class $\mathcal{NP}$ is nonsymmetric in "yes" and "no" answers. In order to get the point, and thus to see that the definition of $\mathcal{NP}$ is not pointless, the reader is invited to try to see whether for a non-Hamiltonian graph the nonexistence of a Hamiltonian circuit has a proof of polynomial length.

The example above motivates the following formal definition. $\mathcal{NP}$ consists of all decision problems Π for which there exists a decision problem Σ in $\mathcal{P}$ and a polynomial Φ such that for each $\sigma \in \{0,1\}^*$,

$$(\sigma, 1) \in \Pi \iff \exists\, \tau \in \{0,1\}^* \text{ such that } ((\sigma, \tau), 1) \in \Sigma \text{ and}$$
$$\text{encoding length } (\tau) \leq \Phi(\text{encoding length } (\sigma)).$$

The string τ is called a **succinct certificate** for σ.

A third equivalent way to define the class $\mathcal{NP}$ is via **nondeterministic polynomial time algorithms**; in fact, this is the definition from which the notation $\mathcal{NP}$ is derived. Roughly speaking, these are algorithms in which "guesses" are allowed, provided the correctness of the guess is verified; e. g., we "guess" a Hamiltonian circuit and then verify in polynomial time that the guess was correct. We shall not, however, go into the details of this definition; the reader may find them in the literature cited above.

It is clear that $\mathcal{P} \subseteq \mathcal{NP}$. It also appears natural that $\mathcal{P} \neq \mathcal{NP}$, since nondeterministic algorithms seem to be more powerful than deterministic ones. However, despite enormous research efforts the problem whether or not $\mathcal{P} = \mathcal{NP}$ is still one of the major open problems in mathematics.

The problem obtained by negating the question of a decision problem Π is called the problem **complementary** to Π. That is, the complementary problem to Π is $\{(\sigma, 1-\tau) \mid (\sigma, \tau) \in \Pi\}$. The class of decision problems that are complementary to problems in a class C of decision problems is denoted by co-C. For example, the class complementary to $\mathcal{NP}$ is the class co-$\mathcal{NP}$ which, e. g., contains as a member the problem "Does a given graph G contain no Hamiltonian circuit?".

Trivially $\mathcal{P} =$ co-$\mathcal{P}$, hence $\mathcal{P} \subseteq \mathcal{NP} \cap$ co-$\mathcal{NP}$. The class $\mathcal{NP} \cap$ co-$\mathcal{NP}$ is of particular importance, since for every problem in this class any answer (positive or negative) to an instance has a proof of polynomial length. Following J. Edmonds these problems are called **well-characterized**. Many deep and well-known theorems in combinatorics (Kuratowski's, Menger's, König's, Tutte's) in fact prove that certain problems are in $\mathcal{NP} \cap$ co-$\mathcal{NP}$.

Another outstanding question is whether $\mathcal{P}$ equals $\mathcal{NP} \cap$ co-$\mathcal{NP}$, i. e., can every well-characterized problem be solved in polynomial time. The Farkas lemma (0.1.41) actually yields that the question: "Given a system of linear equations, does it have a nonnegative solution?" is well-characterized. But no polynomial algorithm for this problem was known until 1979, when Khachiyan published his solution using the ellipsoid method. Much of the interest in the ellipsoid method derives from this fact. Several applications of the ellipsoid method treated in this book can be viewed as methods to make use of good characterizations in the design of algorithms.

It is also unknown whether $\mathcal{NP}$ equals co-$\mathcal{NP}$. Note that $\mathcal{NP} \neq$ co-$\mathcal{NP}$ would imply $\mathcal{P} \neq \mathcal{NP}$.

1.2 Oracles

Informally, we imagine an oracle as a device that solves certain problems for us, i. e., that, for any instance σ, supplies a solution τ. We make no assumption on how a solution is found.

An **oracle algorithm** is an algorithm which can "ask questions" from an oracle, and can use the answers supplied. So an oracle algorithm is an algorithm in the usual sense, whose power is enlarged by allowing as further elementary operations the following: if some string $\sigma \in \{0,1\}^*$ has been produced by the algorithm, it may "call the oracle Π_i" with input σ, and obtain a string τ such that $(\sigma, \tau) \in \Pi_i$. A **realization** of an oracle Π is an algorithm solving (problem) Π.

Mathematically, when working in the Turing machine model, this means that we have to enlarge the concept of Turing machine slightly to that of a (***t*-tape**) ***k*-oracle Turing machine**. (The reader not familiar with these concepts may view an oracle Turing machine as a (real-world) computer which can employ satelite computers, or as a computer program having access to subroutines.)

Let $(X, B, E, \Phi, \Psi, \Xi)$ be a Turing machine with t tapes; the last k of these are called **oracle tapes**. On the oracle tapes special "oracles" for problems $\Pi_1, \ldots, \Pi_k$ are sitting. The set X of states contains a special subset $\{x_1, \ldots, x_k\}$. If the central unit is in state x_i, $i \in \{1, \ldots, k\}$, then nothing happens to the heads and tapes except that the string σ on the i-th oracle tape is erased and (miraculously) replaced by some other string τ such that $(\sigma, \tau) \in \Pi_i$; the string τ starts at the position currently read by the corresponding head.

Note that the $(k+1)$-tuple $(T; \Pi_1, \ldots, \Pi_k)$ entirely describes the machine.

We say that such an oracle Turing machine **solves** problem $\Pi \subseteq \{0,1\}^* \times \{0,1\}^*$ if for each $\sigma \in \{0,1\}^*$, when we give strings $\sigma_1 := \sigma, \sigma_2 := \emptyset, \ldots, \sigma_t := \emptyset$ to it with beginning state B, then, whatever answers ρ are given by the oracle when we are in one of the states $x_1, \ldots, x_k$, it stops after a finite number of steps in state E with $\sigma_1 = \tau$ such that $(\sigma, \tau) \in \Pi$.

The Running Time of Oracle Algorithms

The introduction of a running time function for oracle algorithms requires some care. For our purposes, it is convenient to make one general assumption about oracles:

(1.2.1) General Assumption. *We assume that with every oracle we have a polynomial Φ such that for every question of encoding length at most n the answer of the oracle has encoding length at most $\Phi(n)$.*

This assumption is natural for our purposes, since if an algorithm uses an oracle which produces an exponentially long output, then it would take exponential time just to read the output.

Mathematically, this general assumption means that an **oracle** is a pair (Π, Φ), where $\Pi \subseteq \{0,1\}^* \times \{0,1\}^*$ and Φ is a polynomial such that for each $\sigma \in \{0,1\}^*$ of encoding length n, say, there is a $\tau \in \{0,1\}^*$ of encoding length at most $\Phi(n)$ with $(\sigma, \tau) \in \Pi$.

We define the **number of steps** performed by an oracle Turing machine as the number of interations needed to transform the machine from the beginning state B to the end state E, where a call on an oracle as described above counts as one step. (Reading the answer of an oracle may require more than one step.)

Let POL define the collections of polynomials $p : \mathbb{N} \to \mathbb{N}$. Thus we define the **time complexity function** or **running time function** $f : (\mathbb{N} \times (\mathrm{POL})^k) \to \mathbb{N}$ of a k-oracle Turing machine $(T, \Pi_1, \ldots, \Pi_k)$ by: $f(n; \Phi_1, \ldots, \Phi_k)$ is the maximum number of steps performed by the Turing machine when giving any input of encoding length at most n to the machine $(T, \Pi_1 \cap \tilde{\Phi}_1, \ldots, \Pi_k \cap \tilde{\Phi}_k)$, where $\tilde{\Phi}_i := \{(\sigma, \tau) \in \{0,1\}^* \times \{0,1\}^* \mid \text{encoding length } (\tau) \leq \Phi_i(\text{encoding length } (\sigma))\}$. Here the maximum ranges over all possible inputs of encoding length at most n, and over all possible answers given by the oracles $(\Pi_1, \Phi_1), \ldots, (\Pi_k, \Phi_k)$, while executing the algorithm.

It follows from the General Assumption (1.2.1) by a compactness argument that there are only a finite number of possible runs of the algorithm for any given input. Hence the "maximum" in the definition of f is finite.

An oracle algorithm is called **oracle-polynomial**, or just **polynomial** if, for each fixed $\Phi_1, \ldots, \Phi_k \in \mathrm{POL}$, the function $f(n, \Phi_1, \ldots, \Phi_k)$ is bounded by a polynomial in n.

So if we substitute each problem Π_i in a polynomial time oracle Turing machine T by a polynomial time algorithm, we obtain a polynomial time algorithm for the problem solved by T.

Transformation and Reduction

Suppose we have two decision problems Π and Π' and a fixed encoding scheme. A **polynomial transformation** is an algorithm which, given an encoded instance of Π, produces in polynomial time an encoded instance of Π' such that the following holds: For every instance σ of Π, the answer to σ is "yes" if and only if the answer to the transformation of σ (as an instance of Π') is "yes". Clearly, if there is a polynomial algorithm to solve Π' then by polynomially transforming any instance of Π to an instance of Π' there is also a polynomial algorithm to solve Π.

Optimization problems are of course not decision problems. But we can associate decision problems with them in a natural way. Assume a maximization (or minimization) problem, e. g., a linear program, is given. Then we introduce an additional input $Q \in \mathbb{Q}$ and ask "Is there a feasible solution (e. g., to the LP) whose value is at least (or at most) Q?". Supposing there is a polynomial algorithm to solve the optimization problem, we can solve the associated decision problem in the following way. We first compute the optimal solution and its value, then we compare the optimal value with the bound Q and hence are able to solve the decision problem in polynomial time.

Conversely, one can often use a polynomial time algorithm for the associated decision problem to solve a given optimization problem in polynomial time. For example, consider the traveling salesman (optimization) problem and its decision version. Let s and t be the smallest and largest numbers occuring as edge lengths, respectively. Since every tour contains exactly $n = |V|$ edges, the shortest tour cannot be shorter than ns and cannot be longer than nt. Suppose now there is a polynomial time algorithm for the traveling salesman decision problem, then we can ask whether there is a tour of length at most $n(t-s)/2$. If this is the case we ask whether there is a tour of length at most $n(t-s)/4$; if not we ask whether there is a tour of length at most $3n(t-s)/4$, and continue by successively halving the remaining interval of uncertainty. Since the optimum value is integral, the decision problem has to be solved at most $\lceil \log_2(n(t-s)) \rceil + 1$ times. Hence, if the traveling salesman decision problem could be solved in polynomial time, the traveling salesman (optimization) problem could be solved in polynomial time with this so-called **binary search method**.

The polynomial transformation and the binary search method described above are special cases of a general technique. Suppose Π and Π' are two problems. Informally, a **polynomial time Turing reduction** (or just **Turing reduction**) from Π to Π' is an algorithm A which solves Π by using a hypothetical subroutine A' for solving Π' such that, if A' were a polynomial time algorithm for Π', then A would be a polynomial time algorithm for Π.

More precisely, a **polynomial time Turing reduction** from Π to Π' is a polynomial 1-oracle Turing machine $(T;\Pi')$ solving Π. If such a reduction exists, we say that Π can be **Turing reduced** to Π'.

Now we can state some more notions relating the complexity of one problem to that of others.

$\mathcal{NP}$-Completeness and Related Notions

Cook (1971) and Karp (1972) introduced a class of decision problems which are in a well-defined sense the hardest problems in $\mathcal{NP}$. We call a decision problem Π **$\mathcal{NP}$-complete** if $\Pi \in \mathcal{NP}$ and if every other problem in $\mathcal{NP}$ can be polynomially transformed to Π. Thus, every $\mathcal{NP}$-complete problem Π has the following property: If Π can be solved in polynomial time then all $\mathcal{NP}$-problems can be solved in polynomial time, i. e., if Π is $\mathcal{NP}$-complete and if $\Pi \in \mathcal{P}$ then $\mathcal{P} = \mathcal{NP}$. This justifies saying that $\mathcal{NP}$-complete problems are the hardest $\mathcal{NP}$-problems. The Hamiltonian circuit problem, for instance, is known to be $\mathcal{NP}$-complete, and in fact, many of the natural problems coming up in practice are $\mathcal{NP}$-complete – see Garey and Johnson (1979) and the ongoing "$\mathcal{NP}$-Completeness Column" of D. S. Johnson in the Journal of Algorithms for extensive lists of $\mathcal{NP}$-complete problems.

The main significance of the notion of $\mathcal{NP}$-completeness is that it lends a mathematically exact meaning to the assertion that a certain problem is "hard". But the fact that $\mathcal{NP}$-complete problems exist also suggests a way to standardize problems in $\mathcal{NP}$: we may consider every problem in $\mathcal{NP}$ as a special case of, say, the Hamiltonian circuit problem. The usefulness of such a standardization depends on whether or not the natural special features of various problems in

$\mathcal{NP}$ are translated into manageable properties by the reductions. So far, the most succesful $\mathcal{NP}$-complete problem used for such standardization has been the integer programming problem. Various combinatorial problems (e. g., matching, stable set, matroid intersection, etc.), when reduced to integer programming problems, give rise to polyhedra having good properties from the point of view of integer programming. *Polyhedral combinatorics* is the branch of combinatorics dealing with such questions; this will be our main approach to combinatorial optimization problems in the second half of the book.

A problem Π is called **$\mathcal{NP}$-easy** ("not more difficult than some problem in $\mathcal{NP}$") if there is a problem $\Pi' \in \mathcal{NP}$ such that Π can be Turing reduced to Π'. A problem Π is called **$\mathcal{NP}$-hard** ("at least as difficult as any problem in $\mathcal{NP}$") if there is an $\mathcal{NP}$-complete decision problem Π' such that Π' can be Turing reduced to Π. The discussion above shows, for instance, that the traveling salesman problem is $\mathcal{NP}$-easy, and also, that optimization problems are $\mathcal{NP}$-hard if their associated decision problems are $\mathcal{NP}$-complete. In particular, the traveling salesman problem is $\mathcal{NP}$-hard.

An optimization problem that is both $\mathcal{NP}$-hard and $\mathcal{NP}$-easy is called **$\mathcal{NP}$-equivalent**. By definition, if $\mathcal{P} \neq \mathcal{NP}$ then no $\mathcal{NP}$-hard problem can be solved in polynomial time, and if $\mathcal{P} = \mathcal{NP}$ then every $\mathcal{NP}$-easy problem can be solved in polynomial time. Therefore, any $\mathcal{NP}$-equivalent problem can be solved in polynomial time if and only if $\mathcal{P} = \mathcal{NP}$.

Since we do not want to elaborate on the subtle differences between the various problem classes related to the class $\mathcal{NP}$ we shall use the following quite customary convention. In addition to all decision problems that are $\mathcal{NP}$-complete, we call every optimization problem **$\mathcal{NP}$-complete** for which the associated decision problem is $\mathcal{NP}$-complete.

To close this section we would like to remark that Turing reductions will be the most frequent tools of our complexity analysis. We shall often show that a polynomial algorithm A for one problem implies the existence of a polynomial time algorithm for another problem by using A as an oracle.

1.3 Approximation and Computation of Numbers

The introduction to complexity theory in the foregoing two sections has made some informal assumptions which we would like to discuss now in more detail. One important assumption is that all instances of a problem can be encoded in a finite string of, say, 0's and 1's.

Encoding Length of Numbers

For integers, the most usual encoding is the binary representation, and we will use this encoding, unless we specify differently. To encode an integer $n \neq 0$, we need one cell (or bit) for the sign and $\lceil \log_2(|n| + 1) \rceil$ cells for the $\{0, 1\}$-string of the binary representation of its absolute value. For 0, only one cell is needed.

Hence, the space needed to encode an integer is

$$\langle n \rangle := 1 + \lceil \log_2(|n| + 1) \rceil, \quad n \in \mathbb{Z}, \tag{1.3.1}$$

and we call $\langle n \rangle$ the **encoding length** (or the **input size** or just **size**) of n. If we say that an algorithm computes an integer, we mean that its output is the binary encoding of this integer.

Every rational number r can be written uniquely as p/q with $q > 0$ and p and q coprime integers; so the space needed to encode r is

$$\langle r \rangle := \langle p \rangle + \langle q \rangle, \tag{1.3.2}$$

and we call this the **encoding length** of r. Similarly, if x is a rational vector or matrix then the **encoding length** $\langle x \rangle$ is defined as the sum of the encoding lengths of its entries. If $a^T x \leq b$ is an inequality with $a \in \mathbb{Q}^n$ and $b \in \mathbb{Q}$ then we say that $\langle a \rangle + \langle b \rangle$ is the **encoding length of the inequality**; and the **encoding length of an inequality system** is the sum of the encoding lengths of its inequalities. To shorten notation, for any sequence $a_1, a_2, \ldots, a_n$ of matrices and vectors, we will write $\langle a_1, \ldots, a_n \rangle$ to denote the sum of the encoding lengths of $a_1, \ldots, a_n$. In particular, we write $\langle A, b, c \rangle$ to denote the **encoding length of the linear program** $\max\{c^T x \mid Ax \leq b\}$.

There are some useful relations between encoding lengths and other functions of vectors and matrices.

(1.3.3) Lemma.

(a) *For every rational number* r, $|r| \leq 2^{\langle r \rangle - 1} - 1$.
(b) *For every vector* $x \in \mathbb{Q}^n$, $\|x\| \leq \|x\|_1 \leq 2^{\langle x \rangle - n} - 1$.
(c) *For every matrix* $D \in \mathbb{Q}^{n \times n}$, $|\det D| \leq 2^{\langle D \rangle - n^2} - 1$.

Proof. (a) follows directly from the definition. To prove (b), let $x = (x_1, \ldots, x_n)^T$. By (0.1.6) $\|x\| \leq \|x\|_1$. Then by (a)

$$1 + \|x\|_1 = 1 + \sum_{i=1}^{n} |x_i| \leq \prod_{i=1}^{n} (1 + |x_i|) \leq \prod_{i=1}^{n} 2^{\langle x_i \rangle - 1} = 2^{\langle x \rangle - n}.$$

To prove (c), let $d_1, \ldots, d_n$ be the rows of D. Then by Hadamard's inequality (0.1.28) and (b):

$$1 + |\det D| \leq 1 + \prod_{i=1}^{n} \|d_i\| \leq \prod_{i=1}^{n} (1 + \|d_i\|) \leq \prod_{i=1}^{n} 2^{\langle d_i \rangle - n} = 2^{\langle D \rangle - n^2}.$$

□

(1.3.4) Lemma.

(a) *For* $r, s \in \mathbb{Q}$, $\langle rs \rangle \leq \langle r \rangle + \langle s \rangle$.
(b) *For every matrix* $D \in \mathbb{Q}^{n \times n}$, $\langle \det D \rangle \leq 2\langle D \rangle - n^2$,
and if $D \in \mathbb{Z}^{n \times n}$ *then* $\langle \det D \rangle \leq \langle D \rangle - n^2 + 1$.

Proof. (a) is trivial. The second statement of (b) follows immediately from (1.3.3) (c). To prove the first statement of (b), let $D = (p_{ij}/q_{ij})_{i,j=1,\ldots,n}$ be such that p_{ij} and $q_{ij} \geq 1$ are comprime integers for $i, j = 1, \ldots, n$. The same argument as in the proof of Lemma (1.3.3) shows that

$$|\det D| \leq 2^{\sum_{i,j=1}^{n} \langle p_{ij} \rangle - n^2} - 1.$$

Let $Q = \prod_{i,j=1}^{n} q_{ij}$. Then $\det D = (Q \det D)/Q$, where $Q \det D$ and Q are integers. Hence

$$\begin{aligned}
\langle \det D \rangle &\leq \langle Q \det D \rangle + \langle Q \rangle \\
&= 1 + \lceil \log_2(|Q \det D| + 1) \rceil + \langle Q \rangle \\
&\leq 1 + \lceil \log_2(1 + Q(2^{\sum_{i,j=1}^{n} \langle p_{ij} \rangle - n^2} - 1)) \rceil + \langle Q \rangle \\
&\leq 1 + \lceil \log_2 Q + \sum \langle p_{ij} \rangle - n^2 \rceil + \langle Q \rangle \\
&\leq 2\langle Q \rangle + \sum \langle p_{ij} \rangle - n^2 \\
&\leq 2\langle D \rangle - n^2.
\end{aligned}$$

□

(1.3.5) Exercise.

(a) *Prove that for any* $z_1, \ldots, z_n \in \mathbb{Z}$:

$$\langle z_1 + \ldots + z_n \rangle \leq \langle z_1 \rangle + \ldots + \langle z_n \rangle.$$

(b) *Prove that for any* $r_1, \ldots, r_n \in \mathbb{Q}$:

$$\langle r_1 + \ldots + r_n \rangle \leq 2(\langle r_1 \rangle + \ldots + \langle r_n \rangle).$$

(c) *Show that the coefficient* 2 *in Lemma* (1.3.4) (b) *and in* (b) *above cannot be replaced by any smaller constant.*
(d) *Prove that if* $A \in \mathbb{Q}^{n \times n}$ *is nonsingular then*

$$\langle A^{-1} \rangle \leq 4n^2 \langle A \rangle.$$

(e) *Prove that if* $A \in \mathbb{Q}^{m \times n}$, $B \in \mathbb{Q}^{n \times p}$ *then*

$$\langle AB \rangle \leq p\langle A \rangle + m\langle B \rangle.$$

□

Polynomial and Strongly Polynomial Computations

We will now discuss our model of computing with numbers in more detail. We assume that every integral or rational number is represented in binary notation in the way described above. Then, in the Turing machine model, an arithmetic operation like addition of two integers takes a number of steps (moves of the read-write head) which is bounded by a polynomial in the encoding lengths of the two integers. Unless we state a different model we mean this model of computation throughout the book.

It is often more natural and realistic (cf. real-life computers) to consider the **elementary arithmetic operations**:

addition, subtraction, multiplication, division, comparison,

rather than the number of moves of a head of some fictitious Turing machine, as one step. If we count elementary arithmetic operations, we say that we analyse an algorithm in the **arithmetic model**. For instance, computing the product of two $n \times n$-matrices in the standard way takes $0(n^3)$ steps in the arithmetic model, while in the Turing machine model its running time depends on the encoding lengths of the entries of the matrix. In the arithmetic model, the **encoding length** of an instance of a problem is the number of numbers occuring in the input (so we do not count the encoding length of the input numbers). If the input has some nonnumerical part (e. g., a graph or a name) we assume that it is encoded as a $\{0,1\}$-sequence. Each entry of this sequence is considered a number. Correspondingly, we consider nonnumeric steps in the algorithm (like setting or removing a label, deleting an edge from a graph) as arithmetic operations.

The running time of an algorithm may be bounded by a polynomial in the Turing machine model but not in the arithmetic model, and vice versa. For instance, the well-known Euclidean algorithm to compute the greatest common divisor of two integers runs in polynomial time in the Turing machine model but not in the arithmetic model. On the other hand, the algorithm that reads n numbers and computes 2^{2^n} by repeated squaring is polynomial in the arithmetic model but not in the Turing model since the number of digits of the result is exponentially large.

It is easy to see that the elementary arithmetic operations listed above can be executed on a Turing machine in polynomial time. Using this we can further develop the idea of counting arithmetic operations (instead of moves of a read-write head) if we supplement it by verifying that the number of digits of the numbers occuring during the run of the algorithm is bounded by a polynomial in the encoding length. Then, of course, a polynomial number of arithmetic operations can be executed in polynomial time on a Turing machine.

We say that an algorithm runs in **strongly polynomial time** if the algorithm is a polynomial space algorithm and performs a number of elementary arithmetic operations (addition, subtraction, multiplication, division, comparison) which is bounded by a polynomial in the number of input numbers. So a strongly polynomial algorithm is a polynomial space algorithm (in our standard Turing machine model) and a polynomial time algorithm in the arithmetic model. The

notion of strong polynomiality involves some subtleties which we will discuss in more detail now.

Since the definition of strong polynomiality mixes the arithmetic and the Turing model in some sense we have to make precise how numbers and the results of arithmetic operations are encoded. Let us begin with integers. Integers are encoded in the usual binary notation. Adding, subtracting, multiplying and comparing two integers does not cause any problems. However, we have to specify what dividing two integers means. There are at least four alternatives to define the result of the division "$a : b$" for $a, b \in \mathbb{Z}$, $b \neq 0$:

- the rational number a/b,
- the rational number a/b, and if it is known in advance that a/b is an integer, the integer a/b,
- the rational number a/b, and if a/b is an integer, the integer a/b (so this operation involves a routine testing whether b divides a),
- the integer $\lfloor a/b \rfloor$ (this is equivalent to division with remainder).

We reduce the arithmetic for rational numbers to that for integers by setting

$$\frac{a}{b} + \frac{c}{d} := \frac{ad+bc}{bd}, \quad \frac{a}{b} - \frac{c}{d} := \frac{ad-bc}{bd},$$
$$\frac{a}{b} \cdot \frac{c}{d} := \frac{ac}{bd}, \quad \frac{a}{b} : \frac{c}{d} := \frac{ad}{bc}.$$

It is important to note that in this model of computation we do not assume that a rational number is represented in coprime form. The reason is that a rational number, for instance coming up as a result of a computation as above, may not be in its coprime form. To bring it into coprime form is a polynomial but not a strongly polynomial procedure.

The four versions of division of integers mentioned above lead to increasingly more powerful models of computation. However, it has to be pointed out that a strongly polynomial algorithm – using any of these kinds of division – is always a polynomial time algorithm in the Turing machine model.

We regard the first version of division as too weak. Namely, consider the following example. Given $a \in \mathbb{Z}$, $a \neq 0$, square a/a repeatedly to obtain a^{2^n}/a^{2^n}. Even though the result is just 1, the first version of division does not allow us to get rid of many factors of a. Throughout the book we will adopt the second notion of division. It will allow us to turn most of the polynomial time algorithms (in the Turing machine sense) into strongly polynomial ones. However, there are a few examples where we could obtain a strongly polynomial algorithm only in the sense of making use of the fourth version of division. On the other hand, we will describe a number of strongly polynomial algorithms which could be described in such a way that no multiplication or division is performed at all.

Polynomial Time Approximation of Real Numbers

Since there is no encoding scheme with which all irrational numbers can be represented by finite strings of 0's and 1's it is customary to accept only rational

numbers as inputs or outputs of algorithms. It will, however, be convenient to speak about certain irrational numbers as inputs and outputs. This is done by making use of the fact that irrational numbers can be approximated by rationals. To make this precise, we have to discuss how errors of approximations can be measured.

We say that a rational number r approximates the real number ρ with **absolute error** $\varepsilon > 0$ if

$$|\rho - r| \leq \varepsilon$$

holds, and we say that a rational number r approximates ρ with **relative error** $\varepsilon > 0$ if

$$|\rho - r| \leq \varepsilon|\rho|.$$

These two kinds of error measurements are in general not equivalent. In particular, statements about relative errors are very sensitive with respect to addition of constants, while statements about absolute errors are sensitive with respect to multiplication by constants.

Given some input, we shall say that a real number ρ is **polynomially computable with absolute (relative) error** from this input if for every rational $\varepsilon > 0$ a rational number r can be computed in time polynomial in the encoding length of the input and in $\langle\varepsilon\rangle$ that approximates ρ with absolute (relative) error ε. An algorithm accomplishing this will be called a **polynomial computation algorithm with absolute (relative) error**.

We now want to show that a polynomial computation algorithm with relative error can be used to obtain a polynomial computation algorithm with absolute error. Suppose we have a polynomial computation algorithm A which guarantees a relative error and we want to find a polynomial computation algorithm B which guarantees an absolute error. Assume $\varepsilon > 0$ is the absolute error we want to achieve. Then we call B with relative error $\delta_1 := 1/2$ and obtain a rational number r_1 satisfying $|\rho - r_1| \leq \frac{1}{2}|\rho|$, and hence $|\rho| \leq 2|r_1|$. If $r_1 = 0$, then $\rho = 0$ and we are finished. Since B is a polynomial computation algorithm, $\langle r_1\rangle$ is bounded by a polynomial of the original encoding length. Now we run B again with relative error $\delta_2 := \varepsilon/(2|r_1|)$ which yields a rational number r_2 satisfying $|\rho - r_2| \leq \varepsilon|\rho|/(2|r_1|) \leq \varepsilon$ and obtain the desired polynomial computation with absolute error ε.

The reverse implication is not true in general: small numbers cause problems. If we assume, however, that ρ is nonzero and that a positive lower bound on $|\rho|$ can be computed in polynomial time, say $0 < b \leq |\rho|$, then the existence of a polynomial computation algorithm A with absolute error implies the existence of a polynomial computation algorithm B with relative error as follows. We simply call A with absolute error $\delta := \varepsilon b$ and obtain a rational number r satisfying $|\rho - r| \leq \varepsilon b \leq \varepsilon|\rho|$.

From now on we call a real number ρ **polynomially computable from a given input** if it is polynomially computable with relative error. This is equivalent to the existence of a polynomial computation algorithm with absolute error, together with a polynomial auxiliary algorithm that determines whether or not $\rho = 0$ and if not, computes a positive lower bound for $|\rho|$. In fact, it suffices to have a polynomial auxiliary algorithm that computes a positive rational b such that

either $\rho = 0$ or $|\rho| \geq b$. Then whether or not $\rho = 0$ can be decided by running the computation algorithm with absolute error $b/3$.

If the number ρ to be computed is known to be an integer, then the existence of a polynomial computation algorithm with absolute error (and hence with relative error) means that ρ can be computed exactly in polynomial time. We only have to run the algorithm with absolute error $1/3$ and round the result to the nearest integer.

More generally, if ρ is a rational number polynomially computable from a given input and an upper bound for its denominator can be polynomially computed, then ρ can be computed exactly in polynomial time. This can be done in a manner similar to that above, but for the final rounding one must use the technique of continued fractions – see Section 5.1.

Sometimes an approximation algorithm with relative error ε can be obtained whose running time is bounded by a polynomial in $1/\varepsilon$ (but not necessarily in $\langle\varepsilon\rangle$) and the encoding length of the original input. Such an algorithm is called a **fully polynomial approximation algorithm** (or **scheme**). An even weaker notion is an algorithm that is polynomial in the original encoding length for every fixed $\varepsilon > 0$. Such an algorithm is called a **polynomial approximation algorithm** (or **scheme**).

An integer (depending on some input) for which a fully polynomial approximation algorithm exists is not necessarily computable in polynomial time (at least if $\mathcal{P} \neq \mathcal{NP}$). In fact, a fully polynomial approximation algorithm exists for the $\mathcal{NP}$-complete knapsack problem (IBARRA and KIM (1975)). We remark that the well-known Agmon-Motzkin-Schönberg relaxation method yields a fully polynomial approximation algorithm, but not a polynomial algorithm, for the optimum value of a linear program.

If ρ is a real number for which upper and lower bounds are polynomially computable, and if we have a polynomial time algorithm to decide whether or not a rational number r is smaller than ρ, then binary search can be used to obtain a polynomial computation algorithm with absolute error for ρ. (The simple-minded way of finding an approximation of ρ with absolute error ε by searching the given interval in steps of length ε would yield just a fully polynomial approximation algorithm.)

As an illustration we mention that to approximate the irrational number

$$\zeta(3) = \sum_{n=1}^{\infty} \frac{1}{n^3} (= 1.2020569031\ldots)$$

by the partial sums is a fully polynomial approximation algorithm (the absolute error of the sum of the first k terms is about $1/(2k^2)$). However, if we use the expansion

$$\zeta(3) = \frac{5}{2} \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^3 \binom{2n}{n}},$$

due to R. Apéry – see VAN DER POORTEN (1979) – then we get a polynomial computation algorithm. The difference between these two notions is illuminated by the fact that, to obtain the above 10 digits behind the point, about 100,000

terms of the first expansion have to be added up, while only 14 terms of the second suffice.

1.4 Pivoting and Related Procedures

We shall now describe four fundamental matrix manipulation algorithms in linear algebra, linear programming, and number theory, and we shall prove that three of them have polynomial running time. The running time analyses of these algorithms will show some of the basic techniques used in the sequel.

Gaussian Elimination

Perhaps the most frequently used algorithm in linear algebra is the Gaussian elimination method. This algorithm is mainly employed for solving linear equations but it can also be viewed as a method that transforms a matrix into some "canonical" form. It consists of a successive application of an iteratively applied forward step and an iteratively applied backward step. Let us describe it as a matrix manipulation algorithm.

Suppose an $m \times n$-matrix A is given. We would like to transform A into an $m \times n$-matrix of the following form

$$\begin{pmatrix} \Delta & D \\ 0 & 0 \end{pmatrix}, \tag{1.4.1}$$

where Δ is a diagonal $r \times r$-matrix and r is the rank of A, using the following operations only:

(1.4.2)
(i) adding a multiple of one row to another row,
(ii) permuting rows or columns.

The Gaussian elimination method works as follows. For any given $m \times n$-matrix A, we first find matrices $A_0, A_1, \ldots, A_r$ where each of the matrices A_k, $0 \leq k \leq r$, has the form

$$A_k = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix} \tag{1.4.3}$$

and where B is a nonsingular upper triangular $k \times k$-matrix. We start with $A_0 := A$ (so B is an empty matrix). The matrix A_{k+1} is determined from A_k in the following way :

(1.4.4) Forward Step. *Given a matrix A_k of form* (1.4.3), *choose a nonzero element of D, called the* **pivot element**, *and permute the rows and columns of A_k so that this pivot element is d_{11}. Now add multiples of the first row of D to the*

other rows of D in such a way that d_{11} becomes the only nonzero element in the first column of D. So

$$d_{ij} := d_{ij} - \frac{d_{i1}}{d_{11}} d_{1j} \quad \text{for } i = 1, \ldots, m-k;\ j = 1, \ldots, n-k.$$

The resulting matrix is called A_{k+1}. □

The Forward Step (1.4.4) is applied until D contains no nonzero element. So the last matrix, say A_r, in this sequence has the form $A_r = \begin{pmatrix} B & C \\ 0 & 0 \end{pmatrix}$, with B an upper triangular $r \times r$-matrix. It is obvious that r is the rank of A.

Now we go backwards. Starting with $E_r := A_r$, matrices $E_r, E_{r-1}, \ldots, E_0$ are determined which have the form

$$E_k = \begin{pmatrix} B & 0 & C \\ 0 & \Delta & D \\ 0 & 0 & 0 \end{pmatrix}, \tag{1.4.5}$$

such that B is a nonsingular upper triangular $k \times k$-matrix and Δ is a diagonal $(r-k) \times (r-k)$-matrix. Given the matrix E_k, the matrix E_{k-1} is determined with the following procedure.

(1.4.6) Backward Step. *Add multiples of the k-th row of E_k to the other rows of E_k so that b_{kk} will be the only nonzero entry in the k-th column of E_k. The matrix obtained in this way is called E_{k-1}.* □

After r applications of the Backward Step (1.4.6) we obtain the matrix E_0 which – by construction – is of form (1.4.1).

It is obvious from the descriptions of the Forward Step (1.4.4) and the Backward Step (1.4.6) that the number of elementary arithmetic operations (additions, subtractions, multiplications, divisions, comparisons) of the method is bounded by a polynomial in n and m. For instance, if $n = m$, then $\mathcal{O}(n^2)$ divisions and $\mathcal{O}(n^3)$ additions and multiplications have to be performed. To show polynomiality, one only has to prove that none of the numbers calculated during the execution of the algorithm becomes too large. This has been done by EDMONDS (1967b)). In fact, Edmonds showed that Gaussian elimination is strongly polynomial in the sense described in Section 1.3. To be precise, we have to specify in which form the rational numbers occuring are stored and which divisions are carried out as integer divisions (recall the four divisions "$a : b$" mentioned in 1.3).

One version would be to present each rational number in its coprime form and bringing each update into this form. This procedure is not strongly polynomial, but polynomial as will follow from the discussion below. Another possibility is to store each rational number in the form it is computed. This is polynomial in the arithmetic model but it leads to an exponential growth in the encoding lengths.

To see this, let us assume that the initial matrix A is an integral $n \times n$-matrix. (If A is not integral, we multiply all entries of A by the product of the denominators

of all entries.) After the k-th step, the entries d_{ij} of the $(n-k)\times(n-k)$-matrix D in (1.4.3) can be represented in the form

$$d_{ij} = \frac{f_{ij}^{(k)}}{g^{(k)}},$$

where $g^{(k)}$ is a common denominator for all d_{ij}. The update formula in (1.4.4) yields that we can take

$$f_{i-1,j-1}^{(k+1)} = f_{ij}^{(k)} f_{11}^{(k)} - f_{i1}^{(k)} f_{1j}^{(k)}, \quad g^{(k+1)} = f_{11}^{(k)} g^{(k)}.$$

These formulas are not good for our purposes. Namely if we start the procedure with the $n\times n$-matrix A with 2's on the main diagonal, 1's below the main diagonal, and 0's above the main diagonal and we always pivot on the elements of the main diagonal then, for $k = 0, 1, \ldots, n-1$, we get

$$f_{11}^{(k)} = 2^{2^k}, \; g^{(k)} = 2^{2^k-1}.$$

Clearly, it takes exponential space to represent these numbers in binary.

To achieve strong polynomiality we use a third way of encoding the rationals occuring. This is based on an analysis of the update formula. We assume – as stated before – that A is an integral $m\times n$-matrix. For $k = 0, 1, \ldots, m-1$, we write each entry d_{ij} of D – see (1.4.3) – in the form

$$d_{ij} = \frac{p_{ij}^{(k)}}{q^{(k)}}, \tag{1.4.7}$$

where $q^{(k)} = \det B$. We claim that $p_{ij}^{(k)}$ and $q^{(k)}$ are integers, in fact, they are subdeterminants of A. Let us assume for simplicity that we have pivoted along the first k elements of the main diagonal. Setting $K := \{1, \ldots, k\}$, $I := \{1, \ldots, k, k+i\}$, $J := \{1, \ldots, k, k+j\}$, we see that $(A_k)_{KK} = B$ and that $(A_k)_{IJ}$ is an upper triangular matrix with the entry d_{ij} of D in the lower right hand corner. Thus

$$d_{ij} = \frac{\det((A_k)_{IJ})}{\det((A_k)_{KK})}.$$

Now recall that the determinant of a matrix does not change if a multiple of one row is added to another. Since A_k arises from A by adding multiples of the first k rows of A to the other rows of A, we observe that $\det(A_{IJ}) = \det\big((A_k)_{IJ}\big)$ and $\det(A_{KK}) = \det\big((A_k)_{KK}\big)$ hold. Therefore $p_{ij}^{(k)} = \det(A_{IJ})$ and $q^{(k)} = \det(A_{KK})$, and hence

$$d_{ij} = \frac{\det(A_{IJ})}{\det(A_{KK})}.$$

By Lemma (1.3.4), the encoding of d_{ij} as the ratio of $p_{ij}^{(k)}$ and $q^{(k)}$ takes only polynomial space. Hence the coprime representation of d_{ij} takes only polynomial

space, and therefore the version of the forward step using coprime representations of all rationals can be executed in polynomial time.

Now we show that, using representation (1.4.7), the updates can be performed in strongly polynomial time. In fact, an easy computation shows that the update formulas are the following

$$p_{ij}^{(k+1)} = \frac{p_{ij}^{(k)} p_{11}^{(k)} - p_{i1}^{(k)} p_{1j}^{(k)}}{q^{(k)}},$$
$$q^{(k+1)} = p_{11}^{(k)}.$$

Since we know from the above discussion that $p_{ij}^{(k+1)}$ is an integer, we can obtain it by an integer division.

To analyse the Backward Step (1.4.6), view E_k, $0 \leq k \leq r$, as a matrix of form (1.4.5). Subdivide the matrix A_r accordingly, say

$$A_r = \begin{pmatrix} B_1 & B_2 & C_1 \\ 0 & B_3 & C_2 \\ 0 & 0 & 0 \end{pmatrix}.$$

B_1 and B_3 are upper triangular matrices of order $k \times k$ and $(r-k) \times (r-k)$, respectively. Note that by construction $B_1 = B$, and that the diagonal entries of B_3 are the same as those of Δ. So – by the previous part of the proof – we only have to show that the entries of C and D are small. Let us write the $r-k$ executions of the Backward Step (1.4.6) to obtain E_k from $A_r = E_r$ in compact matrix form. We see that

$$E_k = \begin{pmatrix} I & -B_2 B_3^{-1} \\ 0 & \Delta B_3^{-1} \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} B_1 & B_2 & C_1 \\ 0 & B_3 & C_2 \end{pmatrix} = \begin{pmatrix} B_1 & 0 & C_1 - B_2 B_3^{-1} C_1 \\ 0 & \Delta & \Delta B_3^{-1} C_2 \\ 0 & 0 & 0 \end{pmatrix},$$

and so $C = C_1 - B_2 B_3^{-1} C_1$, $D = \Delta B_3^{-1} C_2$. Since B_2, B_3, C_1 and C_2 are submatrices of A_r their encoding lengths are bounded by a polynomial in $\langle A \rangle$. The same holds for Δ as remarked above. Since the entries of the inverse of a matrix are quotients of subdeterminants of the original matrix, we can conclude that the encoding length of each entry of B_3^{-1} is bounded by a polynomial in $\langle A \rangle$ – see also (1.3.5) (d). Therefore, the encoding lengths of C and D are bounded by a polynomial in $\langle A \rangle$. This shows that the backward step can be performed in polynomial time using coprime representation for all rationals occuring. By a similar analysis as carried out above for the forward step, one can show that the backward step can also be executed in strongly polynomial time. Thus we have Edmonds' theorem:

(1.4.8) Theorem. *For any rational $m \times n$-matrix A, the Gaussian elimination method (using coprime representation of rationals) runs in time polynomial in $\langle A \rangle$. There is a representation scheme for rationals so that the Gaussian elimination method runs in strongly polynomial time.* □

Let us remark that the general results of STRASSEN (1973) for eliminating division imply that $\det A$ can be computed in strongly polynomial time without any division – see also BERKOWITZ (1984).

The fact that Gaussian elimination runs in polynomial time implies the polynomial time solvability of a number of basic problems in linear algebra. In fact, (1.4.8) shows that these problems are also solvable in strongly polynomial time.

(1.4.9) Corollary. *There are strongly polynomial algorithms for the following problems:*

(a) *finding a solution of a rational equation system* $Ax = b$,
(b) *determining the rank of a rational matrix,*
(c) *determining the determinant of a rational matrix,*
(d) *determining the inverse of a nonsingular rational matrix,*
(e) *testing whether* m *given rational vectors are linearly independent.* □

Gram-Schmidt Orthogonalization

The forward step of the Gaussian elimination method transforms a given nonsingular matrix into an upper triangular matrix, using only row operations. We briefly mention another very useful algorithm, the **Gram-Schmidt orthogonalization procedure**, which transforms any nonsingular matrix into one whose column vectors are orthogonal, using only column operations.

If $(b_1, \ldots, b_n)$ is an ordered basis of $\mathbb{R}^n$ then its **Gram-Schmidt orthogonalization** $(b_1^*, \ldots, b_n^*)$ is another ordered basis of $\mathbb{R}^n$ defined by the recurrence

$$b_j^* := b_j - \sum_{i=1}^{j-1} \frac{b_j^T b_i^*}{\|b_i^*\|^2} b_i^* \quad \text{for } j = 1, \ldots, n.$$

This formula implies that, for every $j \in \{1, \ldots, n\}$, the vectors $b_1, \ldots, b_n$ and $b_1^*, \ldots, b_n^*$ span the same subspace of $\mathbb{R}^n$ and that b_{j+1}^* is the "component" of b_{j+1} orthogonal to this subspace. For orthogonal vectors, the Hadamard inequality (0.1.28) holds with equality, and thus

$$\|b_1^*\| \cdot \ldots \cdot \|b_1^*\| = |\det(b_1^*, \ldots, b_n^*)| = |\det(b_1, \ldots, b_n)|,$$

where the last equation follows from the fact that $b_1^*, \ldots, b_n^*$ is obtained from $b_1, \ldots, b_n$ by column operations. In fact, observe that, to obtain $(b_1^*, \ldots, b_n^*)$ from $(b_1, \ldots, b_n)$, a multiple of a column is only added to a column with larger index. Moreover, it is also trivial from the definition that each vector b_j can be expressed as a linear combination of the vectors $b_1^*, \ldots, b_j^*$ as follows:

$$b_j = \sum_{i=1}^{j} \mu_{ji} b_i^*,$$

and that in this formula $\mu_{jj} = 1$.

The running time analysis of the Gram-Schmidt orthogonalization procedure is similar to that of the Gaussian elimination method and is omitted here. Gram-Schmidt orthogonalization will be used in Section 5.3.

The Simplex Method

The simplex method for solving linear programs, due to DANTZIG (1951), is one of the backbones of mathematical programming. It is usually described for linear programs in so-called **standard form**: $\max c^T x$, $Ax = b$, $x \geq 0$. We will outline it for linear programs of the form

$$\max c^T x \quad Ax \leq b. \tag{1.4.10}$$

Let us assume first that A has full column rank. In this case the polyhedron $P = \{x \mid Ax \leq b\}$ has a vertex. Now, roughly speaking, the simplex method starts at a vertex of P and checks whether there is an edge containing this vertex along which the objective function can be increased. If this is not so, the current vertex is an optimum solution of (1.4.10). If there are such edges, we go in the direction of one of these edges. Either we find a new vertex, in which case we repeat the procedure, or the edge is infinite, in which case (1.4.10) is unbounded. The technical problem is to identify edges along which the objective function can be increased. We describe how this can be done.

Let us call every row index set I of cardinality n for which $A_{I.}$ is a basis of A and $A_{I.}^{-1} b_I$ satisfies $Ax \leq b$ a **basic feasible index set**. We know that for every vertex v of P there is at least one basic feasible index set I such that $v = A_{I.}^{-1} b_I$. Instead of creating a sequence of vertices as described above, the simplex method constructs a sequence $I_1, I_2, \ldots$ of basic feasible index sets. In the corresponding sequence $A_{I_1.}^{-1} b_{I_1}, A_{I_2.}^{-1} b_{I_2}, \ldots$ of vertices of P two consecutive vertices of this sequence need not be different.

We now describe one iteration of the simplex method. Suppose we have a basic feasible index set I with corresponding vertex $v = A_{I.}^{-1} b_I$. Calculate

$$u^T := c^T A_{I.}^{-1}. \tag{1.4.11}$$

Note that u is an n-vector indexed by I. We distinguish two cases.

Case 1. $u \geq 0$. Then v is an optimum solution of (1.4.10) because for each x satisfying $Ax \leq b$ we have

$$c^T x = u^T A_{I.} x \leq u^T b_I = u^T A_{I.} v = c^T v.$$

Case 2. $u \not\geq 0$. Choose $i \in I$ such that $u_i < 0$. Define the following "direction"

$$d := -A_{I.}^{-1} e_i, \tag{1.4.12}$$

where e_i is the unit basis vector in $\mathbb{R}^I$ with 1 in coordinate i. Let λ^* be the largest real number λ so that

$$A(v + \lambda d) \leq b.$$

Case 2a. λ^* is finite. This is equivalent to: $Ad \not\leq 0$. Clearly, λ^* can be computed as follows:

$$\lambda^* = \min\left\{\frac{b_j - A_{j.}v}{A_{j.}d} \mid j = 1, \ldots, m \text{ with } A_{j.}d > 0\right\}. \tag{1.4.13}$$

Let this minimum be attained at index k. Note that $k \notin I$ because $A_{I.}d = -e_i \leq 0$. As new basic feasible index set we choose $I' := (I \setminus \{i\}) \cup \{k\}$, which corresponds to the vertex $v' := v + \lambda^* d$. We replace I by I' and repeat the iteration.

Case 2b. λ^* is infinite. This is equivalent to: $Ad \leq 0$. So d belongs to the recession cone of P and, moreover,

$$c^T d = -c^T A_{I.}^{-1} e_i = -u^T e_i = -u_i > 0.$$

Thus the objective function can be increased along d to infinity which shows that (1.4.10) is unbounded.

It can be proved that by appropriately choosing indices i and k (such choices are called **pivoting rules**) this method terminates. For instance, BLAND (1977) proved that the pivoting rule "always choose the smallest i such that $u_i < 0$ and the smallest k attaining the minimum in (1.4.13)" guarantees finite termination. This finishes the description of the simplex method.

Unfortunately, for almost all known pivoting rules, sequences of examples have been constructed – see for instance KLEE and MINTY (1972) – such that the number of iterations is exponential in $n+m$. No pivoting rule is known – to date – to yield a polynomial time method. In practice, however, the simplex method turns out to be very fast. Moreover, BORGWARDT (1982) proved that in a certain natural probabilistic model and with the so-called "Schattenecken" pivoting rule, the expected running time of the simplex method is polynomial – see also SMALE (1983).

The observed practical success of the simplex method is based on skillful pivoting rules which empirically lead to small numbers of iterations and on the fact that each iteration can be performed efficiently. Note that the total work of one iteration is dominated by the determination of the inverse matrix $A_{I.}^{-1}$. This inverse, however, does not have to be computed from scratch in each iteration. The inverse $A_{I'.}^{-1}$ needed in the next iteration can be computed from the inverse $A_{I.}^{-1}$ of the previous iteration with a simple **pivoting** operation, namely we have

$$\begin{aligned} \left(A_{I'.}^{-1}\right)_{.I'\setminus\{k\}} &= \left(I - \frac{1}{A_{k.}d} dA_{k.}\right)\left(A_{I.}^{-1}\right)_{.I\setminus\{i\}}, \\ \left(A_{I'.}^{-1}\right)_{.k} &= \frac{1}{A_{k.}d} d. \end{aligned} \tag{1.4.14}$$

These updates can be executed in practice very efficiently.

We still have to show how the assumptions we made for describing the simplex method can be achieved. To check whether the matrix A in (1.4.10) has full column rank, we use Gaussian elimination. If $\operatorname{rank}(A) < n$ we check whether c is a linear combination of the rows of A by solving the equation system $A^T\lambda = c$.

If this system has no solution then (1.4.10) has no optimal solution. Otherwise we can delete some columns of A and the corresponding components of c to obtain an equivalent linear program with a constraint matrix of full column rank. So this preprocessing can be done with methods based on Gaussian elimination.

If we do not have an initial basic feasible index set, we proceed as follows. We add one new variable λ and consider the following linear program

(1.4.15)
$$\begin{aligned} &\max \lambda \\ Ax - b\lambda &\leq 0 \\ -\lambda &\leq 0 \\ \lambda &\leq 1 \end{aligned}$$

An initial basic feasible index set for this program is easily found: take an arbitrary basis $A_K.$ of A and set $I := K \cup \{m+1\}$. The basic feasible solution corresponding to I is the zero vector. Next solve (1.4.15) with the simplex method. If the optimum value of (1.4.15) is 0, (1.4.10) has no solution. Otherwise the optimum value is 1. Let, in this case, I be the final basic feasible index set, and let $(v^T, 1)^T$ be the corresponding optimum solution. Set $K := I \setminus \{m+2\}$; then $A_K.$ is a basis of A. Clearly, $A_K.v = b_K$ and $Av \leq b$; hence K is a basic feasible index set that can be used as the initial basic feasible index set for solving (1.4.10) with the simplex method.

Computation of the Hermite Normal Form

We shall now describe some results and a pivoting algorithm from elementary number theory that are useful in integer linear programming.

A rational $m \times n$-matrix of rank m (i. e., of full row rank) is said to be in **Hermite normal form** if it has the form $(B\ 0)$ where B is a nonsingular, lower triangular, nonnegative matrix in which each row has a unique maximum entry, and this entry is on the main diagonal of B. Usually, the Hermite normal form is only defined for integral matrices, but for our purposes it is more appropriate to work with this more general definition.

We shall now discuss whether and, if so, how matrices of full rank can be brought into Hermite normal form and what properties this normal form has. Let us call the following operations on a matrix **elementary column operations** :

(1.4.16)
(i) exchanging two columns ,
(ii) multiplying a column by -1 ,
(iii) adding an integral multiple of one column to another column .

The next result is due to HERMITE (1851).

(1.4.17) Theorem. (a) *Every rational matrix A of full row rank, can be brought into Hermite normal form by a series of elementary column operations* (1.4.16). (b) *This Hermite normal form of A is uniquely determined by A.* □

We shall describe an algorithm that proves (a) of Theorem (1.4.17). So suppose that A is a rational $m \times n$-matrix of full row rank. We construct a sequence $A_0, A_1, \ldots, A_m$ of $m \times n$-matrices such that A_m is lower triangular. We start with $A_0 := A$. In the general step we assume that the matrix A_k, $k \geq 0$, has been constructed from A by elementary column operations such that

$$A_k = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix},$$

where B is lower triangular of order $k \times k$ (for $k = 0$, this means B is empty). Now apply the following procedure.

(1.4.18) Triangularization Step. *Use elementary column operations to transform D in such a way that the first row $D_1. = (d_{11}, \ldots, d_{1,n-k})$ of D has the following properties:*
(a) $d_{11} > d_{12} \geq \ldots \geq d_{1,n-k} \geq 0$ *and*
(b) $d_{11} + d_{12} + \ldots + d_{1,n-k}$ *is as small as possible.* □

It should be clear that the result required in (1.4.18) can be obtained by the operations (1.4.16) and that the number of times elementary column operations must be applied to achieve this is bounded by a polynomial in $n - k$. So each step (1.4.18) can be executed using a number of elementary arithmetic operations bounded by a polynomial in n.

Now note that, since A has full row rank, d_{11} must be positive. Moreover, $d_{12} = \ldots = d_{1,n-k} = 0$, for if $d_{12} > 0$, then we can subtract the second column of D from the first column obtaining a new matrix D for which the sum of the entries of the first row is smaller, contradicting (b). So the output of the Triangularization Step (1.4.18) is a matrix with a larger lower triangular matrix B in the upper left hand corner than A_k. Call this new matrix A_{k+1} and repeat.

After m applications of (1.4.18) we end up with a matrix $A_m = (B\ 0)$ with $B = (b_{ij})$ a lower triangular $m \times m$-matrix with positive entries on the main diagonal. Now we apply the following procedure to A_m, resp. B:

(1.4.19) Row Reduction.
For $i = 2, \ldots, m$ do the following:
For $j = 1, \ldots, i-1$ do the following:
Add an integer multiple of the i-th column of B to the j-th column of B such that the (i,j)-th entry b_{ij} of B is nonnegative and smaller than b_{ii}. □

It is straightforward to see that after termination of (1.4.19) the resulting matrix is in Hermite normal form. Moreover, it is also clear that the number of elementary arithmetic operations necessary to execute (1.4.19) is bounded by a polynomial in n.

So altogether we have shown the correctness of part (a) of Theorem (1.4.17) by designing an algorithm whose total number of elementary arithmetic operations is bounded by a polynomial in n. Moreover, as in the proof of Theorem

(1.4.8) (showing that the entries d_{ij} of D are small) we can conclude that all entries on the main diagonal of the Hermite normal form of A are quotients of subdeterminants of A, and thus the encoding length of each of these entries is bounded by a polynomial in $\langle A \rangle$. From this and the fact that the largest element of the Hermite normal form is on the main diagonal we can conclude :

(1.4.20) Proposition. *The encoding length of the Hermite normal form* $(B\ 0)$ *of a rational matrix* A *of full row rank is bounded by a polynomial in* $\langle A \rangle$. □

Can we now conclude that the algorithm described above for the construction of the Hermite normal form has polynomial running time? We cannot! The reason is that although we have a polynomial bound on the size of the final output of the algorithm through (1.4.20) and a polynomial bound on the number of elementary arithmetic operations we cannot guarantee that the sizes of the intermediate numbers do not grow too large. In fact, by performing this algorithm in practice on rather small, say integral 10×10-matrices, one can observe frequently that the intermediate numbers calculated during the execution of the algorithm will grow enormously large. Quite a number of schemes for the order of performing the elementary column operations have been suggested. The first polynomial time algorithm for finding the Hermite normal form was found by FRUMKIN (1976c)) – see also KANNAN and BACHEM (1979) and DOMICH, KANNAN and TROTTER (1987) for more efficient methods.

We will not elaborate on these methods since our results on basis reduction in lattices, to be described in Chapter 5, will yield a polynomial time algorithm for the computation of the Hermite normal form – see (5.4.13). However, we would like to mention a consequence proved by FRUMKIN (1976a,b) and VON ZUR GATHEN and SIEVEKING (1976).

(1.4.21) Theorem. *For any matrix* $A \in \mathbb{Q}^{m \times n}$ *and any vector* $b \in \mathbb{Q}^m$, *one can decide whether* $Ax = b$ *has an integral solution and, if so, find one in time polynomial in* $\langle A \rangle + \langle b \rangle$. □

This shows that the linear diophantine equations problem (0.1.31) can be solved in polynomial time for rational data.

One may wonder whether the Hermite normal form can be computed in strongly polynomial time. However, the problem of finding the greatest common divisor of two integers is a special case, and one can show that, within our model of strongly polynomial time computation, the greatest common divisor of two integers cannot be computed in strongly polynomial time, that is, in a bounded number of elementary arithmetic operations, where the bound is independent of the two integers.

Chapter 2

Algorithmic Aspects of Convex Sets: Formulation of the Problems

Convex sets and convex functions are typical objects of study in mathematical programming, convex analysis, and related areas. Here are some key questions one encounters frequently:

- *Given a point y and a set K, is y a member of K, i. e., is y contained in K?*
- *If y is not a member of K, find a hyperplane separating y from K.*
- *Given a linear inequality, is it valid for each vector in K?*
- *Given a linear function, find a point maximizing (or minimizing) the function on K.*
- *Given a convex function, find its minimum.*

Membership, separation, validity, and optimization problems are fundamental problems in these theories. Optimization problems have been intensively studied, in particular from an algorithmic point of view, because of their relevance for practical applications. Research on membership, separation, and validity problems concentrated much less on algorithmic aspects; existence theorems, characterizations by necessary and/or sufficient conditions and the like were the main lines of investigation. The proof techniques here are often very elegant but nonconstructive.

The basic difference between optimization and separation, membership, and validity is typically that optimization results (theorems or algorithms) are the ultimate goals (from an applied point of view) while separation results etc. are used as important tools for the derivation of such results. Clearly, optimization algorithms are what mathematical programming is looking for, but even so, it is surprising how little attempt has been made to devise efficient separation, membership, or validity algorithms.

We only know of a few examples of good separation algorithms that have been designed (but not published) before the ellipsoid method occured. These are the separation algorithm for the subtour elimination constraints of the traveling salesman problem – see CROWDER and PADBERG (1980) – and the separation algorithm for capacitated (perfect) b-matchings described in PADBERG and RAO (1982). These polynomial time methods came up in attempts to design practically fast cutting plane algorithms for the traveling salesman problem. On the other hand, let us mention that, while optimization over matroid polytopes has been a very extensively studied problem since the pioneering work of EDMONDS (1970, 1971), the first combinatorial algorithm to solve the separation problem for these polytopes was only recently found by CUNNINGHAM (1984) and was prompted by more general results derived from the ellipsoid method.

In Chapter 3 we shall show that there is a strong relation between the polynomial time solvability of the optimization problem and that of the separation problem. In fact, they are equivalent if one is a little careful in defining the concepts to be considered. The precise definition of the problems we want to treat is the purpose of this chapter.

2.1 Basic Algorithmic Problems for Convex Sets

In this section we will introduce a number of notions and problems which will be used throughout the book. The problems we will define are the basic objects of our investigations. One of the main goals of this book is the determination of the algorithmic relations between these problems.

Let K be a convex and compact set in $\mathbb{R}^n$. Then we can formulate the following five algorithmic problems (2.1.1),..., (2.1.5) in connection with K.

(2.1.1) The Strong Optimization Problem (SOPT).

Given a vector $c \in \mathbb{R}^n$, find a vector $y \in K$ that maximizes $c^T x$ on K, or assert that K is empty.

A closely related but slightly weaker question is the following.

(2.1.2) The Strong Violation Problem (SVIOL).

Given a vector $c \in \mathbb{R}^n$ and a number $\gamma \in \mathbb{R}$, decide whether $c^T x \leq \gamma$ holds for all $x \in K$, and if not, find a vector $y \in K$ with $c^T y > \gamma$.

Note that taking $c = 0$ and $\gamma = -1$, the strong violation problem reduces to the problem of checking whether K is empty, and if not, finding a point in K. This problem will be called **strong nonemptiness problem (SNEMPT)**.

(2.1.3) The Strong Validity Problem (SVAL).

Given a vector $c \in \mathbb{R}^n$ and a number $\gamma \in \mathbb{R}$, decide whether $c^T x \leq \gamma$ holds for all $x \in K$.

The following two problems are – in a sense to be made precise later – polar to the ones above.

(2.1.4) The Strong Separation Problem (SSEP).

Given a vector $y \in \mathbb{R}^n$, decide whether $y \in K$, and if not, find a hyperplane that separates y from K; more exactly, find a vector $c \in \mathbb{R}^n$ such that $c^T y > \max\{ c^T x \mid x \in K \}$.

(2.1.5) The Strong Membership Problem (SMEM).

Given a vector $y \in \mathbb{R}^n$, decide whether $y \in K$.

Clearly, if we can solve (2.1.1) we can also solve (2.1.2), and if we can solve (2.1.2) we can solve (2.1.3). Similarly, (2.1.5) can be solved if (2.1.4) is solvable. So there are some obvious algorithmic relations between these five problems. In Figure 2.1 we have graphically displayed these trivial relations between SOPT, SVIOL, SVAL, SSEP, and SMEM (and SNEMPT in addition). An arrow here means that the problem at the head of the arrow can be solved in oracle-polynomial time given an oracle for the problem at the tail of the arrow. (The relation is transitive, but arrows following from transitivity are not shown.)

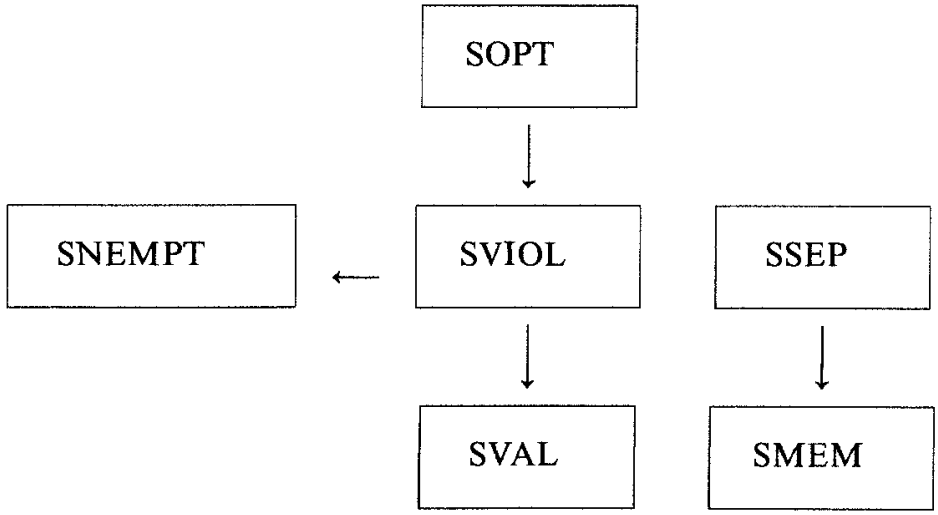


Figure 2.1

These problem formulations immediately give rise to a number of questions. For instance: Is our requirement to provide "exact" solutions of the problems (2.1.1),..., (2.1.5) too restrictive? Are the assumptions that K should be convex

and compact too general or too strong? Are these problems algorithmically solvable at all? Moreover, are there other implications between SOPT,..., SMEM different from those depicted in Figure 2.1? As mentioned before, this last question is among the central topics of this book. But before discussing all these matters let us look at some typical examples.

(2.1.6) Example. Let $K \subseteq \mathbb{R}^n$ be a polyhedron defined by a given system of linear inequalities $a_i^T x \leq b_i$, $i = 1, \ldots, m$. For any vector $y \in \mathbb{R}^n$, the separation problem (2.1.4) and the membership problem (2.1.5) are trivial to solve. We simply substitute y for x in the inequalities. If y satisfies them all, then y is a member of K, otherwise y is not; and any violated inequality yields a separating hyperplane. The strong optimization problem (2.1.1) for K is nothing but a linear program. Linear programs can be solved by several algorithms but not as easily as the separation problem above. □

(2.1.7) Example. Let $K \subseteq \mathbb{R}^n$ be a polytope defined as the convex hull of a given finite set of vectors, i. e., $K = \text{conv}(\{v_1, \ldots, v_k\})$. Here the strong optimization problem (2.1.1) (and similarly problems (2.1.2), (2.1.3)) for K can be solved easily by evaluating the function $c^T x$ at all points v_i and selecting the one with the largest value $c^T v_i$. To solve the strong membership problem (2.1.5) for a vector $y \in \mathbb{R}^n$, one has to check whether y is a convex combination of the v_i or not. So this problem requires deciding whether a system of linear equations and inequalities has a feasible solution. To solve the strong separation problem (2.1.4) requires, in addition, finding a vector $c \in \mathbb{R}^n$ with $c^T y > c^T v_i$ $(i = 1, \ldots, k)$ if the system has no solution. □

(2.1.8) Example. Let $f: \mathbb{R}^n \to \mathbb{R}$ be a convex function and $G_f := \{(x^T, t)^T \in \mathbb{R}^{n+1} \mid f(x) \leq t\}$ be the "epigraph" of f. If $f(x)$ can be computed, then the membership problem (2.1.5) for G_f is trivial to solve; if in addition a subgradient of f at x can be computed, then the separation problem (2.1.4) for G_f can be solved easily. Namely, suppose $(y^T, s)^T \notin G_f$. Then, by definition, $\pi \in \mathbb{R}^n$ is a subgradient of f at y if $f(x) \geq f(y) + \pi^T(x - y)$ for all $x \in \mathbb{R}^n$. So $c := (\pi^T, -1)^T \in \mathbb{R}^{n+1}$ satisfies $(y^T, s)c = \pi^T y - s > \pi^T y - f(y) \geq \pi^T x - f(x) \geq (x^T, t)c$ for all $(x^T, t)^T \in G_f$. So c yields a separating hyperplane. □

(2.1.9) Example. In classical geometry, convex sets K are often described in terms of the support function – see BONNESEN and FENCHEL (1934) for instance. For every vector $v \in \mathbb{R}^n$ with $\|v\| = 1$, consider the supporting hyperplane H of K with outer normal v, and let $\rho(v)$ be the signed distance of H from zero; i. e., $\rho(v) := \max\{v^T x \mid x \in K\}$, $H = \{x \mid v^T x = \rho(v)\}$, $H \cap K \neq \emptyset$, $\{x \mid v^T x > \rho(v)\} \cap K = \emptyset$. If the function ρ is given, then the validity problem (2.1.3) is solvable trivially, since $c^T x \leq c_0$ is valid if and only if $\rho(c/\|c\|) \leq c_0/\|c\|$ (if $c \neq 0$; the case $c = 0$ is solved trivially, as $K \neq \emptyset$). □

The examples above show that the way a set K is given makes an essential difference. The strong separation problem is trivial for a polytope given as in

(2.1.6), whereas it turns out to be nontrivial for a polytope given as in (2.1.7). For the strong optimization problem it is just the other way around!

It is well known that a polytope K given in the form (2.1.6) also has a representation in the form (2.1.7) (and vice versa). Moreover, such a representation can be found in finite time. This fact is of little use in designing polynomial time algorithms, since, to describe the convex hull K of k points, one may need a number of inequalities that is exponential in k. (Consider, for instance, the convex hull of the $2n$ vectors $(0,\dots,0,\pm 1,0,\dots,0)^T$.) Similarly, the solution set of m inequalities may have a number of vertices that is exponential in m. (Consider the n-cube.) So the size of the description may grow exponentially if one goes from one representation of a polytope to another. This illustrates that the polynomial time solvability aspects of the strong optimization (separation etc.) problem depend on the way K is given and not only on K.

Our approach is to describe a set K by certain parameters and algorithms (oracles) that define the set K uniquely. So K is not necessarily given by any of the usual representations. For instance, K can be given by the dimension of the space together with a "membership algorithm" that, for any given point, decides whether it is in K or not. This will allow us to take also nonpolyhedral convex sets into consideration.

But clearly, we have to pay for such a generality. Namely, if we allow arbitrary convex sets, it may well be that K contains only one vector maximizing $c^T x$ on K and that this vector has irrational coordinates. To make the formulation of our problems correct in such cases we recall the discussion of computability of numbers in Section 1.3. In the spirit of this, we introduce the following five problems which come from their strong versions by allowing margins in all inequalities and around the surface of the given set K. It may help to understand the following if we recall that $x \in S(K,\varepsilon)$ means "x is almost in K", and that $x \in S(K,-\varepsilon)$ means "x is deep in K". In the following definitions K denotes a compact and convex set in $\mathbb{R}^n$.

For easy reference the five basic problems are also listed on the inner side of the back cover.

(2.1.10) The Weak Optimization Problem (WOPT).

Given a vector $c \in \mathbb{Q}^n$ and a rational number $\varepsilon > 0$, either

(i) *find a vector $y \in \mathbb{Q}^n$ such that $y \in S(K,\varepsilon)$ and $c^T x \leq c^T y + \varepsilon$ for all $x \in S(K,-\varepsilon)$*
(i. e., y is almost in K and almost maximizes $c^T x$ over the points deep in K), *or*

(ii) *assert that $S(K,-\varepsilon)$ is empty.*

(2.1.11) The Weak Violation Problem (WVIOL).

Given a vector $c \in \mathbb{Q}^n$, a rational number γ, and a rational number $\varepsilon > 0$, either

(i) *assert that $c^T x \leq \gamma + \varepsilon$ for all $x \in S(K, -\varepsilon)$* (i. e., $c^T x \leq \gamma$ is almost valid), *or*
(ii) *find a vector $y \in S(K, \varepsilon)$ with $c^T y \geq \gamma - \varepsilon$* (a vector almost violating $c^T x \leq \gamma$).

(2.1.12) The Weak Validity Problem (WVAL).

Given a vector $c \in \mathbb{Q}^n$, a rational number γ, and a rational number $\varepsilon > 0$, either

(i) *assert that $c^T x \leq \gamma + \varepsilon$ for all $x \in S(K, -\varepsilon)$, or*
(ii) *assert that $c^T x \geq \gamma - \varepsilon$ for some $x \in S(K, \varepsilon)$* (i. e., $c^T x \leq \gamma$ is almost nonvalid).

(2.1.13) The Weak Separation Problem (WSEP).

Given a vector $y \in \mathbb{Q}^n$ and a rational number $\delta > 0$, either

(i) *assert that $y \in S(K, \delta)$, or*
(ii) *find a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ such that $c^T x \leq c^T y + \delta$ for every $x \in S(K, -\delta)$* (i. e., find an almost separating hyperplane).

(2.1.14) The Weak Membership Problem (WMEM).

Given a vector $y \in \mathbb{Q}^n$ and a rational number $\delta > 0$, either

(i) *assert that $y \in S(K, \delta)$, or*
(ii) *assert that $y \notin S(K, -\delta)$.*

Similarly as in the strong version SVIOL (2.1.2), the special case of (2.1.11) where $c = 0$ and $\gamma = -1$ is of particular importance. Note, however, that (for $\varepsilon < 1$) output (i) of WVIOL only means that $S(K, -\varepsilon)$ is empty (K might still be nonempty) and output (ii) means finding a point almost in K. We call this special case of WVIOL the **weak nonemptiness problem (WNEMPT).**

(2.1.15) Remark. (a) All the strict inequalities appearing in the strong versions (2.1.1), ..., (2.1.5) have been turned into weak inequalities in problems (2.1.10), ..., (2.1.14). Clearly, this makes no difference here, since little errors are allowed everywhere. Strict inequalities, if needed, may be obtained by decreasing the ε's and δ's. Moreover, the reader may have noticed that in some of the weak problems above several ε's and δ's are redundant. For example, solving the weak validity problem (2.1.12) with input c, $\gamma' := \gamma + \varepsilon/2$ and $\varepsilon' := \varepsilon/2$ gives a solution to a stronger version of (2.1.12) where we require that $c^T x \geq \gamma$ in (ii). The use of our weaker versions has the advantage of uniformity and will also make the application of polarity easier.

(b) There is one additional, purely technical change in the separation problem. By introducing the δ-error in (ii) of (2.1.13) we have to normalize the vector c, because otherwise every small enough vector c would satisfy (ii). We could not require the Euclidean norm to be 1 since this might lead to scaling by an irrational factor.

(c) Note also that the weak problems above have been modelled so that the output is "continuous" in the input. As an illustration consider the weak optimization problem (2.1.10). Given a vector $c \in \mathbb{Q}^n$, a rational number $\varepsilon > 0$ and in addition a vector $y \in \mathbb{Q}^n$, we can compute a rational number $\delta > 0$ in polynomial time such that if $c', y' \in \mathbb{Q}^n$, $\|c' - c\| < \delta$, $\|y' - y\| < \delta$ and if y' is a valid output for the weak optimization problem with input c' and δ then y is a valid output for the weak optimization problem with input c and ε. We shall not elaborate on this point further, but we feel that this property is essential for the correct posing of the problems. □

Just as in the strong versions of our problems, there are immediate implications between our weak versions. Most of these directly follow from the definitions. For instance, if WVIOL can be solved for K in polynomial time then, obviously, WVAL is polynomially solvable. A little less direct, but still trivial is the implication from WSEP to WMEM. If we have an algorithm for the weak separation problem for K and we want to solve the weak membership problem for K, we proceed as follows. Given $y \in \mathbb{Q}^n$ and $\delta \in \mathbb{Q}$, $\delta > 0$, we solve WSEP for the input y, δ. If our algorithm concludes that $y \in S(K, \delta)$ we can assert (i) of (2.1.14). If the weak separation algorithm finds an almost separating hyperplane we call it again, but now with input y and $\delta/3$. If the algorithm asserts that $y \in S(K, \delta/3)$ then we know that $y \in S(K, \delta)$. If the weak separation algorithm finds a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ and $c^T x \leq c^T y + \delta/3$ for all $x \in S(K, -\delta/3)$, we can conclude that $y \notin S(K, -\delta)$. Namely, suppose this is not so. Then consider the vector $y' := y + (\delta/(2\|c\|))c$, which has distance $\delta/2$ from y. Since $y \in S(K, -\delta)$ we see that $y' \in S(K, -\delta/3)$. But on the other hand we have

$$c^T y' = c^T y + \frac{\delta}{2}\|c\| \geq c^T y + \frac{\delta}{2} > c^T y + \frac{\delta}{3},$$

which contradicts the assertion of the weak separation algorithm.

The trivial relations between the weak problems introduced above are displayed in Figure 2.2 (the arrows have the same interpretations as those of Figure 2.1 – arrows following from transitivity are not drawn.)

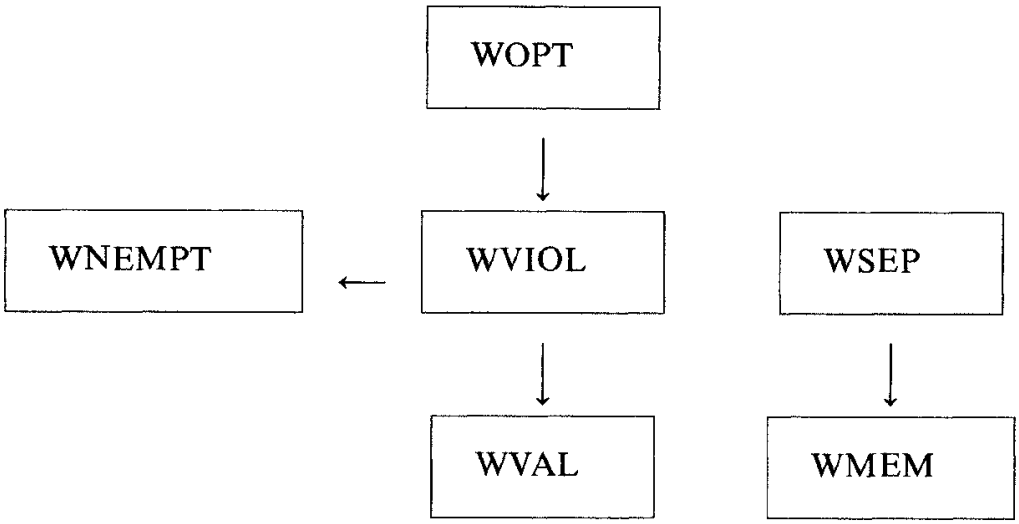


Figure 2.2

We will show in Chapter 4 that there are several more arrows in Figure 2.2, in some cases by making further assumptions on the convex set K under consideration. We shall discuss later to what extent these requirements are necessary.

(2.1.16) Definition. *A convex set K is called* **circumscribed** *if the following information about K is given explicitly:*

(a) *the integer $n = n(K)$ sucht that $K \subseteq \mathbb{R}^n$,*

(b) *a positive rational number $R = R(K)$ such that $K \subseteq S(0, R)$.*

So a circumscribed convex set is a triple $(K; n, R)$.

A circumscribed convex set $(K; n, R)$ is called **well-bounded** *if, moreover, the following is given:*

(c) *a positive rational number $r = r(K)$ for which K contains a ball of radius r.* (The center of the ball need not be known explicitly.)

Thus, a well-bounded convex set is a quadruple $(K; n, R, r)$.

*A well-bounded convex set $(K; n, R, r)$ is called a_0-***centered** *(or just* **centered** *if, moreover, the following is given:*

(d) *a vector $a_0 = a_0(K) \in \mathbb{Q}^n$ such that $S(a_0, r) \subseteq K$.*

Thus, an a_0-centered convex set is a quintuple $(K; n, R, r, a_0)$ with the above properties. □

A centered convex set is always well-bounded, and a well-bounded convex set is always circumscribed. Most of the convex sets we will encounter in our applications are compact and full-dimensional. Convex sets with these two properties are called **convex bodies**. We will speak in such cases of **circumscribed**, or **well-bounded**, or **centered convex bodies**. If K is a convex set containing some ball of radius r we call r an **inner radius** of K. If $K \subseteq S(0, R)$ then R is called an **outer radius** of K.

Before continuing, we would like to mention that assumption (c) of (2.1.16) provides a lower bound on the volume of K. In fact, it follows from the proofs of the results shown later that we could have required equivalently that a lower bound on the volume of K is given. Explicit knowledge of $r(K)$ is not necessary, but more convenient.

In formulating some of our results we shall have to restrict ourselves to convex sets K for which some mathematical description is given, called the **name** of K and denoted by **Name**(K). The same convex set may of course be given by different definitions and therefore may have different names.

(2.1.17) Definition. *A convex set (or body) K together with its encoded name* Name(K) *will be called a* **named convex set** *(or* **body***). The length of the encoding of the name of K will be denoted by* $\langle \text{Name}(K) \rangle$. □

To give an example, consider the polytope P defined in the following two ways.

(2.1.18) The set of vectors $(x_1, x_2)^T \in \mathbb{R}^2$ satisfying $x_1 \geq 0$, $x_2 \geq 0$, $x_1 + x_2 \leq 1$.

(2.1.19) The convex hull of the vectors $\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}$.

Either one of these two definitions could be used to name P. The encoding length of the name of P would be either the length of the encoding of the inequality system in (2.1.18) or the length of the encoding of the three vectors in (2.1.19).

More interesting examples of names will come up in combinatorial applications. For instance, if we study the matching polytope P of a graph G, P may be named by the graph G, and moreover, the name may be used to compute the parameters n, R, r, or a_0.

We still have to specify how we measure the encoding length of a convex set K.

(2.1.20) Definition. *Let K be a convex set. Then we denote the* **encoding length** *of K by $\langle K \rangle$. For the various kinds of convex sets defined above, $\langle K \rangle$ shall have the following meaning* (recall the definition of encoding length for vectors in Section 1.3):

(a) *If K is circumscribed, then* $\langle K \rangle := n(K) + \langle R(K) \rangle$.

(b) *If K is well-bounded, then* $\langle K \rangle := n(K) + \langle R(K) \rangle + \langle r(K) \rangle$.

(c) *If K is centered, then* $\langle K \rangle := n(K) + \langle R(K) \rangle + \langle r(K) \rangle + \langle a_0(K) \rangle$.

(d) *If K is named and circumscribed, then* $\langle K \rangle := n(K) + \langle R(K) \rangle + \langle \text{Name}(K) \rangle$.

(e) *If K is named and well-bounded, then* $\langle K \rangle := n(K) + \langle R(K) \rangle + \langle r(K) \rangle + \langle \text{Name}(K) \rangle$.

(f) *If K is named and centered, then* $\langle K \rangle := n(K) + \langle R(K) \rangle + \langle r(K) \rangle + \langle a_0(K) \rangle + \langle \text{Name}(K) \rangle$. □

In most cases treated in the following, the convex sets considered are not given by a name but by an oracle (or in practice: by a computer subroutine). An oracle for instance may tell us whether a given point is (almost) contained in K or not. Such an oracle would be called a **(weak) membership** oracle. Similarly, we

will consider **separation, optimization, violation,** or **validity oracles** in the weak as well as in strong sense. If oracles are used then of course we have to consider the running times of the oracle algorithms for our complexity calculations. This will make it necessary to utilize the concept of oracle Turing machines introduced in Section 1.2.

In the following we shall speak of a **polynomial time** or **oracle-polynomial time algorithm for a certain problem defined on a class of convex sets** if the running time of the algorithm is bounded by a polynomial in $\langle K \rangle$ and in the encoding length of the possibly existing further input (objective functions, error parameters etc.), for every convex set K in the given class.

We shall now briefly discuss the reasons for restricting our attention to convex sets as defined in (2.1.16).

First, convexity seems basic for our method. The requirement (a) simply states that we have to know the number of variables involved.

The outer radius R tells us about in which portion of $\mathbb{R}^n$ our set K is located. It gives us a rough estimate of the space we have to search for K. Requiring (b) rules out the treatment of unbounded sets; it can be viewed as a strong explicit boundedness condition. (We will show later that – under some extra conditions – unbounded convex sets can also be handled by our methods. This, however, involves several technicalities.) Similarly, condition (c) is an explicit full-dimensionality condition. The roles of (b) and (c), in particular the necessity of knowing r and R, will be discussed in detail in Chapter 4.

In some applications, requirements (a), (b), and (c) of (2.1.16) for a convex set are sufficient to calculate an interior point in polynomial time. This, however, is not always the case, and we may need the explicit a priori knowledge of an interior point $a_0(K)$ with the properties described in (2.1.16).

We shall also show that by strengthening the assumption "K is convex" to "K is a polytope" or "K is given by an inequality system" the requirements (a), (b), and (c) of (2.1.16) can either be removed or be replaced by much simpler and more natural ones. Such investigations will be the subject matter of Chapter 6.

In most cases treated in the sequel we will restrict ourselves to well-bounded convex bodies $(K; n, R, r)$, since they are the relevant objects for our applications. A short look at our definitions shows that, for the weak versions (2.1.10),..., (2.1.14) of our five problems, closedness is a completely irrelevant assumption. Since we allow errors, it does not make any (essential) difference whether we optimize (or separate etc.) over a full-dimensional set K, or its closure, or its interior. In some of the strong versions closedness is however needed, otherwise no solutions of the problems may exist and the problems could not even theoretically be solved. For reasons of uniformity, we have therefore decided to state and prove our results for closed convex sets only.

To close this section, let us formulate one more problem (in its strong and weak version), which is fundamental in mathematical programming.

(2.1.21) The Strong Unconstrained Convex Function Minimization Problem. *Given a convex function $f: \mathbb{R}^n \to \mathbb{R}$, find a vector $y \in \mathbb{R}^n$ such that $f(y) \leq f(x)$ for all $x \in \mathbb{R}^n$.*

The algorithmic solution of (2.1.21) clearly depends on the way the function f is given to us. It may be given by an oracle that gives us the value of $f(x)$ for any given $x \in \mathbb{R}^n$. In addition, we have to know a radius R such that the minimum in (2.1.21), if it exists, is contained in the ball $S(0,R)$. Otherwise no algorithm can be guaranteed to find the minimum.

(2.1.22) The Weak Constrained Convex Function Minimization Problem. *Given a compact convex set $K \subseteq \mathbb{R}^n$, a convex function $f: \mathbb{R}^n \to \mathbb{R}$, and a rational number $\varepsilon > 0$, find a vector $y \in \mathbb{Q}^n \cap S(K,\varepsilon)$ such that $f(y) \leq f(x) + \varepsilon$ for all $x \in S(K,-\varepsilon)$.*

Problem (2.1.22) is a special case of the weak optimization problem (2.1.10), which can be seen as follows. Following example (2.1.8), we set

$$G(f,K) := \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \in \mathbb{R}^{n+1} \mid x \in K,\ f(x) \leq t \leq B \right\},$$

where B is any upper bound on the minimum value of f on K. It is obvious that $G(f,K)$ is a compact convex set in $\mathbb{R}^{n+1}$, and that (2.1.22) can be solved by maximizing the objective function $-x_{n+1}$.

If subgradients of f – cf. (2.1.8) – can be computed, then (2.1.22) can also be reduced to (2.1.10) without adding an additional variable by using "subgradient cuts". This leads back to one of the sources of the subject of our book. Namely, the ellipsoid method arose from the work of SHOR (1977) on minimizing nondifferentiable convex functions with subgradient techniques.

*2.2 Nondeterministic Decision Problems for Convex Sets

Before discussing polynomial time algorithms for the problems introduced above, we deal with their membership in $\mathcal{NP}$ and co-$\mathcal{NP}$. The results of this section are rather technical and will not be needed in what follows; but we believe that an investigation of these problems helps to understand the subtleties of the subject that come up when complexity theory is applied to geometric questions.

The problems that are usually discussed in the $\mathcal{P}$-$\mathcal{NP}$-framework have the property that exactly one of two possible decisions is a legal answer (a graph is either Hamiltonian or not, but never both). In our weak problems (2.1.10),..., (2.1.14) there may be instances for which either of the two possible conclusions is a correct answer. Our definitions of these problems being in $\mathcal{NP}$ or in co-$\mathcal{NP}$ have to take those "fuzzy" instances into account, and we will therefore require a positive answer to have a polynomial length proof only if the other alternative does not hold. Thus our definitions are the following.

Let $\mathcal{K}$ be a class of named well-bounded convex bodies. The **weak membership problem for $\mathcal{K}$ is in $\mathcal{NP}$** means that for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\delta > 0$, and any $y \in \mathbb{Q}^n$ such that $y \in S(K,-\delta)$ (i. e., y is deep in K), there exists a certificate of the fact that $y \in S(K,\delta)$ (i. e., y is almost in K), not longer than a polynomial in the encoding length, that is, in $n+\langle R\rangle+\langle r\rangle+\langle \text{Name}(K)\rangle+\langle y\rangle+\langle \delta\rangle$.

The **weak membership problem for $\mathcal{K}$ is in co-$\mathcal{NP}$** means that for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\delta > 0$ and any $y \in \mathbb{Q}^n$ such that $y \notin S(K, \delta)$, there exists a certificate of the fact that $y \notin S(K, -\delta)$ not longer than a polynomial in the encoding length.

Similarly, we say that the **weak validity problem for $\mathcal{K}$ is in $\mathcal{NP}$** if, for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\varepsilon > 0$, any $c \in \mathbb{Q}^n$, and any $\gamma \in \mathbb{Q}$ such that $c^T x < \gamma - \varepsilon$ for all $x \in S(K, \varepsilon)$, there exists a certificate of the fact that $c^T x \leq \gamma + \varepsilon$ for all $x \in S(K, -\varepsilon)$, not longer than a polynomial in the encoding length.

We say that the **weak validity problem for $\mathcal{K}$ is in co-$\mathcal{NP}$**, if for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\varepsilon > 0$, any $c \in \mathbb{Q}^n$ and $\gamma \in \mathbb{Q}$ such that $c^T x > \gamma + \varepsilon$ for some $x \in S(K, -\varepsilon)$, there exists a certificate of the fact that $c^T x \geq \gamma - \varepsilon$ for some $x \in S(K, \varepsilon)$ not longer than a polynomial in the encoding length.

The violation and separation problems are not decision problems, and therefore they do not fit in the $\mathcal{NP}$-framework. Motivated by the definitions above, however, we can formulate two further complexity properties of the class $\mathcal{K}$. If the membership problem is in co-$\mathcal{NP}$, it is often the case that the infeasibility of a point is proved by exhibiting a separating hyperplane. This motivates the following definitions.

We say that the **weak separation problem for $\mathcal{K}$ is in $\mathcal{NP}$** if, for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\delta > 0$, and any $y \in \mathbb{Q}^n$ such that $y \notin S(K, \delta)$, there exists a $c \in \mathbb{Q}^n$, $\|c\|_\infty = 1$ for which the fact that $c^T x \leq c^T y + \delta$ for all $x \in S(K, -\delta)$ has a certificate whose encoding length is polynomial in $n + \langle R \rangle + \langle r \rangle + \langle \text{Name}(K) \rangle + \langle y \rangle + \langle \delta \rangle$. This in particular includes that $\langle c \rangle$ is also bounded by such a polynomial.

Finally, we say that the **weak violation problem for $\mathcal{K}$ is in $\mathcal{NP}$** if, for any $(K; n, R, r) \in \mathcal{K}$, any rational number $\varepsilon > 0$, any $c \in \mathbb{Q}^n$ and $\gamma \in \mathbb{Q}$ such that $c^T x > \gamma + \varepsilon$ for some $x \in S(K, -\varepsilon)$, there exists a $y \in S(K, \varepsilon)$ for which $c^T y \geq \gamma - \varepsilon$ and the fact that $y \in S(K, \varepsilon)$ has a certificate whose length is polynomial in $n + \langle R \rangle + \langle r \rangle + \langle \text{Name}(K) \rangle + \langle c \rangle + \langle \gamma \rangle + \langle \varepsilon \rangle$.

It is natural to define the weak violation problem for $\mathcal{K}$ to be in co-$\mathcal{NP}$ if the weak validity problem for $\mathcal{K}$ is in $\mathcal{NP}$. Similarly, weak separation for $\mathcal{K}$ is in co-$\mathcal{NP}$ if weak membership is in $\mathcal{NP}$. Some implications between these complexity properties of $\mathcal{K}$ are immediate.

(2.2.1) Proposition. *Let $\mathcal{K}$ be a class of named well-bounded convex bodies.*

(a) *If the weak membership problem for $\mathcal{K}$ is in $\mathcal{NP}$, then also the weak violation problem for $\mathcal{K}$ is in $\mathcal{NP}$.*

(b) *If the weak validity problem for $\mathcal{K}$ is in $\mathcal{NP}$, then also the weak separation problem for $\mathcal{K}$ is in $\mathcal{NP}$.*

(c) *If the weak violation problem for $\mathcal{K}$ is in $\mathcal{NP}$, then the weak validity problem for $\mathcal{K}$ is in co-$\mathcal{NP}$.*

(d) *If the weak separation problem for $\mathcal{K}$ is in $\mathcal{NP}$, then the weak membership problem for $\mathcal{K}$ is in co-$\mathcal{NP}$.* □

The geometrical contents of these statements is trivial, but arithmetic details require some care. For example, when proving (b), one should check that there

exists indeed a vector c with the required properties such that $\langle c \rangle$ is bounded by a polynomial in the encoding length.

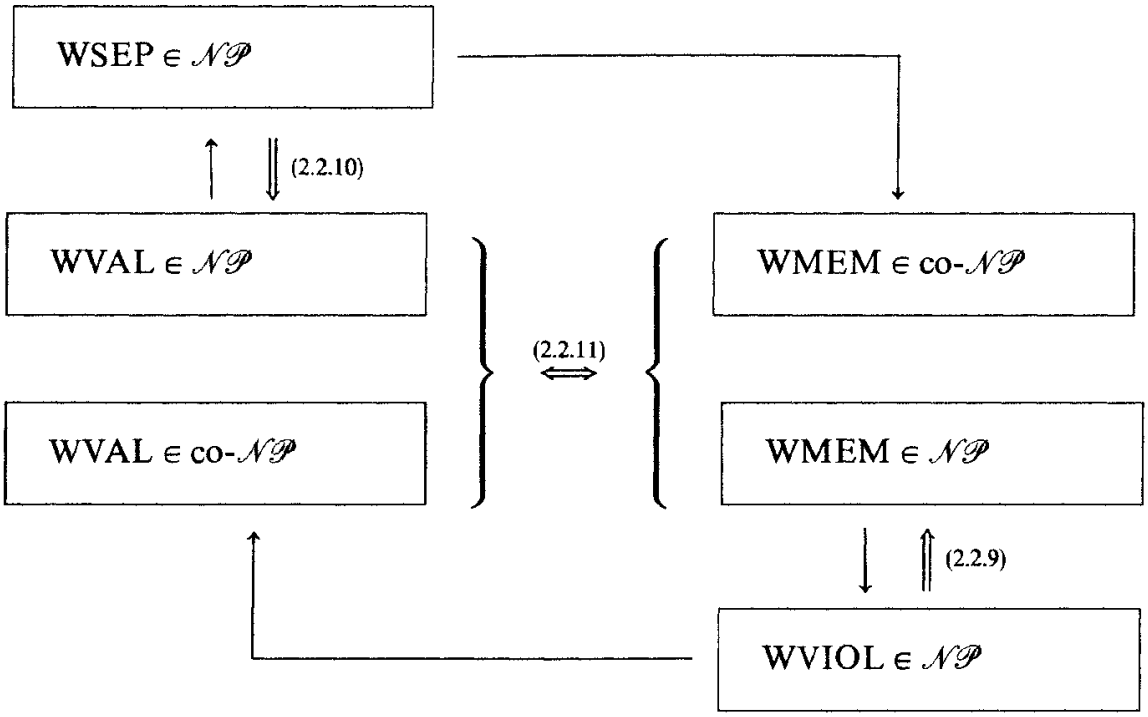


Figure 2.3

The relations between the various complexity properties of $\mathcal{K}$ are depicted in Figure 2.3. An arrow in this diagram means "implication"; e. g., Proposition (2.2.1) (a) is schematically shown by the arrow from the box containing WMEM $\in$ $\mathcal{NP}$ to the box WVIOL $\in$ $\mathcal{NP}$. Arrows drawn with one straight line correspond to the trivial implications of Proposition (2.2.1). The other relations between memberships in $\mathcal{NP}$ and co-$\mathcal{NP}$ respectively are more complicated to prove. The corresponding arrows in Figure 2.3 are drawn with double lines. These relations are proved in the remaining part of this section.

We start with some lemmas. The first of them is an analogue of the well-known Krein-Milman characterization of convex hulls.

Let F be a compact subset of $\mathbb{R}^n$, t and R be positive real numbers with $F \subseteq S(0,R)$ and $t \geq R$. Let the **t-hull of F** (denoted by $\text{hull}_t(F)$) be defined as the intersection of all balls of radius t containing F. Clearly,

$$\text{conv}(F) \subseteq \text{hull}_t(F) ,$$

and in fact,

$$\text{conv}(F) = \bigcap_{t \geq R} \text{hull}_t(F) . \tag{2.2.2}$$

A point $x \in F$ is called **t-extremal** if there exists a ball $S(a,t)$ of radius t around some point $a \in \mathbb{R}^n$ such that $F \subseteq S(a,t)$ and x lies on the boundary of $S(a,t)$. We denote by $\text{ex}_t(F)$ the set of all t-extremal points in F.

(2.2.3) Lemma.

$$\text{hull}_t(F) = \text{hull}_t(\text{ex}_t(F)).$$

Proof. Since $\text{ex}_t(F) \subseteq F$, it is trivial that $\text{hull}_t\big(\text{ex}_t(F)\big) \subseteq \text{hull}_t(F)$.

To prove the reverse inclusion, we show that every ball $S(a,t)$ containing $\text{ex}_t(F)$ also contains F.

Suppose there is a ball $S(a,t) \supseteq \text{ex}_t(F)$ with $S(a,t) \not\supseteq F$. Since $F \subseteq S(0,t)$, as $t \geq R$, the value $\lambda_0 := \max\{\lambda \mid 0 \leq \lambda \leq 1,\ F \subseteq S(\lambda a, t)\}$ exists and is less than 1 by assumption. Clearly, F must contain a point z from the hemisphere

$$\{x \in \mathbb{R}^n \mid \|x - \lambda_0 a\| = t,\ (x - \lambda_0 a)^T a \leq 0\}$$

(i. e., from the hemisphere of $S(\lambda_0 a, t)$ visible from infinity in the direction a). But then $z \in \text{ex}_t(F)$ and $z \notin S(a,t)$, a contradiction. □

(2.2.4) Lemma.

$$\text{hull}_t(F) \subseteq S\big(\text{conv}(\text{ex}_t(F)), \frac{R^2}{t}\big).$$

Proof. Let $H = \text{ex}_t(F)$. By Lemma (2.2.3) it suffices to prove

$$\text{hull}_t(H) \subseteq S\big(\text{conv}(H), \frac{R^2}{t}\big)\ .$$

Let $a \in \text{hull}_t(H)$. If $a \in \text{conv}(H)$, we have nothing to prove. So suppose $a \notin \text{conv}(H)$. Let a_1 be the point of $\text{conv}(H)$ nearest to a and set

$$d := \frac{1}{\|a_1 - a\|}(a_1 - a)\ ,$$

$$M := \{x \in \mathbb{R}^n \mid d^T x \geq d^T a_1\}.$$

The halfspace M contains $\text{conv}(H)$ and thus H by construction, and therefore, H is contained in $M \cap S(0,R)$. We now include this intersection in a ball $S(b,t)$ with as small a piece outside M as possible. By elementary computations we obtain that the center of this ball is

$$b := \big(\sqrt{t^2 - R^2 + (d^T a_1)^2} + d^T a_1\big)d,$$

and the height of the section $S(b,t) \setminus M$ is at most $t - \sqrt{t^2 - R^2} < R^2/t$. Since $a \in \text{hull}_t(H)$, we have $a \in S(b,t)$, and so by the definition of M, $a \in S(b,t) \setminus M$. But then the distance of a from M, and hence from a_1, is less than R^2/t. Thus $a \in S\big(\text{conv}(H), R^2/t\big)$. □

(2.2.5) Lemma. *Let $z \in \mathrm{ex}_t(F)$ and $\varepsilon > 0$. Then there exists a simplex $\Sigma \subseteq \mathbb{R}^n$ such that $S(z,\varepsilon) \subseteq \Sigma$, $\mathrm{diam}(\Sigma) \leq 4n\sqrt{2t\varepsilon}$, and all but one of the facet defining inequalities of Σ are valid for F. Moreover, the facet defining inequalities of Σ have encoding length bounded by a polynomial of $n + \langle\varepsilon\rangle + \langle t\rangle$.*

Proof. We may assume that $\varepsilon < 1/2$. Note that $t \geq R$. By the definition of $\mathrm{ex}_t(F)$, there exists a vector $a \in \mathbb{R}^n$ such that $\|a - z\| = t$ and $F \subseteq S(a,t)$. Then we may assume that $F = S(a,t)$.

Let H_0 be the hyperplane perpendicular to the vector $z - a$ and tangent to the ball $S(z,\varepsilon)$ at point u separating z from a. Then H_0 intersects F in an $(n-1)$-dimensional ball F'. Let $v_1, \ldots, v_n$ be the vertices of any regular simplex inscribed in F' and let $H_1, \ldots, H_n$ be the hyperplanes tangent to F at $v_1, \ldots, v_n$, respectively. It follows by elementary geometry that $H_1, \ldots, H_n$ are also tangent to $S(z,\varepsilon)$. So if Σ is the simplex bounded by $H_0, \ldots, H_n$, then $S(z,\varepsilon) \subseteq \Sigma$. Moreover, n of the facet defining inequalities for Σ are valid for F.

The diameter of Σ can be estimated by computing the distances of its vertices from the point u: they turn out to be $(n-1)\sqrt{\varepsilon(2t-\varepsilon)}$ (n times) and $\varepsilon(2t-\varepsilon)/(t-\varepsilon)$. Hence $\mathrm{diam}(\Sigma) < 2n\sqrt{2t\varepsilon}$.

To obtain a rational simplex with such properties and with small encoding length, round z to a rational vector $\tilde{z}$ with small encoding length, such that $\|z - \tilde{z}\| \leq \varepsilon/2$. Then we can write the facet defining inequalities of Σ in the form

$$c_i^T(x - \tilde{z}) \leq 1 \quad (i = 0, \ldots, n).$$

Since $\tilde{z} + \frac{\varepsilon}{2\|c_i\|}c_i \in S(z,\varepsilon) \subseteq \Sigma$, we know $\|c_i\| \leq 2/\varepsilon$.

Round each vector $(1 + \varepsilon/(8t))c_i$ to a rational vector $\tilde{c}_i$ with small encoding length such that

$$\left\| (1 + \frac{\varepsilon}{8t})c_i - \tilde{c}_i \right\| < \frac{\sqrt{\varepsilon}}{32nt\sqrt{t}}.$$

We claim that the following inequalities define a simplex Σ' with the right properties:

$$\tilde{c}_i^T(x - \tilde{z}) \leq 2 \quad (i = 0, \ldots, n).$$

It is straightforward to see that Σ' contains the ball $S(z,\varepsilon)$, and that all but one of these constraints are valid for $S(a,t)$. To see that $\mathrm{diam}(\Sigma') \leq 4n\sqrt{2t\varepsilon}$, we consider the simplex Σ'' obtained by blowing up Σ from $\tilde{z}$ by a factor of 2. If $\Sigma' \not\subseteq \Sigma''$ there exists a vector v in $\Sigma' \cap \Sigma''$ so that $c_i^T(v - \tilde{z}) = 2$ for some $i \in \{0, \ldots, n\}$ (as $\Sigma' \cap \Sigma'' \neq \emptyset$, since $\tilde{z} \in \Sigma' \cap \Sigma''$). Thus

$$\begin{aligned}
\tilde{c}_i^T(v - \tilde{z}) &= (1 + \frac{\varepsilon}{8t})c_i^T(v - \tilde{z}) + (\tilde{c}_i^T - (1 + \frac{\varepsilon}{8t})c_i)^T(v - \tilde{z}) \\
&\geq (1 + \frac{\varepsilon}{8t}) \cdot 2 - \|\tilde{c}_i^T - (1 + \frac{\varepsilon}{8t})c_i\| \cdot \|v - \tilde{z}\| \\
&\geq (1 + \frac{\varepsilon}{8t}) \cdot 2 - \frac{\sqrt{\varepsilon}}{33nt\sqrt{t}} \cdot 4n\sqrt{2t\varepsilon} \\
&> 2
\end{aligned}$$

(here we use that $\|v - \tilde{z}\| \leq \operatorname{diam}(\Sigma'') = 2\operatorname{diam}(\Sigma) < 4n\sqrt{2t\varepsilon}$), contradicting the fact that $v \in \Sigma'$. So $\Sigma' \subseteq \Sigma''$, and hence

$$\operatorname{diam}(\Sigma') \leq \operatorname{diam}(\Sigma'') = 2\operatorname{diam}(\Sigma) \leq 4n\sqrt{2t\varepsilon}.$$

□

(2.2.6) Lemma. *Let $\mathcal{K}$ be a class of named well-bounded convex bodies. Suppose that for any $(K; n, R, r) \in \mathcal{K}$, any rational numbers $\varepsilon > 0$ and $t > R$, and any $y \in \mathbb{Q}^n$ such that $y \in S(\operatorname{ex}_t(K), \varepsilon)$, there exists a certificate of the fact that $y \in S(K, 2\varepsilon)$ not longer than a polynomial of the encoding length. Then the weak membership problem for $\mathcal{K}$ is in $\mathcal{NP}$.*

Proof. Given any $(K; n, R, r) \in \mathcal{K}$, a rational number $\delta > 0$ and $y \in \mathbb{Q}^n$ such that $y \in S(K, -\delta)$, we have to find a proof of the fact that $y \in S(K, \delta)$ not longer than a polynomial in the encoding length $n + \langle R \rangle + \langle r \rangle + \langle \text{Name}(K) \rangle + \langle y \rangle + \langle \delta \rangle$ using the assumption of the lemma. To do this we will prove the existence of points $u_0, u_1, \ldots, u_n \in \mathbb{Q}^n$ such that:

(2.2.7) $u_0, \ldots, u_n \in S(K, \delta/2)$ and this fact can be shown in time polynomial in the encoding length,

(2.2.8) $y \in S\big(\operatorname{conv}(\{u_0, \ldots, u_n\}), \delta/2\big)$.

Clearly, if (2.2.7) and (2.2.8) hold, then $y \in S(K, \delta)$. For given vectors $y, u_0, \ldots, u_n \in \mathbb{Q}^n$, (2.2.8) can be verified in time polynomial in $\langle y \rangle + \langle u_0 \rangle + \ldots + \langle u_n \rangle + \langle \delta \rangle$ (this is left as an exercise). Note that (2.2.7) in particular implies that the numbers $\langle u_i \rangle$ are bounded by a polynomial in the encoding length. So if points $u_0, \ldots, u_n \in \mathbb{Q}^n$ satisfying (2.2.7) and (2.2.8) exist, $y \in S(K, \delta)$ has a polynomial length proof. The existence of such points can be shown as follows. Let $t = R^2/\delta$. Then it follows from Lemma (2.2.4) that

$$K \subseteq \operatorname{hull}_t(K) \subseteq S(\operatorname{conv}(\operatorname{ex}_t(K)), \delta)$$

and hence by (0.1.12),

$$S(K, -\delta) \subseteq \operatorname{conv}(\operatorname{ex}_t(K)).$$

In particular, $y \in \operatorname{conv}(\operatorname{ex}_t(K))$. By Carathéodory's theorem, there exist points $z_0, \ldots, z_n \in \operatorname{ex}_t(K)$ such that $y \in \operatorname{conv}\{z_0, \ldots, z_n\}$.

Now guess points $u_i \in S(z_i, \delta/4)$ such that $u_i \in \mathbb{Q}^n$ and $\langle u_i \rangle \leq \langle R \rangle + n + \langle \delta \rangle + 3$ (such u_i's clearly exist). Then, by hypothesis, there is a polynomial length proof of the fact that $u_i \in S(K, \delta/2)$ $(i = 0, \ldots, n)$. Moreover, trivially $y \in S(\operatorname{conv}\{u_0, \ldots, u_n\}, \delta/2)$. So (2.2.7) and (2.2.8) hold. □

(2.2.9) Theorem. *Let $\mathcal{K}$ be a class of named well-bounded convex bodies. If the weak violation problem for $\mathcal{K}$ is in $\mathcal{NP}$ then the weak membership problem for $\mathcal{K}$ is also in $\mathcal{NP}$.*

Proof. We prove that the condition of Lemma (2.2.6) is satisfied. Let $y \in S(\operatorname{ex}_t(K), \varepsilon)$; we describe a polynomial length certificate of the fact that $y \in S(K, 2\varepsilon)$. There

exists a vector $z \in \mathrm{ex}_t(K)$ such that $\|z-y\| \leq \varepsilon$. By Lemma (2.2.5), there exists a simplex $\Sigma \subseteq \mathbb{R}^n$ such that $\mathrm{diam}(\Sigma) \leq \varepsilon/3$, $S(z,\delta) \subseteq \Sigma$, where $\delta := \varepsilon^2/(96tn^2)$, all but one facet-inequalities are valid for K, and Σ is described by inequalities of polynomial encoding length. Suppose that all facet-inequalities except possibly $c^T x \leq \gamma$ are valid for K; we may assume that $\|c\|_\infty = 1$. Then consider the inequality $c^T x \geq \gamma - \delta\|c\|/2$. This inequality is not valid for $S(K,-\delta r/(8R))$, since, if it were, then by (0.1.14) the inequality $c^T x \geq \gamma - 3\delta\|c\|/4$ would be valid for K. But it is in fact violated by z.

Hence by the hypothesis that the weak violation problem is in $\mathcal{NP}$, there is a rational vector $u \in S(K,\delta)$ such that $c^T u \leq \gamma + \delta\|c\|$ and there exists a polynomial length certificate of the fact that $u \in S(K,\delta)$. Let Σ' be the simplex obtained by blowing up Σ from z by a factor of 2. Then $u \in \Sigma'$ and hence $\|u-z\| \leq \mathrm{diam}(\Sigma') \leq 2\varepsilon/3$. Hence $\|y-u\| \leq 5\varepsilon/3$.

So if we present the vector u, together with the (direct) computation showing that $\|y-u\| \leq 5\varepsilon/3$, and the above mentioned certificate for $u \in S(K,\delta) \subseteq S(K,\varepsilon/3)$, we have certified that $y \in S(K,2\varepsilon)$. □

The following result is in a sense dual to Theorem (2.2.9).

(2.2.10) Theorem. *Let $\mathcal{K}$ ba a class of named well-bounded convex bodies. If the weak separation problem for $\mathcal{K}$ is in $\mathcal{NP}$, then the weak validity problem for $\mathcal{K}$ is also in $\mathcal{NP}$.* □

The proof of this fact is analogous to the proof of Theorem (2.2.9) and is omitted. We prove one more result of this type.

(2.2.11) Theorem. *Let $\mathcal{K}$ be a class of named well-bounded convex bodies. The weak membership problem for $\mathcal{K}$ is in $\mathcal{NP} \cap$ co-$\mathcal{NP}$ if and only if the weak validity problem for $\mathcal{K}$ is in $\mathcal{NP} \cap$ co-$\mathcal{NP}$.*

Proof. We only give the proof of the "if" part, the reverse direction is analogous. So suppose that the weak validity problem for $\mathcal{K}$ is in $\mathcal{NP} \cap$co-$\mathcal{NP}$. It is obvious that the weak membership problem for $\mathcal{K}$ is in co-$\mathcal{NP}$. To show that it is in $\mathcal{NP}$, we use Lemma (2.2.6). So we have to show that for every $y \in S(\mathrm{ex}_t(K),\varepsilon)$ there exists a concise certificate of the fact that $y \in S(K,2\varepsilon)$. This is done quite analogously to the previous proof. Let $z \in \mathrm{ex}_t(K)$ be a vector such that $\|y-z\| \leq \varepsilon$. By Lemma (2.2.5), there exists a simplex Σ such that $\mathrm{diam}(\Sigma) \leq \varepsilon/3$ and $S(z,\delta) \subseteq \Sigma$, where $\delta = \varepsilon^2/(200tn^2)$. Moreover, Σ is described by inequalities

$$c_i^T x \leq \gamma_i \quad (i = 0, \ldots, n)$$

of polynomial encoding length for which $c_i^T x \leq \gamma_i$ is valid for K if $i = 1, \ldots, n$.

Since, just as in the previous proof, the inequality $c_0^T x \geq \gamma_0 - \delta\|c_0\|/2$ is not valid for $S(K,-\delta r/(8R))$, it follows by the hypothesis that the weak validity problem is in co-$\mathcal{NP}$, that there exists a polynomial length certificate "C_0" of the fact that the inequality $c_0^T x \geq \gamma_0 + \delta\|c_0\|/2$ is not valid for $S(K,\delta/2)$. Hence $c_0^T x \geq \gamma_0 + \delta\|c_0\|$ is not valid for K. Furthermore, $c_i^T x \leq \gamma_i$ $(1 \leq i \leq n)$ is valid

for K and so $c_i^T x \leq \gamma_i + \delta \|c_i\|/4$ is valid for all $x \in S(K, \delta/4)$. Hence, as the weak validity problem for $\mathcal{K}$ is in $\mathcal{NP}$, there exists a polynomial length certificate "C_i" of the fact that $c_i^T x \leq \gamma_i + \delta \|c_i\|/2$ is valid for $S(K, -\delta r/(8R))$. This also certifies, by (0.1.14), that $c_i^T x \leq \gamma_i + \delta \|c_i\|$ is valid for K.

Let Σ' be the solution set of

$$c_i^T x \leq \gamma_i + \delta \|c_i\| \quad (i = 0, \ldots, n).$$

The certificates $C_0, \ldots, C_n$ imply that K contains a point from Σ'. Moreover, $\operatorname{diam}(\Sigma') \leq \varepsilon$ since Σ' is contained in the simplex obtained by blowing up Σ by a factor of 2 from z. Hence $\Sigma' \subseteq S(y, 2\varepsilon)$. So the certificates $C_0, \ldots, C_n$, together with a (direct) computation showing that $\Sigma' \subseteq S(y, 2\varepsilon)$, give a certificate of the fact that $y \in S(K, 2\varepsilon)$. □

Chapter 3

The Ellipsoid Method

In 1979 a note of L. G. Khachiyan indicated how an algorithm, the so-called **ellipsoid method**, originally devised for nonlinear nondifferentiable optimization, can be modified in order to check the feasibility of a system of linear inequalities in polynomial time. This result caused great excitement in the world of mathematical programming since it implies the polynomial time solvability of linear programming problems.

This excitement had several causes. First of all, many researchers all over the world had worked hard on the problem of finding a polynomial time algorithm for linear programming for a long time without success. So a really major open problem had been solved.

Secondly, many people believe that $\mathcal{P} = \mathcal{NP} \cap \text{co-}\mathcal{NP}$ – cf. Section 1.1 – and the linear programming problem was one of the few problems known to belong to $\mathcal{NP} \cap \text{co-}\mathcal{NP}$ but that had not been shown to be in $\mathcal{P}$. Thus, a further indication for the correctness of this conjecture was obtained.

Thirdly, the ellipsoid method together with the additional number theoretical "tricks" was so different from all the algorithms for linear programming considered so far that the method itself and the correctness proof were a real surprise.

Fourthly, the ellipsoid method, although "theoretically efficient", did not prove to be "practically efficient". Therefore, controversies in complexity theory about the value of polynomiality of algorithms and about how to measure encoding lengths and running times – cf. Chapter 1 – were put into focus.

For almost all presently known versions of the simplex method, there exist a few (artificial) examples for which this algorithm has exponential running time. The first examples of this type have been discovered by KLEE and MINTY (1972). Such bad examples do not exist for the ellipsoid method. But the ellipsoid method has been observed to be much slower than the simplex algorithm on the average in practical computation. In fact, BORGWARDT (1982) has shown that the expected running time of a version of the simplex method is polynomial and much better than the running time of the ellipsoid method. Although the ellipsoid method does not seem to be a breakthrough in applied linear programming, it is of value in nonlinear (in particular nondifferentiable) optimization – see for instance ECKER and KUPFERSCHMID (1983).

As mentioned, nonlinear optimization is one of the roots of the ellipsoid method. The method grew out of work in convex nondifferential optimization (relaxation, subgradient, space dilatation methods, methods of central sections) as well as of studies on computational complexity of convex programming

problems. The history of the ellipsoid method and its antecedents has been covered extensively by BLAND, GOLDFARB and TODD (1981) and SCHRADER (1982). Briefly, the development was as follows.

Based on his earlier work, SHOR (1970a,b) described a new gradient projection algorithm with space dilatation for convex nondifferential programming. YUDIN and NEMIROVSKIĬ (1976a,b) observed that Shor's algorithm provides an answer to a problem discussed by LEVIN (1965) and – in a somewhat cryptical way – gave an outline of the ellipsoid method. The first explicit statement of the ellipsoid method, as we know it today, is due to SHOR (1977). In the language of nonlinear programming, it can be viewed as a rank-one update algorithm and is quite analogous to a variable metric quasi-Newton method – see GOFFIN (1984) for such interpretations of the ellipsoid method. This method was adapted by KHACHIYAN (1979) to state the polynomial time solvability of linear programming. The proofs appeared in KHACHIYAN (1980). Khachiyan's 1979-paper stimulated a flood of research aiming at accelerating the method and making it more stable for numerical purposes – cf. BLAND, GOLDFARB and TODD (1981) and SCHRADER (1982) for surveys. We will not go into the numerical details of these modifications. Our aim is to give more general versions of this algorithm which will enable us to show that the problems discussed in Chapter 2 are equivalent with respect to polynomial time solvability and, by applying these results, to unify various algorithmic approaches to combinatorial optimization. The applicability of the ellipsoid method to combinatorial optimization was discovered independently by KARP and PAPADIMITRIOU (1980), PADBERG and RAO (1981), and GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981).

We do not believe that the ellipsoid method will become a true competitor of the simplex algorithm for practical calculations. We do, however, believe that the ellipsoid method has fundamental theoretical power since it is an elegant tool for proving the polynomial time solvability of many geometric and combinatorial optimization problems.

YAMNITSKI and LEVIN (1982) gave an algorithm – in the spirit of the ellipsoid method and also based on the research in the Soviet Union mentioned above – in which ellipsoids are replaced by simplices. This algorithm is somewhat slower than the ellipsoid method, but it seems to have the same theoretical applicability.

Khachiyan's achievement received an attention in the nonscientific press that is – to our knowledge – unpreceded in mathematics. Newspapers and journals like The Guardian, Der Spiegel, Nieuwe Rotterdamsche Courant, Népszabadság, The Daily Yomiuri wrote about the "major breakthrough in the solution of real-world problems". The ellipsoid method even jumped on the front page of The New York Times: "A Soviet Discovery Rocks World of Mathematics" (November 7, 1979). Much of the excitement of the journalists was, however, due to exaggerations and misinterpretations – see LAWLER (1980) for an account of the treatment of the implications of the ellipsoid method in the public press.

Similar attention has recently been given to the new method of KARMARKAR (1984) for linear programming. Karmarkar's algorithm uses an approach different from the ellipsoid method and from the simplex method. Karmarkar's algorithm has a better worst-case running time than the ellipsoid method, and it seems that this method runs as fast or even faster than the simplex algorithm in practice.

But Karmarkar's algorithm requires – like the simplex method – the complete knowledge of the constraint system for the linear programmming problem. And thus – as far as we can see – it cannot be used to derive the consequences to be discussed in this book.

The unexpected theoretical and practical developments in linear programming in the recent years have prompted a revival of research in this area. The nonlinear approach to linear programming – using techniques like the Newton method and related descent procedures – receives particular attention. A number of further polynomial time methods for the solution of linear programming problems have been suggested – see for instance IRI and IMAI (1986), DE GHELLINCK and VIAL (1986), BETKE and GRITZMANN (1986), and SONNEVEND (1986) – and are under investigation with respect to their theoretical and practical behaviour. It is conceivable that careful implementations of these methods and, possibly, combinations of these methods with the simplex algorithm will lead to good codes for linear programming problems. Such codes may become serious competitors for the simplex codes that presently dominate the "LP-market". A thorough computational and theoretical study of such prospects is the recent paper GILL, MURRAY, SAUNDERS, TOMLIN and WRIGHT (1986), where variants of Karmarkar's algorithm are discussed in a general framework of projected Newton barrier methods and where these are compared with several versions of the simplex method with respect to practical efficiency for various classes of LP-problems.

3.1 Geometric Background and an Informal Description

The purpose of this section is to explain the geometric idea behind the ellipsoid method, to give an informal description of it, to demonstrate some proof techniques, and to discuss various modifications. We begin by summarizing well-known geometric facts about ellipsoids. Then we describe the ellipsoid method for the special case of finding a point in a polytope that is explicitly given by linear inequalities and known to be empty or full-dimensional. We also present proofs of a few basic lemmas, and finally, computational aspects of the ellipsoid method are discussed, in particular questions of "rounding" and quicker "shrinking". Proofs of more general results than described in this section can be found in Sections 3.2 and 3.3.

The problems we address are trivial for the case of one variable. So we assume in the proofs throughout Chapter 3 that $n \geq 2$.

Properties of Ellipsoids

A set $E \subseteq \mathbb{R}^n$ is an **ellipsoid** if there exists a vector $a \in \mathbb{R}^n$ and a positive definite $n \times n$-matrix A such that

$$E = E(A,a) := \{x \in \mathbb{R}^n \mid (x-a)^T A^{-1}(x-a) \leq 1\}. \tag{3.1.1}$$

(It will become clear soon that using A^{-1} here, which is also a positive definite matrix, is more convenient than using A.) Employing the ellipsoidal norm $\| \ \|_A$

defined in Section 0.1 we can write equivalently

$$E(A,a) = \{x \in \mathbb{R}^n \mid \|x-a\|_A \leq 1\}, \tag{3.1.2}$$

that is, the ellipsoid $E(A,a)$ is the unit ball around a in the vector space $\mathbb{R}^n$ endowed with the norm $\| \ \|_A$. So in particular, the unit ball $S(0,1)$ around zero (in the Euclidean norm) is the ellipsoid $E(I,0)$. Note that E determines A and a uniquely. The vector a is called the **center** of E, and we say that $E(A,a)$ is the **ellipsoid associated with** A **and** a.

For every positive definite matrix A there exists a unique positive definite matrix, denoted by $A^{1/2}$, such that $A = A^{1/2}A^{1/2}$. It follows by a simple calculation that

$$E(A,a) = A^{1/2}S(0,1) + a. \tag{3.1.3}$$

Thus every ellipsoid is the image of the unit ball under a bijective affine transformation.

There are some interesting connections between geometric properties of the ellipsoid $E = E(A,a)$ and algebraic properties of the matrix A which we want to point out here. Recall from (0.1.3) that all eigenvalues of A are positive reals. The diameter of E is the length of a longest axis and is equal to $2\sqrt{\Lambda}$, where Λ is the largest eigenvalue of A. The longest axes of E correspond to the eigenvectors belonging to Λ. The width of E is the length of a shortest axis which is $2\sqrt{\lambda}$, where λ is the smallest eigenvalue of A. These observations imply that the ball $S(a,\sqrt{\lambda})$ is the largest ball contained in $E(A,a)$ and that $S(a,\sqrt{\Lambda})$ is the smallest ball containing $E(A,a)$. In fact, this is the geometric contents of inequality (0.1.9). Moreover, the axes of symmetry of E correspond to the eigenvectors of A.

Figure 3.1 shows an ellipsoid graphically. It is the ellipsoid $E(A,0) \subseteq \mathbb{R}^2$ with $A = \operatorname{diag}((16,4)^T)$. The eigenvalues of A are $\Lambda = 16$ and $\lambda = 4$ with corresponding eigenvectors $e_1 = (1,0)^T$ and $e_2 = (0,1)^T$. So the diameter of $E(A,0)$ is $2\sqrt{\Lambda} = 8$, while the width is $2\sqrt{\lambda} = 4$.

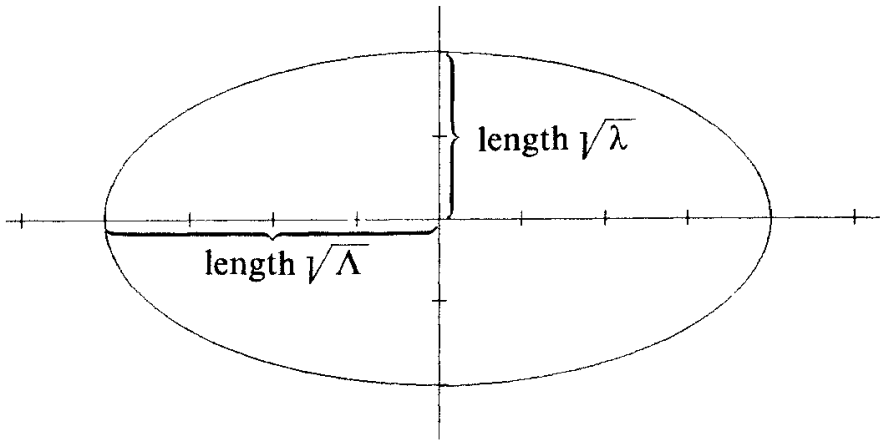


Figure 3.1

The volume of an ellipsoid $E = E(A,a)$, denoted by $\operatorname{vol}(E)$, depends only on the determinant of A and on the dimension of the space. More exactly, we have

$$\operatorname{vol}(E(A,a)) = \sqrt{\det A} \cdot V_n, \tag{3.1.4}$$

where V_n is the volume of the unit ball $S(0,1)$ in $\mathbb{R}^n$. It is well known that

$$V_n = \frac{\pi^{n/2}}{\Gamma(n/2+1)} \sim \frac{1}{\sqrt{\pi n}}\left(\frac{2e\pi}{n}\right)^{n/2}, \tag{3.1.5}$$

where

$$\Gamma(x) := \int_0^\infty e^{-t}t^{x-1}\,dt, \quad \text{for } x > 0$$

is the **gamma-function**. The gamma-function satisfies

$$\Gamma(n) = (n-1)! \quad \text{for all } n \in \mathbb{N}.$$

We will frequently need bounds on V_n. It turns out that for our purposes it will be enough to use the very rough estimates

$$n^{-n} \leq V_n \leq 2^n, \tag{3.1.6}$$

which are derived from the facts that $S(0,1)$ contains $\{x \in \mathbb{R}^n \mid 0 \leq x_i \leq 1/n, i = 1, \ldots, n\}$ and that $S(0,1)$ is contained in $\{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$.

If $x \longmapsto Dx + d$ is a bijective affine transformation T then $\operatorname{vol}(T(E(A,a))) = \det D\sqrt{\det A} \cdot V_n$. This in particular shows that

$$\frac{\operatorname{vol}(E(A,a))}{\operatorname{vol}(E(B,b))} = \frac{\operatorname{vol}(T(E(A,a)))}{\operatorname{vol}(T(E(B,b)))},$$

that is, the quotient of the volumes of two ellipsoids is invariant under bijective affine transformations.

It will be necessary for us to optimize linear objective functions over ellipsoids. This is easy and can be derived from the obvious fact that, for $c \neq 0$, $\max c^T x$, $x \in S(a,1)$ is achieved at the vector $a + c/\|c\|$. Namely, suppose that $E(A,a) \subseteq \mathbb{R}^n$ is an ellipsoid and let $c \in \mathbb{R}^n \setminus \{0\}$. Set $Q := A^{1/2}$. Recall from (3.1.3) that $Q^{-1}E(A,a) = S(0,1) + Q^{-1}a = S(Q^{-1}a, 1)$ and thus

$$\begin{aligned}
\max\{c^T x \mid x \in E(A,a)\} &= \max\{c^T QQ^{-1}x \mid Q^{-1}x \in Q^{-1}E(A,a)\} \\
&= \max\{c^T Qy \mid y \in S(Q^{-1}a, 1)\} \\
&= c^T Q\frac{1}{\|Qc\|}Qc + c^T QQ^{-1}a \\
&= c^T \frac{1}{\sqrt{c^T Ac}}Ac + c^T a \\
&= c^T a + \sqrt{c^T Ac}.
\end{aligned}$$

By setting

$$\begin{aligned}
b &:= \frac{1}{\sqrt{c^T Ac}}Ac, \\
z_{\max} &:= a + b, \\
z_{\min} &:= a - b,
\end{aligned} \tag{3.1.7}$$

we therefore obtain

$$(3.1.8)\quad c^T z_{\max} = \max\{c^T x \mid x \in E(A,a)\} = c^T a + \sqrt{c^T A c} = c^T a + \|c\|_{A^{-1}},$$
$$c^T z_{\min} = \min\{c^T x \mid x \in E(A,a)\} = c^T a - \sqrt{c^T A c} = c^T a - \|c\|_{A^{-1}}.$$

This means that $z_{\max}$ maximizes and $z_{\min}$ minimizes $c^T x$ over $E(A,a)$. The line connecting $z_{\max}$ and $z_{\min}$ goes through the center a of $E(A,a)$ and has direction b. In Figure 3.2 we show the ellipsoid $E(A,a)$ with $A = \operatorname{diag}((16,4)^T)$, $a^T = (1,0.5)$ and objective function $c^T = (-2,-3)$. From (3.1.7) we obtain $b^T = (-3.2,-1.2)$, $z_{\max}^T = (1,0.5) + (-3.2,-1.2) = (-2.2,-0.7)$ and $z_{\min}^T = (1,0.5) - (-3.2,-1.2) = (4.2,1.7)$.

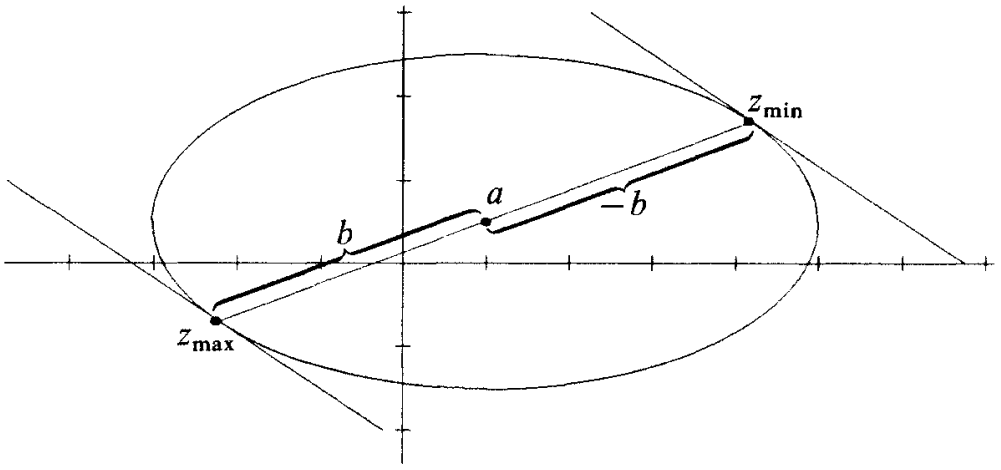


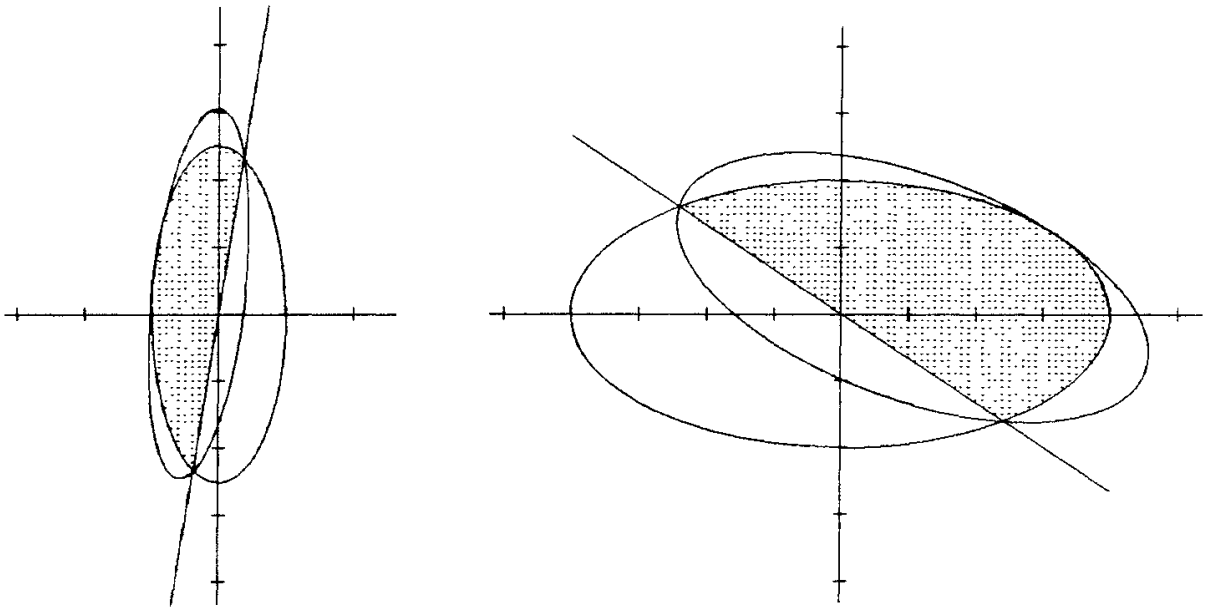
Figure 3.2

It is a well-known fact that every convex body is contained in a unique ellipsoid of minimal volume and contains a unique ellipsoid of maximal volume. Apparently, these two results have been discovered independently by several mathematicians – see for instance DANZER, GRÜNBAUM and KLEE (1963, p. 139). In particular, these authors attribute the first result to K. Löwner. JOHN (1948) proved the following more general theorem, the "moreover" part of which will be of interest later.

(3.1.9) Theorem. *For every convex body $K \subseteq \mathbb{R}^n$ there exists a unique ellipsoid E of minimal volume containing K. Moreover, K contains the ellipsoid obtained from E by shrinking it from its center by a factor of n.* □

Let us call the minimum volume ellipsoid containing a convex body K the **Löwner-John ellipsoid** of K. In formulas, the second part of Theorem (3.1.9) states that, if $E(A,a)$ is the Löwner-John ellipsoid of K, then K contains the ellipsoid $E(n^{-2}A,a)$.

For a regular simplex S, the Löwner-John ellipsoid is a ball $E(R^2 I,a)$ with an appropriate radius R around the center of gravity a of S. The concentrical ball $E(n^{-2}R^2 I,a)$ is the largest ellipsoid contained in S. This shows that the parameter n in Theorem (3.1.9) is best possible.



(a) (b)

Figure 3.3

In general, the Löwner-John ellipsoid of a convex body K is hard to compute. We will, however, show in Section 4.6 that, under certain assumptions on K, good approximations of it can be computed in polynomial time. In the ellipsoid method and its variants, Löwner-John ellipsoids of certain ellipsoidal sections are used; and for these Löwner-John ellipsoids, explicit formulas are known. We will describe some of them.

Suppose $E(A,a)$ is an ellipsoid and $c \in \mathbb{R}^n \setminus \{0\}$. Set

$$E'(A,a,c) := E(A,a) \cap \{x \in \mathbb{R}^n \mid c^T x \leq c^T a\}. \tag{3.1.10}$$

So $E'(A,a,c)$ is one half of the ellipsoid $E(A,a)$ obtained by cutting $E(A,a)$ through the center a using the hyperplane $\{x \in \mathbb{R}^n \mid c^T x = c^T a\}$. The Löwner-John ellipsoid of $E'(A,a,c)$ is the ellipsoid $E(A',a')$ given by the following formulas:

$$a' := a - \frac{1}{n+1} b, \tag{3.1.11}$$

$$A' := \frac{n^2}{n^2-1}\left(A - \frac{2}{n+1} bb^T\right), \tag{3.1.12}$$

where b is the vector defined in (3.1.7). In Figure 3.3 we show the Löwner-John ellipsoids $E(A',a')$ of two halfellipsoids $E'(A,a,c)$, where in (a), $A = \mathrm{diag}((1,25/4)^T)$, $a = 0$, $c = (25,-4)$ and in (b), $A = \mathrm{diag}((16,4)^T)$, $a = 0$, $c = (-2,-3)^T$. The halfellipsoids $E'(A,a,c)$ are the dotted regions.

Note that the center a' of the Löwner-John ellipsoid $E(A',a')$ of $E'(A,a,c)$ lies on the line through the vectors $z_{\max}$ and $z_{\min}$ – see (3.1.7). More exactly, one can

get from a to a' by making a step from a towards $z_{\min}$ of length $\frac{1}{n+1}\|z_{\min} - a\|$. Moreover, the boundary of $E(A', a')$ touches $E'(A, a, c)$ in the point $z_{\min}$ and in the set $\{x \mid \|x - a\|_A = 1\} \cap \{x \mid c^T x = c^T a\}$. In $\mathbb{R}^2$ the last set consists of two points only – see Figure 3.3 – while in $\mathbb{R}^3$ this is an ellipse.

We will see later that the algorithm described by Khachiyan makes use of the Löwner-John ellipsoid of $E'(A, a, c)$. There are modifications of this method using Löwner-John ellipsoids of other ellipsoidal sections. Here we will describe those that we use later – see BLAND, GOLDFARB and TODD (1981) for further details. So suppose $E(A, a)$ is an ellipsoid and $c \in \mathbb{R}^n \setminus \{0\}$, $\gamma \in \mathbb{R}$. It follows from (3.1.8) that the hyperplane

$$H := \{x \in \mathbb{R}^n \mid c^T x = \gamma\}$$

has a nonempty intersection with $E(A, a)$ if and only if $c^T z_{\min} \leq \gamma \leq c^T z_{\max}$, that is, if and only if

$$|c^T a - \gamma| \leq \sqrt{c^T A c}.$$

For notational convenience, let us set

$$\alpha := \frac{c^T a - \gamma}{\sqrt{c^T A c}}. \tag{3.1.13}$$

Then H intersects $E(A, a)$ if and only if

$$-1 \leq \alpha \leq 1. \tag{3.1.14}$$

The number α can be interpreted as the signed distance of the center a from the boundary of the halfspace $\{x \mid c^T x \leq \gamma\}$ in the space $\mathbb{R}^n$ endowed with the norm $\| \|_{A^{-1}}$. (The distance is nonpositive if a is contained in the halfspace.)

We want to cut $E(A, a)$ into two pieces using H and to compute the Löwner-John ellipsoid of the piece contained in $\{x \mid c^T x \leq \gamma\}$. For

$$E'(A, a, c, \gamma) := E(A, a) \cap \{x \in \mathbb{R}^n \mid c^T x \leq \gamma\}, \tag{3.1.15}$$

the Löwner-John ellipsoid $E(A', a')$ can be determined as follows.

If $-1 \leq \alpha \leq -1/n$ *then* $E(A', a') = E(A, a)$.

If $-1/n \leq \alpha < 1$ *then* $E(A', a')$ *is given by*

$$a' := a - \frac{1 + n\alpha}{n+1} b, \tag{3.1.16}$$

$$A' := \frac{n^2}{n^2 - 1}(1 - \alpha^2)\left(A - \frac{2(1 + n\alpha)}{(n+1)(1+\alpha)} bb^T\right), \tag{3.1.17}$$

where b is the vector defined in (3.1.7). Note that if $\gamma = c^T a$ then $E'(A, a, c, \gamma) = E'(A, a, c)$ and formulas (3.1.16) and (3.1.17) reduce to formulas (3.1.11) and

(3.1.12). So in formulas (3.1.11) and (3.1.12) we compute the Löwner-John ellipsoid of the section $E(A,a,c)$ of $E(A,a)$ that is obtained by cutting with H through the center a. We call this a **central cut**. If $0 < \alpha < 1$ then $E'(A,a,c,\gamma)$ is strictly contained in $E'(A,a,c)$. This means that we cut off a larger piece of $E(A,a)$, and therefore $c^T x \leq \gamma$ is called a **deep cut**. If $-1/n < \alpha < 0$ then we leave more of $E(A,a)$ than the "half" $E'(A,a,c)$, and we call $c^T x \leq \gamma$ a **shallow cut**; but in this case the Löwner-John ellipsoid of $E'(A,a,c,\gamma)$ is still strictly smaller than $E(A,a)$, in the sense that it has a smaller volume. (We shall see later that a volume reduction argument will prove the polynomial time termination of the ellipsoid method.)

It is also possible to compute explicit formulas for the Löwner-John ellipsoid of $E(A,a) \cap \{x \mid \gamma' \leq c^T x \leq \gamma\}$, i. e., for an ellipsoidal section determined by **parallel cuts**. However, the formulas are quite horrible – see BLAND, GOLDFARB and TODD (1981). We will need a special case of this, namely the Löwner-John ellipsoid of an ellipsoidal section determined by **centrally symmetric parallel cuts**, i. e., of

$$E''(A,a,c,\gamma) := E(A,a) \cap \{x \in \mathbb{R}^n \mid c^T a - \gamma \leq c^T x \leq c^T a + \gamma\}, \tag{3.1.18}$$

where $\gamma > 0$. Similarly as in (3.1.13), let $\alpha = -\gamma/\sqrt{c^T A c}$. It turns out that the Löwner-John ellipsoid of $E''(A,a,c,\gamma)$ is the ellipsoid $E(A',a')$ defined as follows.

If $\alpha < -1/\sqrt{n}$ *then* $E(A',a') = E(A,a)$.

If $-1/\sqrt{n} \leq \alpha < 0$ *then* $E(A',a')$ *is given by*

$$a' := a, \tag{3.1.19}$$

$$A' := \frac{n}{n-1}(1-\alpha^2)\left(A - \frac{1-n\alpha^2}{1-\alpha^2} bb^T\right). \tag{3.1.20}$$

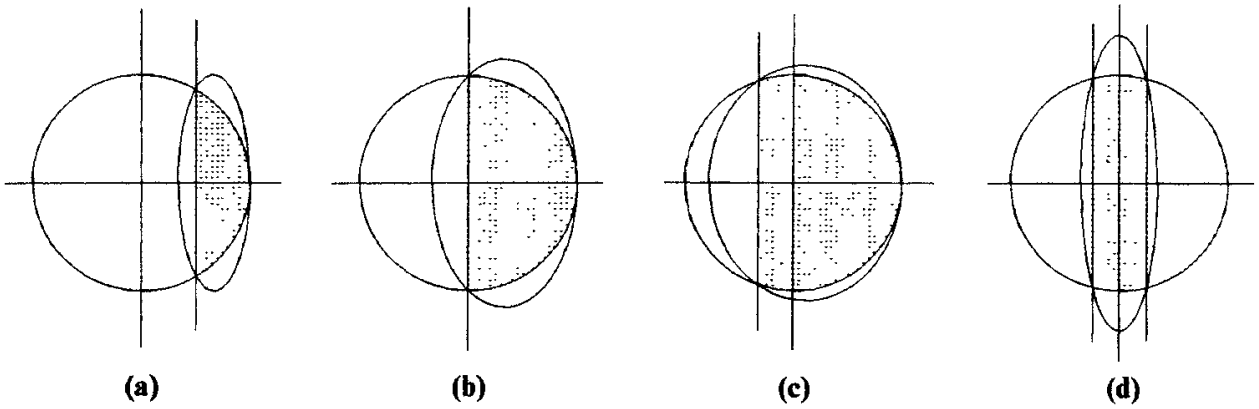


Figure 3.4a–d

Figure 3.4 shows the four types of cuts used in various versions of the ellipsoid method. In all cases we assume that $E(A,a)$ is the unit ball $S(0,1)$. Let $c :=$

$(-1,0)^T$ be the vector used to cut through $E(A,a)$. Picture (b) shows the Löwner-John ellipsoid of the dotted area $E'(A,a,c) = S(0,1) \cap \{x \mid x_1 \geq 0\}$, a central cut. Picture (a) shows the Löwner-John ellipsoid of the dotted set $E'(A,a,c,-1/2) = S(0,1) \cap \{x \mid x_1 \geq 1/2\}$, a deep cut. Picture (c) shows the Löwner-John ellipsoid of $E'(A,a,c,1/3) = S(0,1) \cap \{x \mid x_1 \geq -1/3\}$, a shallow cut; while the Löwner-John ellipsoid of $E''(A,a,c,1/4) = S(0,1) \cap \{x \mid -1/4 \leq x_1 \leq 1/4\}$ is displayed in picture (d). This set is determined by a centrally symmetric parallel cut.

Description of the Basic Ellipsoid Method

To show the basic idea behind the ellipsoid method, we describe it now as a method to solve the strong nonemptiness problem for explicitly given polytopes that are either empty or full-dimensional. The input for our algorithm is a **system of inequalities** $c_i^T x \leq \gamma_i$, $i = 1, \ldots, m$, with n variables in **integral coefficients**. We would like to determine whether

$$P := \{x \in \mathbb{R}^n \mid c_i^T x \leq \gamma_i,\ i = 1, \ldots, m\} = \{x \mid Cx \leq d\} \tag{3.1.21}$$

is empty or not, and if it is nonempty, we would like to find a point in P. In order to get a correct answer the input must be accompanied by the following guarantees:

(3.1.22) P is bounded.

(3.1.23) If P is nonempty, then P is full-dimensional.

It will turn out later that the certificates (3.1.22) and (3.1.23) are not necessary. The (appropriately modified) method also works for possibly unbounded polyhedra that are not full-dimensional. Moreover, we can handle polyhedra defined by inequalities that are provided by a separation oracle, that is, the inequalities need not be given in advance. To treat all these additional possibilities now would only obscure the lines of thought. Thus, in order to explain the ideas underlying the ellipsoid method, we restrict ourselves to the special case described above.

Recall the well-known method of catching a lion in the Sahara. It works as follows. Fence in the Sahara, and split it into two parts; check which part does not contain the lion, fence the other part in, and continue. After a finite number of steps we will have caught the lion – if there was any – because the fenced-in zone will be so small that the lion cannot move anymore. Or we realize that the fenced-in zone is so small that it cannot contain any lion, i. e., there was no lion at all. In order to illustrate the ellipsoid method by this old hunter's story we have to describe what our Sahara is, how we split it into two parts, how we fence these in, and when we can declare the lion caught or nonexistent.

For the Sahara we choose a ball around the origin, say $S(0,R)$, that contains our polytope P, the lion. If the system of inequalities (3.1.21) contains explicit upper and lower bounds on the variables, say $l_i \leq x_i \leq u_i$, $i = 1, \ldots, n$ then a radius R with $P \subseteq S(0,R)$ is easily found. Take for instance

$$R := \sqrt{\textstyle\sum_{i=1}^n \max\{u_i^2, l_i^2\}}. \tag{3.1.24}$$

If bounds on the variables are not given explicitly we can use the information that C and d are integral and that P is a polytope to compute such a radius R, namely we have:

(3.1.25) Lemma. *$P \subseteq S(0,R)$, where $R := \sqrt{n}\, 2^{\langle C,d\rangle - n^2}$.* □

Recall that $\langle C,d\rangle$ denotes the encoding length of C and d – see Section 1.3. (Since we do not want to interrupt the flow of thought, the proofs of all lemmas stated in this subsection are postponed to the next subsection.)

Now we have the starting point for the ellipsoid method. We know that P is in $S(0,R)$ with R given by (3.1.24) or (3.1.25). This ball around the origin will be our first ellipsoid $E(A_0,a_0)$ (which is clearly given by setting $A_0 := R^2 I$ and $a_0 = 0$).

Let us now describe the k-th step, $k \geq 0$, of the procedure. By construction, the current ellipsoid

$$E_k := E(A_k,a_k) \tag{3.1.26}$$

contains P. The ellipsoid E_k has one distinguished point, its center a_k, and we have it explicitly at hand. So we can substitute a_k into the inequality system (3.1.21) and check whether all inequalities are satisfied. If this is the case, we can stop, having found a feasible solution. If the center is not feasible, then at least one inequality of the system (3.1.21) must be violated, let us say $c^T x \leq \gamma$. So we have $c^T a_k > \gamma$. The hyperplane $\{x \mid c^T x = c^T a_k\}$ through the center a_k of E_k cuts E_k into two "halves", and we know from the construction that the polytope P is contained in the half

$$E'(A_k,a_k,c) = \{x \in E(A_k,a_k) \mid c^T x \leq c^T a_k\}. \tag{3.1.27}$$

Therefore we choose, as the next ellipsoid E_{k+1} in our sequence, the Löwner-John ellipsoid of $E'(A_k,a_k,c)$, which is given by formulas (3.1.11) and (3.1.12). And we continue this way by successively including P into smaller and smaller ellipsoids.

The question to be answered now is: When can we stop? Clearly, if we find a point in P, we terminate. But how long do we have to continue the iterations if no feasible point is obtained? The stopping criterion comes from a volume argument. Namely, we know the initial ellipsoid $S(0,R)$ and therefore we can estimate its volume, e. g., through (3.1.6). In each step k we construct a new ellipsoid E_{k+1} whose volume is strictly smaller than that of E_k. More precisely, one can show:

(3.1.28) Lemma.

$$\frac{\mathrm{vol}(E_{k+1})}{\mathrm{vol}(E_k)} = \left(\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}\right)^{1/2} < e^{-1/(2n)} < 1.$$

□

By our guarantees (3.1.22) and (3.1.23) we are sure that our polytope P has a positive volume, unless it is empty. One can use the integrality of the data and the fact that P is a polytope to prove

(3.1.29) Lemma. *If the polytope P is full-dimensional, then*

$$\text{vol}(P) \geq 2^{-(n+1)\langle C\rangle + n^3}.$$

□

Now we can finish our analysis. We can estimate the volume of the initial ellipsoid from above, and the volume of P, if P is nonempty, from below; and we know the shrinking rate of the volumes. Therefore, we iterate until the present ellipsoid has a volume that is smaller than the lower bound (3.1.29) on the volume of P. If we have reached this situation, in step N say, we can stop since we know that

$$P \subseteq E_N, \quad \text{vol}(E_N) < \text{vol}(P). \tag{3.1.30}$$

This is clearly impossible, and we can conclude that P is empty. The number N of iterations that have to be done in the worst case can be estimated as follows. If we choose

$$N := 2n((2n+1)\langle C\rangle + n\langle d\rangle - n^3) \tag{3.1.31}$$

then it is easy to see that $\text{vol}(E_N) < 2^{-(n+1)\langle C\rangle + n^3}$ (for an elaboration, see Lemma (3.1.36)). Combining this with (3.1.29) we see that (3.1.30) holds for this N. Our description of the ellipsoid method (except for implementational details) is complete and we can summarize the procedure.

(3.1.32) The Basic Ellipsoid Method (for the strong nonemptiness problem for full-dimensional or empty polytopes).

Input: *An $m \times n$-inequality system $Cx \leq d$ with integral coefficients. We assume that the data are accompanied by a guarantee that $P = \{x \in \mathbb{R}^n \mid Cx \leq d\}$ is bounded and either empty or full-dimensional.*

Initialization: *Set*

(a) $k := 0$,
(b) $N := 2n((2n+1)\langle C\rangle + n\langle d\rangle - n^3)$,
(c) $A_0 := R^2 I$ with $R := \sqrt{n}\, 2^{\langle C,d\rangle - n^2}$ (or use any valid smaller R as, for example, given in (3.1.24)),
$a_0 := 0$ (so $E_0 := E(A_0, a_0)$ is the initial ellipsoid).

General Step:

(d) *If $k = N$, STOP!* (Declare P empty.)
(e) *If $a_k \in P$, STOP!* (A feasible solution is found.)
(f) *If $a_k \notin P$, then choose an inequality, say $c^T x \leq \gamma$, of the system $Cx \leq d$ that is violated by a_k.*

Set

(g) $$b := \frac{1}{\sqrt{c^T A_k c}} A_k c \qquad \text{(see (3.1.7))},$$

(h) $$a_{k+1} := a_k - \frac{1}{n+1} b \qquad \text{(see (3.1.11))},$$

(i) $$A_{k+1} := \frac{n^2}{n^2-1}\left(A_k - \frac{2}{n+1} bb^T\right) \qquad \text{(see (3.1.12))},$$

and go to (d). □

We call algorithm (3.1.32) the **basic** ellipsoid method since it contains all the fundamental ideas of the procedure. To make it a polynomial time algorithm or to make it more efficient certain technical details have to be added. For instance, we have to specify how the vector a_{k+1} in (h) is to be calculated. Our formula on the right hand side of (h) leads to irrational numbers in general since a square root is computed in the formula for b in (g). Nevertheless, the lemmas stated before prove that the basic ellipsoid method works – provided that the single steps of algorithm (3.1.32) can be implemented correctly (and efficiently). We will discuss this issue in the subsection after the next one.

Proofs of Some Lemmas

We now give the proofs of the lemmas stated above. We begin with a slight generalization of Lemma (3.1.25).

(3.1.33) Lemma. *If $P = \{x \in \mathbb{R}^n \mid Cx \leq d\}$ is a polyhedron and C, d are integral, then all vertices of P are contained in the ball around 0 with radius $R := \sqrt{n}\, 2^{\langle C,d\rangle - n^2}$. In particular, if P is a polytope then $P \subseteq S(0,R)$.*

Proof. In order to show that the vertices of P are in $S(0,R)$ with the R given above, we estimate the largest (in absolute value) component of a vertex of P. Clearly, if we can find a number $t \geq 0$ such that no vertex of P has a component which (in absolute value) is larger than t, then the vertices of P are contained in $\{x \mid -t \leq x_i \leq t, i = 1, \ldots, n\}$, and thus in $S(0, \sqrt{n}\, t)$. From polyhedral theory we know that for each vertex v of P there is a nonsingular subsystem of $Cx \leq d$ containing n inequalities, say $Bx \leq b$, such that v is the unique solution of $Bx = b$. By Cramer's rule, each component v_i of v is given by

$$v_i = \frac{\det B_i}{\det B}$$

where we obtain B_i from B by replacing the i-th column of B by the vector b. Since B is integral and nonsingular, we have that $|\det B| \geq 1$, and so $|v_i| \leq |\det B_i|$. By (1.3.3) (c)

$$|\det B_i| \leq 2^{\langle B_i\rangle - n^2} - 1.$$

And thus, since $\langle B_i \rangle \leq \langle C, d \rangle$,

$$|\det B_i| \leq 2^{\langle C,d \rangle - n^2}.$$

Therefore, setting $R := \sqrt{n}\, 2^{\langle C,d \rangle - n^2}$ we can conclude that all vertices of P are contained in $S(0, R)$. □

The reader is invited to find better estimates for R, for instance by using the Hadamard inequality (0.1.28) directly and not only via (1.3.3).

Next we prove (3.1.28) and estimate how much the volume of the Löwner-John ellipsoid of the halfellipsoid $E'(A_k, a_k, c)$ shrinks compared with the volume of $E(A_k, a_k)$.

(3.1.34) Lemma. *Let $E \subseteq \mathbb{R}^n$ be an ellipsoid and E' be the ellipsoid obtained from E using formulas* (3.1.7), (3.1.11), (3.1.12) *for some vector $c \in \mathbb{R}^n \setminus \{0\}$. Then*

$$\frac{\text{vol}(E')}{\text{vol}(E)} = \left(\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}\right)^{1/2} < e^{-1/(2n)} < 1.$$

Proof. To estimate the volume quotient, let us assume first, that the initial ellipsoid is $F := E(I, 0)$ i. e., the unit ball around zero, and that the update vector c used to compute b in (3.1.7) is the vector $(-1, 0, \ldots, 0)^T$. In this case we obtain from (3.1.7), (3.1.11), and (3.1.12):

$$\begin{aligned} b &= (-1, 0, \ldots, 0)^T, \\ a' &= a - \frac{1}{n+1} b = \left(\frac{1}{n+1}, 0, \ldots, 0\right)^T, \\ A' &= \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}(-1, 0, \ldots, 0)^T(-1, 0, \ldots, 0)\right) \\ &= \text{diag}\left(\left(\frac{n^2}{(n+1)^2}, \frac{n^2}{n^2-1}, \ldots, \frac{n^2}{n^2-1}\right)^T\right). \end{aligned}$$

From this and (3.1.4) we conclude for the volumes of F and $F' := E(A', a')$:

$$\begin{aligned} \frac{\text{vol}(F')}{\text{vol}(F)} &= \frac{\sqrt{\det A'}\, V_n}{\sqrt{\det A}\, V_n} = \sqrt{\det A'} = \left(\frac{n^{2n}}{(n+1)^n (n-1)^n} \frac{n-1}{n+1}\right)^{1/2} \\ &= \left(\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}\right)^{1/2}. \end{aligned}$$

Hence, by taking the natural logarithm ln, we obtain:

$$\begin{aligned} \frac{\text{vol}(F')}{\text{vol}(F)} < e^{-1/(2n)} &\iff e^{1/n} < \left(\frac{n+1}{n}\right)^{n+1}\left(\frac{n-1}{n}\right)^{n-1} \\ &\iff \frac{1}{n} < (n+1)\ln\left(1 + \frac{1}{n}\right) + (n-1)\ln\left(1 - \frac{1}{n}\right). \end{aligned}$$

Using the power series expansion $\ln x = \sum_{k=1}^{\infty}(-1)^{k+1}(x-1)^k/k$ for $0 < x \leq 2$, we obtain

$$\begin{aligned}
(n+1)\ln\left(1+\frac{1}{n}\right) + (n-1)\ln\left(1-\frac{1}{n}\right) &= \\
&= \sum_{k=1}^{\infty}(-1)^{k+1}\frac{n+1}{kn^k} - \sum_{k=1}^{\infty}\frac{n-1}{kn^k} \\
&= \sum_{k=1}^{\infty}(-1)^{k+1}\frac{2}{kn^k} - \sum_{k=1}^{\infty}\frac{2(n-1)}{2kn^{2k}} \\
&= \sum_{k=1}^{\infty}\frac{2}{(2k-1)n^{2k-1}} - \sum_{k=1}^{\infty}\frac{2}{2kn^{2k}} - \sum_{k=1}^{\infty}\frac{1}{kn^{2k-1}} + \sum_{k=1}^{\infty}\frac{1}{kn^{2k}} \\
&= \sum_{k=1}^{\infty}\frac{1}{(2k-1)k}\frac{1}{n^{2k-1}} \\
&> \frac{1}{n},
\end{aligned}$$

and thus the claim is proved for our special choice of the initial ellipsoid and c.

Now let $E = E(A,a)$ be any ellipsoid and $E' = E(A',a')$ be an ellipsoid constructed from E using the formulas (3.1.7), (3.1.11), and (3.1.12) for some vector $c \in \mathbb{R}^n \setminus \{0\}$. By (3.1.3), $E = A^{1/2}E(I,0) + a = A^{1/2}F + a$. Clearly, there is an orthogonal matrix, say Q, which rotates $A^{1/2}c$ into a positive multiple of $(-1,0,\ldots,0)$, that is, there is an orthogonal matrix Q such that

$$(-1,0,\ldots,0)^T = \frac{1}{\|QA^{1/2}c\|}QA^{1/2}c.$$

Then

$$T(x) := A^{1/2}Q^Tx + a$$

is a bijective affine transformation with $T^{-1}(x) = QA^{-1/2}(x-a)$. Now

$$\begin{aligned}
T(F) &= \{T(y) \mid y^Ty \leq 1\} \\
&= \{x \mid (T^{-1}x)^T(T^{-1}x) \leq 1\} \\
&= \{x \mid (x-a)^TA^{-1/2}Q^TQA^{-1/2}(x-a) \leq 1\} \\
&= \{x \mid (x-a)^TA^{-1}(x-a) \leq 1\} \\
&= E,
\end{aligned}$$

and similarly for the ellipsoid F' defined above, we obtain $T(F') = E'$. In the subsection about properties of ellipsoids we have seen that the quotient of the volumes of two ellipsoids is invariant under bijective transformations. Thus we have

$$\frac{\mathrm{vol}(E')}{\mathrm{vol}(E)} = \frac{\mathrm{vol}(T(F'))}{\mathrm{vol}(T(F))} = \frac{\mathrm{vol}(F')}{\mathrm{vol}(F)} = \left(\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}\right)^{1/2} < e^{-1/(2n)},$$

which completes the proof. □

The following lemma proves (3.1.29) and shows the interesting fact that the volume of a full-dimensional polytope cannot be "extremely small", that is, a lower bound on $\mathrm{vol}(P)$ can be computed in polynomial time.

(3.1.35) Lemma. *If $P = \{x \in \mathbb{R}^n \mid Cx \leq d\}$ is a full-dimensional polytope and the matrix C and the vector d are integral then*

$$\mathrm{vol}(P) \geq 2^{-(n+1)\langle C\rangle + n^3}.$$

Proof. From polyhedral theory we know that every full-dimensional polytope P in $\mathbb{R}^n$ contains $n+1$ vertices $v_0, v_1, \ldots, v_n$, say, that are affinely independent. The convex hull of these vertices forms a simplex S that is contained in P. Thus, the volume of P is bounded from below by the volume of S. The volume of a simplex can be determined by a well-known formula, namely we have

$$\mathrm{vol}(S) = \frac{1}{n!}\left|\det \begin{pmatrix} 1 & 1 & \ldots & 1 \\ v_0 & v_1 & \ldots & v_n \end{pmatrix}\right|.$$

Now recall that, for every vertex v_j, its i-th component is of the form $\dfrac{\det B_{ji}}{\det B_j}$ by Cramers rule, where B_j is a submatrix of C and B_{ji} a submatrix of (C, d). So we obtain

$$\left|\det \begin{pmatrix} 1 & \ldots & 1 \\ v_0 & \ldots & v_n \end{pmatrix}\right| = \left|\frac{1}{\det B_0 \cdot \ldots \cdot \det B_n}\right| \left|\det \begin{pmatrix} \det B_0 & \ldots & \det B_n \\ u_0 & \ldots & u_n \end{pmatrix}\right|$$

where $u_0, \ldots, u_n$ are integral vectors in $\mathbb{R}^n$. Since the last determinant above is a nonzero integer, and since by (1.3.3) (c), $|\det B_j| \leq 2^{\langle C\rangle - n^2}$ holds, we get

$$\left|\det \begin{pmatrix} 1 & \ldots & 1 \\ v_0 & \ldots & v_n \end{pmatrix}\right| \geq \frac{1}{|\det B_0| \cdot \ldots \cdot |\det B_n|} \geq (2^{\langle C\rangle - n^2})^{-(n+1)}.$$

Therefore, using $n! \leq 2^{n^2}$, we obtain

$$\mathrm{vol}(P) \geq \mathrm{vol}(S) \geq \frac{1}{n!}(2^{\langle C\rangle - n^2})^{-(n+1)} \geq 2^{-(n+1)\langle C\rangle + n^3}.$$

□

The last lemma in this sequence justifies the choice (3.1.31) of the number N of iterations of the general step in the basic ellipsoid method.

(3.1.36) Lemma. *Suppose $P = \{x \in \mathbb{R}^n \mid Cx \leq d\}$ is a full-dimensional polytope and $Cx \leq d$ is an integral inequality system. If $E_0 = E(A_0, a_0)$ with $a_0 = 0$, $A_0 = R^2 I$ and $R = \sqrt{n}\, 2^{\langle C,d\rangle - n^2}$ is the initial ellipsoid, and if the general step of the basic ellipsoid method* (3.1.32) *is applied $N := 2n((2n+1)\langle C\rangle + n\langle d\rangle - n^3)$ times then*

$$\mathrm{vol}(E_N) < 2^{-(n+1)\langle C\rangle + n^3} \leq \mathrm{vol}(P).$$

Proof. Since $E_0 \subseteq \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq R\}$, we have

$$\text{vol}(E_0) \leq 2^n R^n = n^{n/2}\, 2^{n\langle C,d\rangle - n^3 + n} = 2^{n(\langle C,d\rangle - n^2 + 1 + \log(n)/2)} < 2^{n\langle C,d\rangle}.$$

By (3.1.34), in each step of the basic ellipsoid method the volume of the next ellipsoid shrinks at least with the factor $e^{-1/(2n)}$. Thus after N steps we get

$$\text{vol}(E_N) < e^{-N/(2n)}\, \text{vol}(E_0) < 2^{-N/(2n) + n\langle C,d\rangle} = 2^{-(n+1)\langle C\rangle + n^3} \leq \text{vol}(P),$$

where the last inequality follows from (3.1.35). The equality above is, in fact, the defining equality for N. This proves our claim. □

The reader will have noticed that most of the estimates in the proofs above are quite generous. By a somewhat more careful analysis one can improve all crucial parameters slightly to obtain a smaller upper bound N on the number of iterations of the ellipsoid method. Since our main point is that the upper bound is polynomial we have chosen a short way to achieve this. As far as we can see, however, the order of magnitude of N, which is $O(n^2\langle C,d\rangle)$, cannot be improved.

Implementation Problems and Polynomiality

To estimate the running time of the basic ellipsoid method (3.1.32) in the worst case, let us count each arithmetic operation as one elementary step of the algorithm. The initialization (a), (b), (c) can obviously be done in $O(m \cdot n)$ elementary steps. In each execution of the general step (d), ..., (i) of (3.1.32) we have to substitute a_k into the system $Cx \leq d$ which requires $O(m \cdot n)$ elementary steps, and we have to update a_k and A_k in (h), (i). Clearly (h) needs $O(n)$ elementary steps while (i) requires $O(n^2)$ elementary steps. The maximum number of times the general step of (3.1.32) is executed is bounded from above by $N = 2n((2n+1)\langle C\rangle + n\langle d\rangle - n^3) = O(n^2\langle C,d\rangle)$. Since P is a polytope we have $m \geq n$, and thus the total number of elementary steps of the basic ellipsoid method is $O(mn^3\langle C,d\rangle)$. So we have found a polynomial upper bound on the number of elementary steps of the basic ellipsoid method.

However, there are certain problems concerning the implementation of some of the elementary steps. We take a square root in step (c) to calculate R, so we may obtain an irrational number. One can take care of this by rounding R up to the next integer. But, in fact, we only need the rational number R^2 in the initialization of A_0, so we do not have to compute R at all. A more serious problem comes up in the general step. The vector b calculated in (g) is in general irrational, and so is a_{k+1}. Since irrational numbers have no finite representation in binary encoding, we are not able to calculate the center a_{k+1} exactly. Thus, in any implementation of the ellipsoid method we have to round the coefficients of the entries of the center of the next ellipsoid. This causes difficulties in our analysis. Namely, by rounding a_{k+1} to some vector $\tilde{a}_{k+1} \in \mathbb{Q}^n$, say, we will translate the ellipsoid E_{k+1} slightly to the ellipsoid $\tilde{E}_{k+1} := E(A_{k+1}, \tilde{a}_{k+1})$. Recalling that E_{k+1} is the ellipsoid of minimum volume containing the halfellipsoid $E'(A_k, a_k, c)$ –

cf. (3.1.27) – we see that $\tilde{E}_{k+1}$ does not contain $E'(A_k, a_k, c)$ any more. So it may happen that the polytope $P \subseteq E'(A_k, a_k, c)$ is not contained in $\tilde{E}_{k+1}$ either. And therefore, our central proof idea of constructing a sequence of shrinking ellipsoids each containing P is not valid any more.

There is a further issue. Observe that the new matrix A_{k+1} calculated in (i) is rational, provided A_k and c are. But it is not clear a priori that the entries of A_{k+1} do not grow too fast.

All these difficulties can be overcome with some effort. We will describe here the geometric ideas behind this. The concrete mathematical analysis of the correctness and implementability of these ideas will be given in Section 3.2.

We have seen that rounding is unavoidable in the calculation of the center a_{k+1}. We will do this as follows. We write each entry of a_{k+1} in its binary representation and cut off after p digits behind the binary point, where p is to be specified. In other words, we fix a denominator 2^p and approximate each number by a rational number with this denominator. To keep the growth of the encoding lengths of the matrices A_{k+1} under control, we apply the same rounding method to the entries of A_{k+1}. As remarked before, rounding of the entries of the center results in a translation of the ellipsoid. By rounding the entries of A_{k+1} we induce a change of the shape of the ellipsoid in addition. The two roundings may produce a new ellipsoid which does not contain P. Moreover, by rounding a positive definite matrix too roughly positive definiteness may be lost. So we also have to take care that the rounded matrix is still positive definite. It follows that we should make p large enough, subject to the condition that all numbers coming up during the execution of the algorithm are of polynomial encoding length.

We still have to find a way to keep P inside the newly calculated ellipsoid. Since we cannot move P, we have to do something with the new ellipsoid. One idea that works is to maintain the (rounded) center and to blow up the ellipsoid obtained by rounding the entries of A_{k+1} in such a way that the blow-up will compensate the translation and the change of shape of the Löwner-John ellipsoid induced by rounding. The blow-up factor must be so large that the enlarged ellipsoid contains P. Clearly, we have to blow up carefully. We know from (3.1.28) that the shrinking rate is below 1; but it is very close to 1. In order to keep polynomial time termination, we have to choose a blow-up factor that on the one hand gives a sufficient shrinking rate and on the other, guarantees that P is contained in the blown-up ellipsoid. This shows that the blow-up factor and the number p determining the precision of the arithmetic influence each other, and they have to be chosen simultaneously to achieve all the goals at the same time.

It turns out that appropriate choices of N, p, and the blow-up factor ξ can be made, namely if we choose

$$
\begin{aligned}
N &:= 50(n+1)^2\langle C, d\rangle, \\
p &:= 8N, \\
\xi &:= 1 + \frac{1}{4(n+1)^2},
\end{aligned}
\tag{3.1.37}
$$

we can make all the desired conclusions. These modifications turn the basic ellipsoid method (3.1.32) into an algorithm that runs in polynomial time.

From the practical point of view these considerations and modifications seem quite irrelevant. Note that the ellipsoid method requires problem-specific precision. But usually, our computers have fixed precision. Even software that allows the use of variable precision would not help, since the precision demands of the ellipsoid method in this version are so gigantic that they are hardly satisfiable in practice. Thus, in a computer implementation of the ellipsoid method with fixed precision it will be possible to conclude that one of the calculated centers is contained in P, but by stopping after N steps we cannot surely declare P empty. To prove emptiness, additional tests have to be added, based for instance on the Farkas lemma.

By looking at the formulas we have stated for Löwner-John ellipsoids of various ellipsoidal sections – see (3.1.11), (3.1.12); (3.1.16), (3.1.17); (3.1.19), (3.1.20) – one can immediately see that a speed-up of the shrinking can be obtained by using deep or other cuts. There are many possibilities to "play" with the parameters of the ellipsoid method. To describe some of them let us write the basic iteration of the ellipsoid method in the following form

$$a_{k+1} :\approx a_k - \rho \frac{1}{\sqrt{c^T A_k c}} A_k c, \tag{3.1.38}$$

$$A_{k+1} :\approx \xi \cdot \sigma\left(A_k - \tau \frac{1}{c^T A_k c} A_k c c^T A_k\right), \tag{3.1.39}$$

where "$\approx$" means cutting off after the p-th digit behind the point in the binary representation of the number on the right hand side. Following BLAND, GOLDFARB and TODD (1981) we call ρ the **step parameter** (it determines the length of the step from a_k in the direction of $-b$ to obtain a_{k+1}), σ the **dilatation parameter**, τ the **expansion parameter** and ξ the **blow-up parameter** (this is the factor used to blow up the ellipsoid to compensate for the rounding errors). The ellipsoid method in perfect arithmetic (no rounding) as stated in (3.1.32) is thus given by

$$\rho := \frac{1}{n+1}, \quad \sigma := \frac{n^2}{n^2-1}, \quad \tau := \frac{2}{n+1}, \quad \xi := 1. \tag{3.1.40}$$

It is a so-called central-cut method since it always uses cuts through the center of the current ellipsoid. The ellipsoid method with rounding and blow-up, i. e., the polynomial time version of (3.1.32) is thus defined by

$$\rho := \frac{1}{n+1}, \quad \sigma := \frac{n^2}{n^2-1}, \quad \tau := \frac{2}{n+1}, \quad \xi := 1 + \frac{1}{4(n+1)^2}. \tag{3.1.41}$$

If $c^T x \leq \gamma$ is the violated inequality found in (f) of (3.1.32), then defining $\alpha := (c^T a_k - \gamma)/\sqrt{c^T A_k c}$ as in (3.1.13) and setting

$$\rho := \frac{1+n\alpha}{n+1}, \quad \sigma := \frac{n^2(1-\alpha^2)}{n^2-1}, \quad \tau := \frac{2(1+n\alpha)}{(n+1)(1+\alpha)}, \quad \xi := 1 \tag{3.1.42}$$

we obtain the **deep-cut ellipsoid method** (in perfect arithmetic). Note that since $c^T a_k > \gamma$, the hyperplane $\{x \mid c^T x = \gamma\}$ is indeed a deep cut, and so the Löwner-John ellipsoid of $E'(A_k, a_k, c, \gamma)$ – see (3.1.15) – which is given through formulas (3.1.38), (3.1.39) (without rounding), and (3.1.42) has indeed a smaller volume than that of $E'(A_k, a_k, c)$ used in (3.1.32).

The use of deep cuts will – due to faster shrinking – in general speed up the convergence of the ellipsoid method. But one can also easily construct examples where the standard method finds a feasible solution more quickly. However, the use of deep cuts does not seem to change the order of magnitude of the number N of iterations of the general step necessary to correctly conclude that P is empty. So from a theoretical point of view, deep cuts do not improve the worst-case running time of the algorithm.

Using a number α in (3.1.42) with $-1/n < \alpha < 0$ we obtain a **shallow-cut ellipsoid method**. Although – from a practical point of view – it seems ridiculous to use a shallow-cut method, since we do not shrink as much as we can, it will turn out that this version of the ellipsoid method is of particular theoretical power. Using this method we will be able to prove results which – as far as we can see – cannot be derived from the central-cut or deep-cut ellipsoid method.

Some Examples

The following examples have been made up to illustrate the iterations of the ellipsoid method geometrically. Let us first consider the polytope $P \subseteq \mathbb{R}^2$ defined by the inequalities

$$\begin{array}{lrcl} (1) & -x_1 - & x_2 \leq & -2 \\ (2) & 3x_2 & \leq & 4 \\ (3) & -2x_1 + 2x_2 \leq & & 3. \end{array}$$

This polytope is contained in the ball of radius 7 around zero. To find a point in P, we start the basic ellipsoid method (3.1.32) with $E(A_0, a_0)$ where

$$A_0 = \begin{pmatrix} 49 & 0 \\ 0 & 49 \end{pmatrix}, \quad a_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

see Figure 3.5. The center a_0 violates inequality (1). One iteration of the algorithm yields the ellipsoid $E(A_1, a_1)$ also shown in Figure 3.5. The new center a_1 violates (2). We update and continue this way. The fifth iteration produces the ellipsoid $E(A_5, a_5)$ displayed in Figure 3.6. In the 7-th iteration the ellipsoid $E(A_7, a_7)$ shown in Figure 3.7 is found. Its center $a_7 = (1.2661, 2.3217)^T$ is contained in P. The whole sequence of ellipsoids $E(A_0, a_0), \ldots, E(A_7, a_7)$ produced in this example is shown in Figure 3.8.

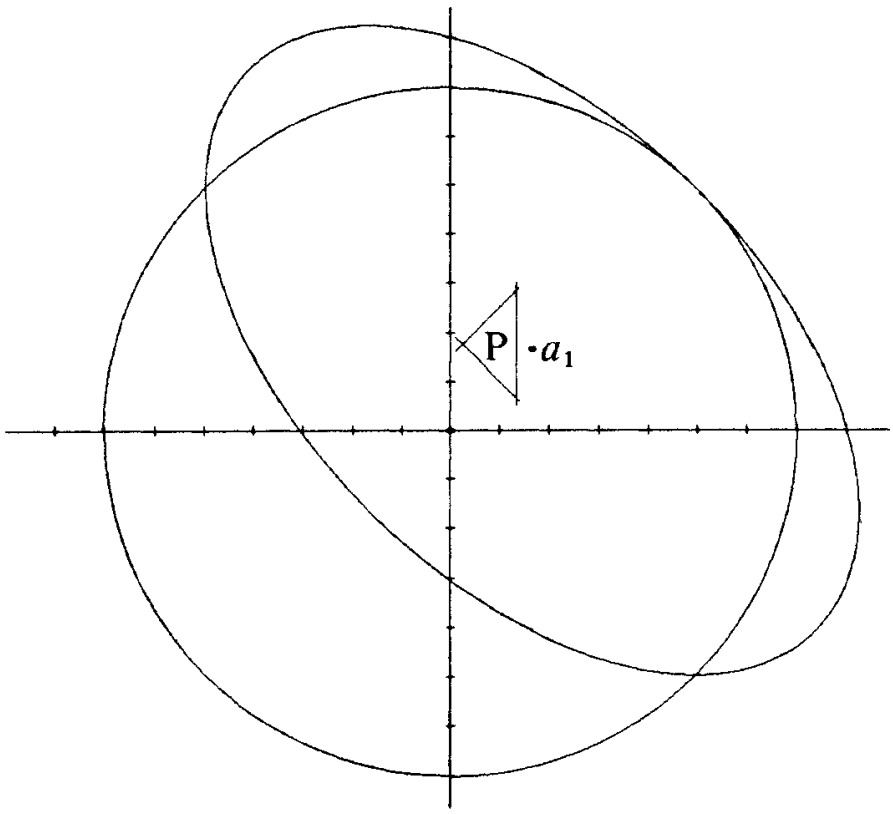


Figure 3.5

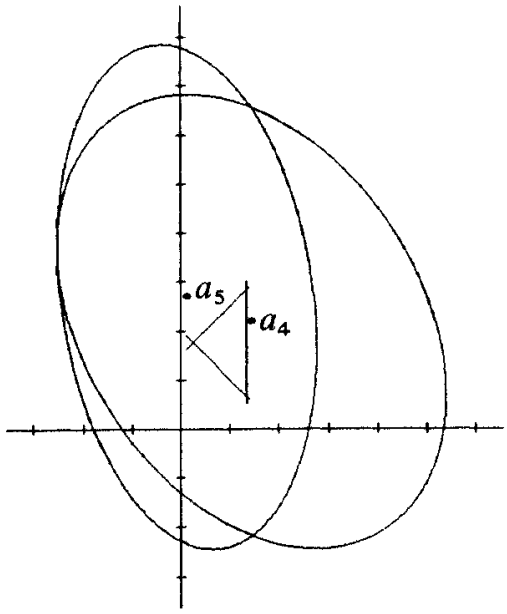


Figure 3.6

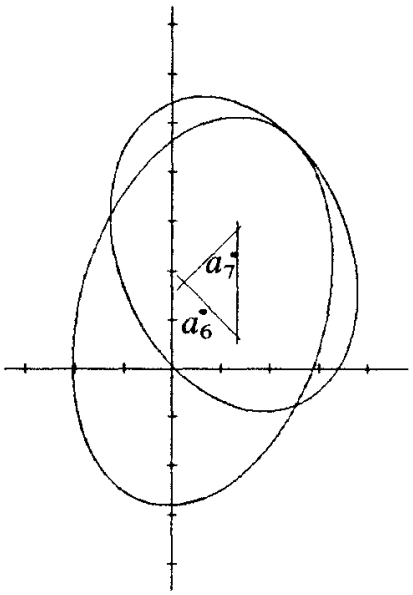


Figure 3.7

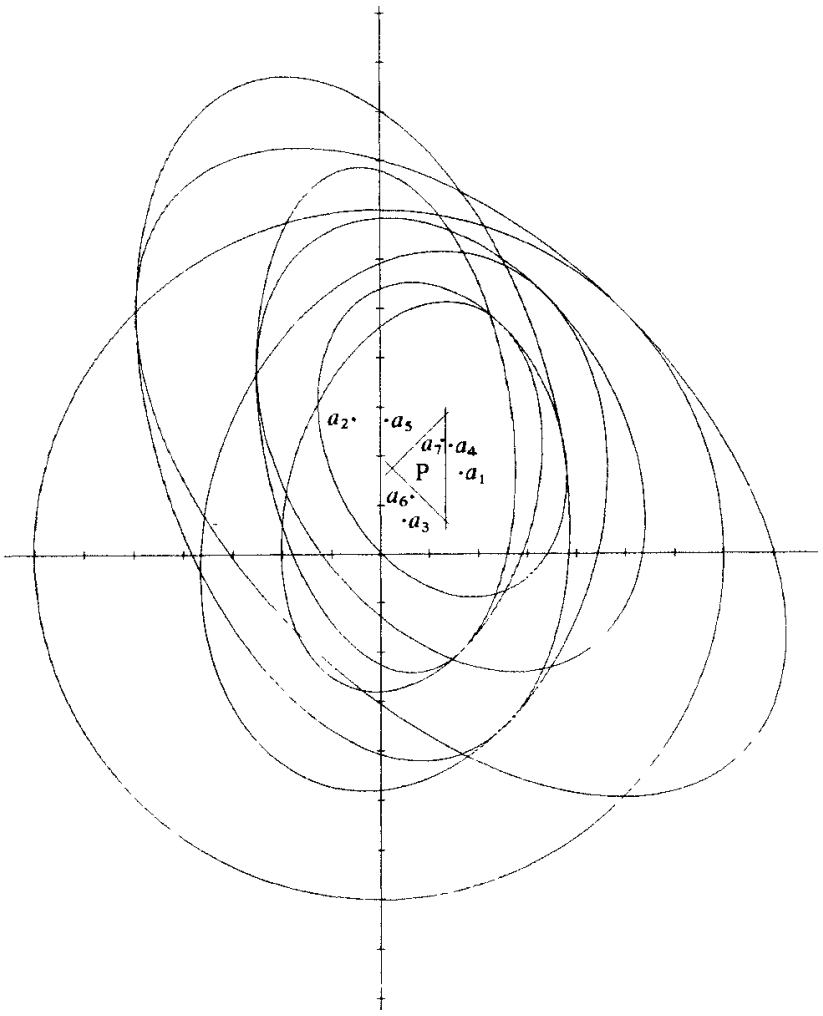
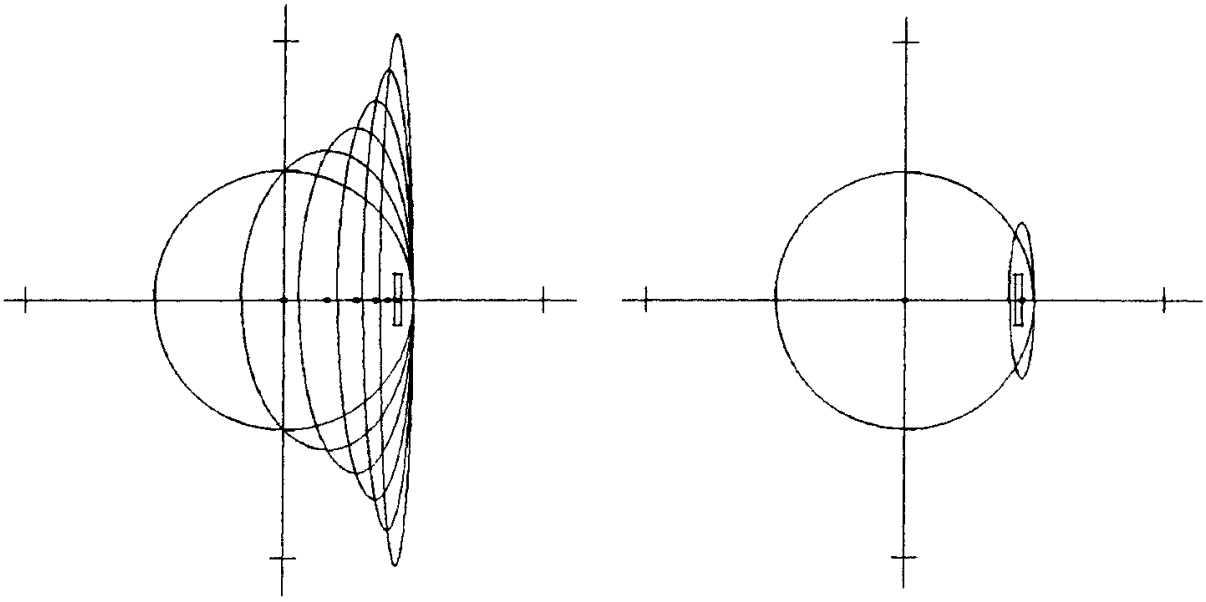


Figure 3.8

The pictures of the iterations of the ellipsoid method show clearly that in an update the ellipsoid is squeezed in the direction of c while it is expanded in the direction orthogonal to c. If many updates are done with one and the same vector c the ellipsoids become "needles". To demonstrate this effect, let us consider the polytope $P \subseteq \mathbb{R}^2$ defined by

$$\frac{17}{20} \leq x_1 \leq \frac{18}{20}$$
$$-\frac{1}{5} \leq x_2 \leq \frac{1}{5}.$$

Starting the basic ellipsoid method with $E(A_0, a_0) = S(0, 1)$, we make five iterations until the center $a_5 = (211/243, 0)^T$ of the fifth ellipsoid $E(A_5, a_5)$ is contained in P. The six ellipsoids $E(A_0, a_0), \ldots, E(A_5, a_5)$ of this sequence are displayed in Figure 3.9 (a). If the ellipsoid method receives "flat" polytopes as its input, it is likely that the algorithm produces extremely flat ellipsoids, say of several kilometers length and a few millimeters width. This inevitably leads to numerical difficulties in practice. In such cases, some variants of the ellipsoid method frequently perform better empirically. For instance, the deep-cut ellipsoid method finds a feasible point of the polytope P defined above in one iteration. This iteration is shown in Figure 3.9 (b). But there is no general result quantifying this improvement in performance.



(a) (b)

Figure 3.9

*3.2 The Central-Cut Ellipsoid Method

We shall now give a description and analysis of the version of the basic ellipsoid method (3.1.32) where arithmetic operations are performed in finite precision and the errors induced by rounding are compensated for by blowing up the

"rounded" Löwner-John ellipsoid. This section provides proofs of the claims about this method made in the previous section. In particular, the polynomial running time is established. Moreover, we do not restrict ourselves to the case of an explicitly given polytope, but we treat the general case of a circumscribed closed convex set given by a certain separation oracle. Our main result of this section can be stated as follows.

(3.2.1) Theorem. *There exists an oracle-polynomial time algorithm, called the* **central-cut ellipsoid method**, *that solves the following problem:*

Input: *A rational number* $\varepsilon > 0$ *and a circumscribed closed convex set* $(K; n, R)$ *given by an oracle* SEP_K *that, for any* $y \in \mathbb{Q}^n$ *and any rational number* $\delta > 0$, *either asserts that* $y \in S(K, \delta)$ *or finds a vector* $c \in \mathbb{Q}^n$ *with* $\|c\|_\infty = 1$ *such that* $c^T x \leq c^T y + \delta$ *for every* $x \in K$.

Output: *One of the following:*

(i) *a vector* $a \in S(K, \varepsilon)$,

(ii) *a positive definite matrix* $A \in \mathbb{Q}^{n \times n}$ *and a point* $a \in \mathbb{Q}^n$ *such that* $K \subseteq E(A, a)$ *and* $\mathrm{vol}(E(A, a)) \leq \varepsilon$.

Whenever we speak of the ellipsoid method without referring to a special version we will mean the central-cut ellipsoid method, to be described in the proof of Theorem (3.2.1). This method specifies a sequence of ellipsoids $E_0, E_1, \ldots, E_N$ (i. e., a sequence of positive definite matrices $A_0, A_1, \ldots, A_N$, and a sequence of centers $a_0, a_1, \ldots, a_N$, with $E_k = E(A_k, a_k)$, $k = 0, \ldots, N$), that contain the given set K, such that either at least one of the centers a_k satisfies $a_k \in S(K, \varepsilon)$ (so alternative (i) of (3.2.1) is achieved), or the last ellipsoid E_N has volume at most ε.

Proof of Theorem (3.2.1). The proof will be given in several steps. We first describe the method, then prove its correctness assuming that several lemmas hold. Finally the truth of the lemmas will be established.

So let numbers n, R, ε and an oracle SEP_K be given as required in the theorem. Without loss of generality $\varepsilon < 1$.

I. For the precise description of the algorithm we need the following parameters:

$$N := \lceil 5n|\log \varepsilon| + 5n^2|\log(2R)| \rceil, \tag{3.2.2}$$

$$p := 8N, \tag{3.2.3}$$

$$\delta := 2^{-p}. \tag{3.2.4}$$

The integer N is the maximum number of iterations of the central-cut ellipsoid method, the rational number δ is the error we allow the oracle SEP_K to make, and the integer p is the precision parameter for representation of numbers. We assume throughout the algorithm that all numbers occurring are represented in binary form and are rounded to p digits behind the point. Clearly, for every rational number given by a numerator and denominator, this binary approximation can be easily computed.

We initialize the procedure by setting:

$$a_0 := 0, \tag{3.2.5}$$
$$A_0 := R^2 I,$$

so that $E_0 = S(0, R)$ and hence $K \subseteq E_0$.

Assume a_k, A_k are found for some $k \geq 0$. If $k = N$, then the ellipsoid $E_N = E(A_N, a_N)$ has the property required in (ii) of (3.2.1) (we shall prove this later), and we stop.

If $k < N$, we call the oracle SEP_K with $y = a_k$ and the error parameter δ defined in (3.2.4).

If SEP_K concludes that $a_k \in S(K, \delta)$, then by the choice of δ, $a_k \in S(K, \varepsilon)$, and we stop having achieved (i) of (3.2.1).

If SEP_K gives a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ and $c^T x \leq c^T a_k + \delta$ for all $x \in K$, then we do the following computations:

$$a_{k+1} :\approx a_k - \frac{1}{n+1} \frac{A_k c}{\sqrt{c^T A_k c}}, \tag{3.2.6}$$

$$A_{k+1} :\approx \frac{2n^2+3}{2n^2} \left(A_k - \frac{2}{n+1} \frac{A_k c c^T A_k}{c^T A_k c} \right), \tag{3.2.7}$$

where the sign "$\approx$" in (3.2.6), (3.2.7) means that the left hand side is obtained by cutting the binary expansions of the numbers on right hand side after p digits behind the binary point. This finishes the description of the central-cut ellipsoid method.

II. To prove the correctness of the algorithm, we establish the following facts. (Recall that for a vector x, $\|x\|$ denotes the Euclidean norm, and for a matrix A, $\|A\|$ denotes the spectral norm (0.1.16).)

(3.2.8) Lemma. *The matrices $A_0, A_1, \ldots$ are positive definite. Moreover,*

$$\|a_k\| \leq R2^k, \quad \|A_k\| \leq R^2 2^k, \quad \text{and} \quad \|A_k^{-1}\| \leq R^{-2} 4^k.$$

(3.2.9) Lemma. *$K \subseteq E_k$ for $k = 0, 1, \ldots$*

(3.2.10) Lemma. *$\mathrm{vol}(E_{k+1})/\mathrm{vol}(E_k) \leq e^{-1/(5n)}$ for $k = 0, 1, \ldots$*

We shall prove these lemmas in III, IV, and V below.

It follows from Lemma (3.2.8) that all the formulas on the right hand sides of (3.2.6), (3.2.7) are meaningful (no division by 0) and that the intermediate numbers (entries of a_k and A_k) do not grow too large, i. e., have polynomial encoding lengths. This shows that all the arithmetic operations can be carried out in polynomial time.

If the algorithm stops with $a_k \in S(K, \varepsilon)$, then of course we have nothing to prove. If it stops with $k = N$, then by Lemma (3.2.9), E_N indeed contains K. Moreover, by Lemma (3.2.10) the volume of E_N satisfies

$$\mathrm{vol}(E_N) \leq e^{-N/(5n)} \mathrm{vol}(E_0).$$

E_0 is the ball around zero with radius R, so E_0 is contained in the hypercube $Q = \{x \in \mathbb{R}^n \mid -R \leq x_i \leq R,\ i = 1, \ldots, n\}$. From the very rough estimate $\text{vol}(E_0) \leq \text{vol}(Q) = (2R)^n$ we get

$$\text{vol}(E_N) \leq e^{-N/(5n)}(2R)^n < 2^{-N/(5n)}(2R)^n \leq \varepsilon. \tag{3.2.11}$$

The last inequality in (3.2.11) in fact is the inequality from which the value of N is derived. So the truth of this inequality directly follows from the choice of N. This finishes the proof of Theorem (3.2.1) subject to the correctness of (3.2.8), (3.2.9), (3.2.10), which we show now. □

III. **Proof of Lemma** (3.2.8). We prove the lemma by induction on k. Since $a_0 = 0$ and since the largest eigenvalue of A_0 is R^2, all the statements of (3.2.8) are clearly true for $k = 0$. Assume that they are true for $k \geq 0$. Let a_{k+1}^*, A_{k+1}^* be the right hand sides of (3.2.6) and (3.2.7) without rounding, i. e.,

$$a_{k+1}^* := a_k - \frac{1}{n+1}\frac{A_k c}{\sqrt{c^T A_k c}}, \tag{3.2.12}$$

$$A_{k+1}^* := \frac{2n^2+3}{2n^2}\left(A_k - \frac{2}{n+1}\frac{A_k c c^T A_k}{c^T A_k c}\right). \tag{3.2.13}$$

Then note first that

$$(A_{k+1}^*)^{-1} = \frac{2n^2}{2n^2+3}\left(A_k^{-1} + \frac{2}{n-1}\frac{cc^T}{c^T A_k c}\right), \tag{3.2.14}$$

which is easy to verify by computation. Thus, $(A_{k+1}^*)^{-1}$ is the sum of a positive definite and a positive semidefinite matrix, and so it is positive definite. Hence also A_{k+1}^* is positive definite.

Equation (0.1.17) immediately implies that for positive semidefinite matrices A and B, $\|A\| \leq \|A+B\|$ holds. Using this, the fact that A_{k+1}^* is positive definite, and the induction hypothesis, we have:

$$\begin{aligned}\|A_{k+1}^*\| &= \frac{2n^2+3}{2n^2}\left\|A_k - \frac{2}{n+1}\frac{A_k c c^T A_k}{c^T A_k c}\right\| \\ &\leq \frac{2n^2+3}{2n^2}\|A_k\| \leq \frac{11}{8}R^2 2^k.\end{aligned} \tag{3.2.15}$$

Further, since each entry of A_{k+1} differs from the corresponding entry of A_{k+1}^* by at most 2^{-p} (due to our rounding procedure) we have from (0.1.23)

$$\|A_{k+1} - A_{k+1}^*\| \leq \|A_{k+1} - A_{k+1}^*\|_{\max} \leq n2^{-p}. \tag{3.2.16}$$

So (3.2.15), (3.2.16), and the choice of p give

$$\|A_{k+1}\| \leq \|A_{k+1} - A_{k+1}^*\| + \|A_{k+1}^*\| \leq n2^{-p} + \frac{11}{8}R^2 2^k \leq R^2 2^{k+1}. \tag{3.2.17}$$

This proves the second claim of Lemma (3.2.8).

Moreover, setting $Q := A_k^{1/2}$, using formula (3.2.12) and the definition of the spectral norm (0.1.16) we obtain

$$\|a_{k+1}^* - a_k\| = \frac{1}{n+1}\frac{\|A_k c\|}{\sqrt{c^T A_k c}} = \frac{1}{n+1}\sqrt{\frac{c^T A_k^2 c}{c^T A_k c}} = \frac{1}{n+1}\sqrt{\frac{(c^T Q^T)A_k(Qc)}{c^T Q^T Q c}} \tag{3.2.18}$$

$$\leq \frac{1}{n+1}\sqrt{\|A_k\|} \leq \frac{1}{n+1} R 2^{k-1}.$$

Our rounding procedure and (0.1.7) give

$$\|a_{k+1} - a_{k+1}^*\| \leq \sqrt{n}\|a_{k+1} - a_{k+1}^*\|_\infty \leq \sqrt{n}2^{-p}; \tag{3.2.19}$$

and therefore, by the induction hypothesis, the choice of p, and the inequalities (3.2.18) and (3.2.19) derived above we get

$$\begin{aligned} \|a_{k+1}\| &\leq \|a_{k+1} - a_{k+1}^*\| + \|a_{k+1}^* - a_k\| + \|a_k\| \\ &\leq \sqrt{n}2^{-p} + \frac{1}{n+1} R 2^{k-1} + R 2^k \\ &\leq R 2^{k+1}. \end{aligned} \tag{3.2.20}$$

This proves the first claim of (3.2.8). Finally, we observe that

$$\begin{aligned} \|(A_{k+1}^*)^{-1}\| &\leq \frac{2n^2}{2n^2+3}\left(\|A_k^{-1}\| + \frac{2}{n-1}\frac{\|cc^T\|}{c^T A_k c}\right) \\ &\leq \frac{2n^2}{2n^2+3}\left(\|A_k^{-1}\| + \frac{2}{n-1}\|A_k^{-1}\|\right) \\ &< \frac{n+1}{n-1}\|A_k^{-1}\| \\ &\leq 3R^{-2}4^k. \end{aligned} \tag{3.2.21}$$

(The first inequality above follows from (3.2.14). To get the second inequality, note that (0.1.16) immediately yields $\|cc^T\| = c^T c$. Setting $Q = A_k^{1/2}$, by (0.1.18) we have

$$\frac{\|cc^T\|}{c^T A_k c} = \frac{c^T c}{c^T Q^2 c} \leq \|Q^{-1}\|^2 = \|A_k^{-1}\|.$$

The last inequality in (3.2.21) follows from $n \geq 2$ and our induction hypothesis.)

Let λ_0 denote the least eigenvalue of A_{k+1} and let v be a corresponding eigenvector with $\|v\| = 1$. Then by (3.2.16), (3.2.21), and (0.1.17), (0.1.23), and the choice of p:

$$\begin{aligned} \lambda_0 = v^T A_{k+1} v &= v^T A_{k+1}^* v + v^T (A_{k+1} - A_{k+1}^*) v \\ &\geq \|(A_{k+1}^*)^{-1}\|^{-1} - \|A_{k+1} - A_{k+1}^*\| \\ &> \frac{1}{3} R^2 4^{-k} - n2^{-p} \\ &\geq R^2 4^{-(k+1)}. \end{aligned} \tag{3.2.22}$$

From $\lambda_0 > 0$ we can conclude that A_{k+1} is positive definite. Moreover, by (0.1.18) and (3.2.22),

$$\|A_{k+1}^{-1}\| = \lambda_0^{-1} \leq R^{-2}4^{k+1}.$$

This proves the third assertion of (3.2.8). □

IV. **Proof of Lemma** (3.2.9). By induction on k. The claim holds by construction for $k = 0$. Suppose the claim is true for k, and let a^*_{k+1} and A^*_{k+1} be the vector resp. matrix defined in (3.2.12) and (3.2.13) in the proof of the previous lemma. Take any $x \in K$. We have to prove that

$$(x - a_{k+1})^T A_{k+1}^{-1}(x - a_{k+1}) \leq 1\,. \tag{3.2.23}$$

We shall estimate the left-hand side of (3.2.23) in several steps. Using (3.2.14) we get

$$\begin{aligned}(x - a^*_{k+1})^T (A^*_{k+1})^{-1}(x - a^*_{k+1}) &= \\ &= \frac{2n^2}{2n^2+3}\Big(x - a_k + \frac{1}{n+1}\frac{A_k c}{\sqrt{c^T A_k c}}\Big)^T \\ &\qquad \Big(A_k^{-1} + \frac{2}{n-1}\frac{cc^T}{c^T A_k c}\Big)\Big(x - a_k + \frac{1}{n+1}\frac{A_k c}{\sqrt{c^T A_k c}}\Big) \\ &= \frac{2n^2}{2n^2+3}\Big((x-a_k)^T A_k^{-1}(x-a_k) + \\ &\qquad + \frac{1}{n^2-1} + \frac{2}{n-1}\frac{c^T(x-a_k)}{\sqrt{c^T A_k c}} + \frac{2}{n-1}\frac{(c^T(x-a_k))^2}{c^T A_k c}\Big)\,.\end{aligned} \tag{3.2.24}$$

By induction hypothesis, we know that $K \subseteq E_k$. So x belongs to E_k, and thus the first term in the last formula of (3.2.24) above is at most 1. Setting

$$t := \frac{c^T(x - a_k)}{\sqrt{c^T A_k c}}, \tag{3.2.25}$$

we therefore obtain

$$(x - a^*_{k+1})^T (A^*_{k+1})^{-1}(x - a^*_{k+1}) \leq \frac{2n^2}{2n^2+3}\Big(\frac{n^2}{n^2-1} + \frac{2}{n-1}t(t+1)\Big)\,. \tag{3.2.26}$$

To estimate $t(t+1)$, we proceed as follows. Using the fact that A_k can be written as QQ, where $Q = A_k^{1/2}$, we can bound t from above employing the Cauchy-Schwarz inequality (0.1.26):

$$\begin{aligned}|c^T(x - a_k)| &= |c^T Q(Q^{-1}(x - a_k))| \\ &\leq \|c^T Q\|\,\|Q^{-1}(x - a_k)\| \\ &= \sqrt{c^T QQc}\,\sqrt{(x-a_k)^T Q^{-1}Q^{-1}(x-a_k)} \\ &= \sqrt{c^T A_k c}\,\sqrt{(x-a_k)^T A_k^{-1}(x-a_k)}\,,\end{aligned}$$

and hence, since $x \in E_k$

$$|t| = \Big|\frac{c^T(x-a_k)}{\sqrt{c^T A_k c}}\Big| \leq \sqrt{(x-a_k)^T A_k^{-1}(x-a_k)} \leq 1. \tag{3.2.27}$$

The oracle SEP_K guarantees that $c^T(x-a_k) \le \delta$, and so

$$t = \frac{c^T(x-a_k)}{\sqrt{c^T A_k c}} \le \frac{\delta}{\|c\|\sqrt{\|A_k^{-1}\|^{-1}}} \le \delta\sqrt{\|A_k^{-1}\|} \le \delta R^{-1}2^k.$$

Using this estimate and (3.2.27), we can conclude

$$t(t+1) \le 2\delta R^{-1}2^N.$$

Substituting this upper bound for $t(t+1)$ in (3.2.26), we get

$$\begin{aligned}(3.2.28)\quad (x-a_{k+1}^*)^T(A_{k+1}^*)^{-1}(x-a_{k+1}^*) &\le \frac{2n^2}{2n^2+3}\left(\frac{n^2}{n^2-1} + 4\delta R^{-1}2^N\right)\\ &\le \frac{2n^4}{2n^4+n^2-3} + 4\delta R^{-1}2^N.\end{aligned}$$

The rest of the proof is standard error estimation:

$$\begin{aligned}\Delta &:= |(x-a_{k+1})^T A_{k+1}^{-1}(x-a_{k+1}) - (x-a_{k+1}^*)^T(A_{k+1}^*)^{-1}(x-a_{k+1}^*)|\\ &\le |(x-a_{k+1})^T A_{k+1}^{-1}(a_{k+1}^*-a_{k+1})| + |(a_{k+1}^*-a_{k+1})A_{k+1}^{-1}(x-a_{k+1}^*)|\\ &\quad + |(x-a_{k+1}^*)^T(A_{k+1}^{-1}-(A_{k+1}^*)^{-1})(x-a_{k+1}^*)|\\ &\le \|x-a_{k+1}\|\,\|A_{k+1}^{-1}\|\,\|a_{k+1}^*-a_{k+1}\| + \|a_{k+1}^*-a_{k+1}\|\,\|A_{k+1}^{-1}\|\,\|x-a_{k+1}^*\|\\ &\quad + \|x-a_{k+1}^*\|^2\|A_{k+1}^{-1}\|\,\|(A_{k+1}^*)^{-1}\|\,\|A_{k+1}^*-A_{k+1}\|.\end{aligned}$$

To continue the estimation, we observe that by Lemma (3.2.8), by the choice of p, and by the facts that $x \in S(0,R)$ and $k \le N-1$:

$$\|x-a_{k+1}\| \le \|x\| + \|a_{k+1}\| \le R + R2^{k+1} \le R2^{N+1},$$

$$\|x-a_{k+1}^*\| \le \|x-a_{k+1}\| + \|a_{k+1}-a_{k+1}^*\| \le R + R2^{k+1} + \sqrt{n}\,2^{-p} \le R2^{N+1}.$$

So again by (3.2.8) and by (3.2.21), we conclude:

$$\begin{aligned}(3.2.29)\quad \Delta &\le (R2^{N+1})(R^{-2}4^N)(\sqrt{n}\,2^{-p}) + (\sqrt{n}\,2^{-p})(R^{-2}4^N)(R2^{N+1})\\ &\quad + (R^2 2^{2N+2})(R^{-2}4^N)(R^{-2}4^N)(n2^{-p})\\ &\le nR^{-1}2^{3N+2-p} + nR^{-2}2^{6N+2-p}.\end{aligned}$$

Now we can put the estimates (3.2.28) and (3.2.29) together to get (3.2.23):

$$\begin{aligned}(x-a_{k+1})^T\, A_{k+1}^{-1}(x-a_{k+1}) &\le\\ &\le |(x-a_{k+1})^T A_{k+1}^{-1}(x-a_{k+1}) - (x-a_{k+1}^*)(A_{k+1}^*)^{-1}(x-a_{k+1}^*)|\\ &\quad + (x-a_{k+1}^*)^T(A_{k+1}^*)^{-1}(x-a_{k+1}^*)\\ &\le \frac{2n^4}{2n^4+n^2-3} + 4\delta R^{-1}2^N + nR^{-1}2^{3N+2-p} + nR^{-2}2^{6N+2-p}\\ &\le 1,\end{aligned}$$

where the last inequality follows from the choice of p. □

V. **Proof of Lemma** (3.2.10). As mentioned in (3.1.4), the volume of an ellipsoid $E(A, a) \subseteq \mathbb{R}^n$ is known to be $\sqrt{\det(A)}\ \mathrm{vol}(S(0,1))$. Hence we obtain

$$\frac{\mathrm{vol}(E_{k+1})}{\mathrm{vol}(E_k)} = \sqrt{\frac{\det(A_{k+1})}{\det(A_k)}} = \sqrt{\frac{\det(A^*_{k+1})}{\det(A_k)}}\ \sqrt{\frac{\det(A_{k+1})}{\det(A^*_{k+1})}}, \tag{3.2.30}$$

where A^*_{k+1} is the matrix defined in (3.2.13). To estimate the first factor on the right hand side of (3.2.30), write $A_k = QQ$ where $Q = A_k^{1/2}$ and use the definition of A^*_{k+1}:

$$\frac{\det(A^*_{k+1})}{\det(A_k)} = \det(Q^{-1}A^*_{k+1}Q^{-1}) = \left(\frac{2n^2+3}{2n^2}\right)^n \det\left(I - \frac{2}{n+1}\frac{Qcc^TQ}{c^TQQc}\right).$$

Since Qcc^TQ/c^TQQc has rank one and trace one, the matrix in the last determinant has eigenvalues $1, \ldots, 1, 1-2/(n+1)$. So

$$\frac{\det(A^*_{k+1})}{\det(A_k)} = \left(\frac{2n^2+3}{2n^2}\right)^n \frac{n-1}{n+1} \leq e^{3/(2n)}e^{-2/n} = e^{-1/(2n)}. \tag{3.2.31}$$

To obtain the last estimate we have used the well-known facts that $1 + x \leq e^x$ for all $x \in \mathbb{R}$ and $(1 + \frac{2}{n-1})^n > e^2$, for $n \geq 2$ – cf. PÓLYA and SZEGÖ (1978), I. 172.

To estimate the second factor of (3.2.30) write it as follows (and recall inequality (3.2.21) and the fact that $\det B \leq \|B\|^n$):

$$\begin{aligned}
\frac{\det A_{k+1}}{\det A^*_{k+1}} &= \det(I + (A^*_{k+1})^{-1}(A_{k+1} - A^*_{k+1})) \\
&\leq \|I + (A^*_{k+1})^{-1}(A_{k+1} - A^*_{k+1})\|^n \\
&\leq (\|I\| + \|(A^*_{k+1})^{-1}\|\,\|A_{k+1} - A^*_{k+1}\|)^n \\
&\leq (1 + (R^{-2}4^{k+1})(n2^{-p}))^n \\
&\leq e^{n^2 2^{2N-p}R^{-2}} \\
&\leq e^{1/(10n)},
\end{aligned} \tag{3.2.32}$$

where the last inequality follows from the choice of N and p. Thus, from the estimates (3.2.31) and (3.2.32) we get

$$\frac{\mathrm{vol}(E_{k+1})}{\mathrm{vol}(E_k)} \leq \sqrt{\frac{\det A^*_{k+1}}{\det A_k}}\ \sqrt{\frac{\det A_{k+1}}{\det A^*_{k+1}}} \leq e^{-1/(4n)+1/(20n)} = e^{-1/(5n)}.$$

This finishes the proof of Lemma (3.2.10). □

Thus the proof of Theorem (3.2.1) is complete.

(3.2.33) Remark. Suppose that, instead of the oracle SEP_K in the input of Theorem (3.2.1), we have an oracle SEP_{K,K_1} (for some fixed subset $K_1 \subseteq K$), which is weaker than SEP_K in the following sense: for any $y \in \mathbb{Q}^n$ and any rational $\delta > 0$, SEP_{K,K_1} either asserts that $y \in S(K,\delta)$ or finds a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ such that $c^T x \leq c^T y + \delta$ for every $x \in K_1$. Then we obtain the same conclusion as in (3.2.1) except that we can only guarantee that K_1 is contained in $E(A,a)$. □

(3.2.34) Remark. For technical reasons, the "separation" oracle used in Theorem (3.2.1) is slightly stronger than a weak separation oracle. However, for well-bounded convex bodies (where we know an inner radius for the convex set), any weak separation oracle can be turned into one necessary for Theorem (3.2.1). This is immediate from the following lemma. □

(3.2.35) Lemma. *Let $(K;n,R,r)$ be a well-bounded convex body, $c \in \mathbb{R}^n$ with $\|c\|_\infty = 1$, and $\gamma,\delta \in \mathbb{R}$ with $r > \delta > 0$. Suppose that $c^T x \leq \gamma$ is valid for $S(K,-\delta)$. Then*

$$c^T x \leq \gamma + \frac{2R\delta}{r}\sqrt{n} \quad \textit{for all } x \in K.$$

Proof. Let x_0 be a point in K maximizing $c^T x$ over K. By definition (2.1.16), there is a point $a_0 \in \mathbb{R}^n$ with $S(a_0,r) \subseteq K$. Consider the function $f(x) := \frac{\delta}{r}x + (1-\frac{\delta}{r})x_0$. Set $a := f(a_0)$, then $f(S(a_0,r)) = S(a,\delta)$, and moreover, $S(a,\delta)$ is contained in $\mathrm{conv}(\{x_0\} \cup S(a_0,r))$. Since $x_0 \in K$, and $S(a_0,r) \subseteq K$, $S(a,\delta)$ is contained in K by convexity. Hence $a \in S(K,-\delta)$, and so we know that $c^T a \leq \gamma$. Now note that $x_0 = a + \frac{\delta}{r}(x_0 - a_0)$. Thus in order to estimate $c^T x_0$, we have to find a bound on $c^T(x_0 - a_0)$. Since $x_0, a_0 \in K$ and $K \subseteq S(0,R)$, we can conclude that $\|x_0 - a_0\| \leq 2R$, and since $\|c\|_\infty = 1$, we know that $\|c\| \leq \sqrt{n}$. Putting this together and using the Cauchy-Schwarz inequality (0.1.26) we get:

$$c^T x_0 \leq c^T(a + \frac{\delta}{r}(x_0 - a_0)) \leq \gamma + \frac{\delta}{r}c^T(x_0 - a_0) \leq \gamma + \frac{\delta}{r}\|c\|\|x_0 - a_0\| \leq \gamma + \frac{\delta}{r}\sqrt{n}\,2R,$$

and the claim is proved. □

*3.3 The Shallow-Cut Ellipsoid Method

We have already mentioned before that the central-cut ellipsoid method as described in the previous section does not make full use of the geometric idea behind it. It has been observed by many authors that, for instance, using deep cuts can speed up the method (see SHOR and GERSHOVICH (1979), and BLAND, GOLDFARB and TODD (1981) for a survey). A deeper idea due to YUDIN and NEMIROVSKIĬ (1976b) is the use of shallow cuts. These provide slower (though still polynomial time) termination but work for substantially weaker separation oracles. This method will allow us to derive (among other results) the polynomial

time equivalence of the weak membership (2.1.14) and weak separation problems (2.1.13) for centered convex bodies.

To formulate this method precisely, we have to define a **shallow separation oracle** for a convex set K. Before giving an exact definition, we describe the geometric idea on which the shallow-cut method is based. In the central-cut ellipsoid method we stop as soon as we have found a point almost in the convex set K. Now we want to find a point deep in K – even more, we are looking for an ellipsoid $E(A,a)$ containing K such that the concentrical ellipsoid $E((n+1)^{-2}A,a)$ is contained in K. The method stops as soon as such an ellipsoid $E(A,a)$ is found. If an ellipsoid $E(A,a)$ does not have this property, then $E((n+1)^{-2}A,a) \setminus K$ is nonempty, and we look for a halfspace $c^Tx \leq \gamma$ which contains K but does not completely contain $E((n+1)^{-2}A,a)$. Such a halfspace will be called a **shallow cut** since it may contain the center a of $E(A,a)$ in its interior – see Figure 3.10.

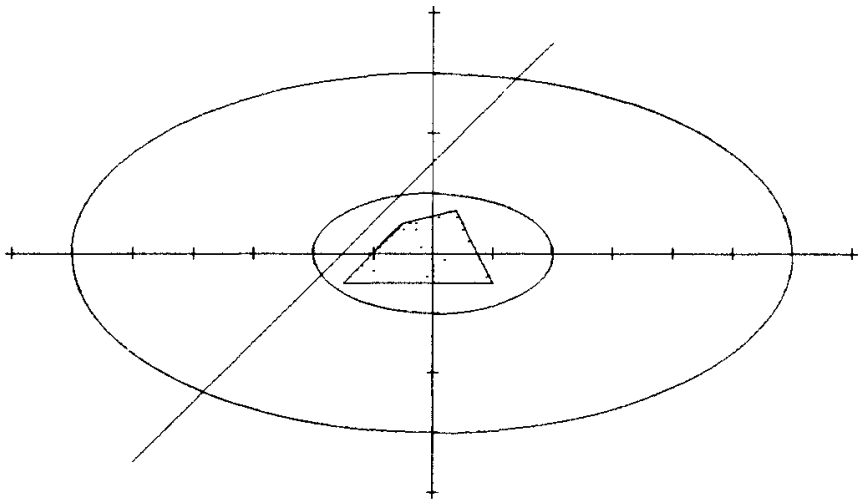


Figure 3.10

The method proceeds by determining the minimum volume ellipsoid containing

$$E(A,a) \cap \{x \mid c^Tx \leq c^Ta + \frac{1}{n+1}\sqrt{c^TAc}\} \tag{3.3.1}$$

and continues this way. Of course, since irrational numbers may come up, it will be necessary to round, and therefore the Löwner-John-ellipsoid has to be blown up a little bit as in the central-cut ellipsoid method.

Note that, by (3.1.8), the right hand side $c^Ta+(n+1)^{-1}\sqrt{c^TAc} = c^Ta+(n+1)^{-1}\|c\|_{A^{-1}}$ in (3.3.1) is the maximum value the linear function c^Tx assumes on the ellipsoid $E((n+1)^{-2}A,a)$. So the halfspace $\{x \in \mathbb{R}^n \mid c^Tx \leq c^Ta+(n+1)^{-1}\sqrt{c^TAc}\}$ contains this ellipsoid and supports it at the point $a+((n+1)\sqrt{c^TAc})^{-1}Ac$.

(3.3.2) Definition. *A* **shallow separation oracle for a convex set** $K \subseteq \mathbb{R}^n$ *is an oracle whose input is an ellipsoid* $E(A,a)$ *described by a positive definite matrix* $A \in \mathbb{Q}^{n\times n}$ *and a vector* $a \in \mathbb{Q}^n$. *A shallow separation oracle for* K *can write one of the following two possible answers on its output tape:*

(i) *a vector* $c \in \mathbb{Q}^n$, $c \neq 0$, *so that the halfspace* $H := \{x \in \mathbb{R}^n \mid c^T x \leq c^T a + (n+1)^{-1}\sqrt{c^T Ac}\}$ *contains* $K \cap E(A,a)$ *(a vector* c *with this property is called a* **shallow cut** *for* K *and* $E(A,a)$*),*

(ii) *the assertion that* $E(A,a)$ *is* **tough**.

□

At least two remarks are necessary to explain this definition. In answer (i), the inequality $c^T x \leq \gamma$ with $\gamma = c^T a + \frac{1}{n+1}\sqrt{c^T Ac}$ defining the halfspace H containing $K \cap E(A,a)$ has an irrational right hand side in general. But note that this right hand side γ is not written on the output tape. The oracle only confirms that H contains $K \cap E(A,a)$. In answer (ii) we have used the yet undefined word "tough". Loosely speaking, the word "tough" stands for "cutting is impossible". "Toughness" is a parameter left open, and in every instance of a shallow separation oracle the particular meaning of "tough" has to be specified. For instance, in the example described above a tough ellipsoid would be an ellipsoid $E(A,a)$ such that $E((n+1)^{-2}A, a)$ is contained in K. But there are other meaningful and interesting definitions of toughness possible.

We assume, as usual, that with each shallow separation oracle a polynomial function Φ is associated such that for every input to the oracle of encoding length at most L the encoding length of its output is at most $\Phi(L)$.

The aim of this section is to prove the following.

(3.3.3) Theorem. *There exists an oracle-polynomial time algorithm, called the* **shallow-cut ellipsoid method**, *that, for any rational number* $\varepsilon > 0$ *and for any circumscribed closed convex set* $(K; n, R)$ *given by a shallow separation oracle, finds a positive definite matrix* $A \in \mathbb{Q}^{n\times n}$ *and a point* $a \in \mathbb{Q}^n$ *such that one of the following holds:*

(i) $E(A,a)$ *has been declared tough by the oracle,*

(ii) $K \subseteq E(A,a)$ *and* $\mathrm{vol}(E(A,a)) \leq \varepsilon$.

□

Before giving a proof of a slightly more general version of this theorem, we want to illustrate it by three special cases.

(3.3.4) Example. Suppose that $K \subseteq \mathbb{R}^n$, $n \geq 2$, is a full-dimensional polytope given as the solution set of a system of linear inequalities

$$a_i^T x \leq \alpha_i, \quad i = 1, \ldots, m,$$

where $a_i \in \mathbb{Q}^n$, $\alpha_i \in \mathbb{Q}$ for $i = 1, \ldots, m$. We design a shallow separation oracle as follows. Let $E(A,a)$ be an ellipsoid given by A and a. For $i = 1, \ldots, m$, determine whether all points in the ellipsoid $E((n+1)^{-2}A, a)$ satisfy $a_i^T x \leq \alpha_i$. It follows from (3.1.8) that this can be done by checking whether $(n+1)^{-2} a_i^T A a_i \leq (\alpha_i - a_i^T a)^2$ holds. If an index i is found for which this inequality does not hold, then the oracle gives the shallow cut a_i as answer. If all inequalities of the given system are satisfied by all points in $E((n+1)^{-2}A, a)$, then the oracle declares $E(A,a)$ tough.

If L denotes the encoding length of the inequality system $a_i^T x \leq \alpha_i$, $i = 1, \ldots, m$, then we know from Lemma (3.1.33) that $K \subseteq S(0, R(K))$ with $R(K) := \sqrt{n}\, 2^{L-n^2}$ and from Lemma (3.1.35) that $\text{vol}(K) \geq \varepsilon(K) := 2^{-(n+1)L+n^3}$. So, by running the shallow-cut ellipsoid method of Theorem (3.3.3) with input $(K; n, R(K))$ and $\varepsilon = \varepsilon(K)$ and with the shallow separation oracle defined above, we will obtain an ellipsoid $E(A, a)$ containing K such that the concentrical ellipsoid $E((n+1)^{-2}A, a)$ is contained in K. □

Example (3.3.4) applies to more general situations. Namely, if for a circumscribed convex set K we have a shallow separation oracle where toughness of an ellipsoid $E(A, a)$ means that $E((n+1)^{-2}A, a) \subseteq K$, then the shallow-cut ellipsoid method solves the weak nonemptiness problem for K with the additional advantage that it gives a point deep inside K.

(3.3.5) Example. The central-cut ellipsoid method can be simulated by the shallow-cut ellipsoid method, and Theorem (3.2.1) is a consequence of Theorem (3.3.3). In fact, suppose that we have an oracle SEP_K as in (3.2.1) for a circumscribed closed convex set $(K; n, R)$ and that a positive rational number $\varepsilon > 0$ is given. Then we can design a shallow separation oracle for K as follows. Let an ellipsoid $E(A, a)$ be given by A and a. Compute a positive strict lower bound ε_1 for the square root of the least eigenvalue λ of A. Let $\delta' := \min\{\varepsilon, \varepsilon_1\}$ and $\delta := (n+1)^{-1}\delta'$. Call the oracle SEP_K for K with input $y := a$ and error parameter δ. If the oracle SEP_K asserts that $y \in S(K, \delta)$, then we declare the ellipsoid $E(A, a)$ tough. If SEP_K finds a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ and $c^T x \leq c^T y + \delta$ for all $x \in K$, then we take the vector c as output of the shallow separation oracle. By the choice of δ, the vector c is indeed a shallow cut for K and $E(A, a)$, namely, for all $x \in K$ we have

$$
\begin{aligned}
(3.3.6) \qquad c^T x &\leq c^T a + \delta \leq c^T a + (n+1)^{-1}\varepsilon_1 \leq c^T a + (n+1)^{-1}\sqrt{\lambda} \\
&= c^T a + (n+1)^{-1}\sqrt{\lambda}\, \|c\|_\infty \leq c^T a + (n+1)^{-1}\sqrt{\lambda}\, \|c\| \\
&\leq c^T a + (n+1)^{-1} \|c\|_{A^{-1}},
\end{aligned}
$$

where the last inequality follows from (0.1.9). In this case, toughness implies that the center of the tough ellipsoid is in $S(K, \delta)$ and hence in $S(K, \varepsilon)$. □

(3.3.7) Example. We can also turn a weak separation oracle into a shallow separation oracle provided an inner radius r for K is known. This follows directly by combining Remark (3.2.33) with Example (3.3.5). □

By definition (3.3.2), a shallow cut for K and $E(A, a)$ is a vector $c \in \mathbb{Q}^n$ such that $c^T x \leq c^T a + (n+1)^{-1}\sqrt{c^T A c}$ for all $x \in K \cap E(A, a)$. The parameter $(n+1)^{-1}$ used in the right hand side of this inequality is just a convenient choice out of an interval of possible parameters with which a shallow-cut method can be defined and for which it works. For our applications, greater generality is not necessary. But we will state and prove a slightly more general theorem to show what can be done.

(3.3.8) Definition. *For any rational number β with $0 < \beta < 1/n$ a* **shallow β-separation oracle for a convex set** $K \subseteq \mathbb{R}^n$ *is an oracle which, for an input a, A, where $a \in \mathbb{Q}^n$ and A is a rational positive definite $n \times n$-matrix, writes one of the following two answers on its output tape:*

(i) *a vector $c \in \mathbb{Q}^n$, $c \neq 0$, such that the halfspace $\{x \mid c^T x \leq c^T a + \beta\sqrt{c^T A c}\}$ contains $K \cap E(A,a)$ (such a vector c is called a* **shallow β-cut for K and** *$E(A,a)$),*

(ii) *the assertion that $E(A,a)$ is tough.* □

Observe that the halfspace $\{x \in \mathbb{R}^n \mid c^T x \leq c^T a + \beta\sqrt{c^T A c}\}$ contains and supports the ellipsoid $E(\beta^2 A, a)$. Clearly a shallow separation oracle as defined in (3.3.2) is a shallow $\frac{1}{n+1}$-oracle as defined above.

(3.3.9) Theorem. *There exists an algorithm, called the* **shallow-β-cut ellipsoid method**, *that, for any $\beta \in \mathbb{Q}$, $0 < \beta < 1/n$, and for any circumscribed closed convex set $(K; n, R)$ given by a shallow β-separation oracle, and for any rational $\varepsilon > 0$, finds, in time oracle-polynomial in $n + \langle R\rangle + \langle\varepsilon\rangle + \lceil(1-n\beta)^{-1}\rceil$, a positive definite matrix $A \in \mathbb{Q}^{n\times n}$ and a vector $a \in \mathbb{Q}^n$ such that one of the following holds:*

(i) *$E(A,a)$ has been declared tough by the oracle;*

(ii) *$K \subseteq E(A,a)$ and* $\mathrm{vol}(E(A,a)) \leq \varepsilon$.

Note that the algorithm we are going to design is not polynomial in the encoding length $\langle\beta\rangle$ of β. But if we choose β such that the encoding length of the number $(1-n\beta)^{-1}$ is bounded by a polynomial in the encoding length $n+\langle R\rangle+\langle\varepsilon\rangle$ of the other input (e. g., if we set $\beta := (n+1)^{-1}$), then the algorithm is truely oracle-polynomial. So Theorem (3.3.3) directly follows from Theorem (3.3.9).

Proof of Theorem (3.3.9). As in the proof of Theorem (3.2.1) we are going to describe a sequence of ellipsoids $E_0, E_1, \ldots$,i. e., we are going to construct a sequence of positive definite matrices $A_0, A_1, \ldots$ and a sequence of centers $a_0, a_1, \ldots$ such that $E_k = E(A_k, a_k)$. The algorithm we describe is the **shallow-β-cut ellipsoid method**. Set

$$N := \left\lceil \frac{5n}{(1-n\beta)^2} |\log\varepsilon| + \frac{5n^2}{(1-n\beta)^2} |\log(2R)| + |\log(1-n\beta)| \right\rceil \tag{3.3.10}$$

$$p := 8N. \tag{3.3.11}$$

We initialize the procedure by setting

$$\begin{aligned} a_0 &:= 0 \\ A_0 &:= R^2 I. \end{aligned} \tag{3.3.12}$$

Assume a_k, A_k are defined for some $k \geq 0$. If $k = N$, we stop. In this case the ellipsoid E_N contains K and has volume at most ε, so alternative (ii) of (3.3.9) is achieved. If $k < N$, we call the shallow β-separation oracle with $a = a_k$ and $A = A_k$.

If the oracle concludes that $E(A, a)$ is tough, then E_k has the desired property.

If the oracle gives a shallow β-cut c, then we perform the following computations.

$$a_{k+1} :\approx a_k - \rho \frac{A_k c}{\sqrt{c^T A_k c}}, \tag{3.3.13}$$

$$A_{k+1} :\approx A^*_{k+1} := \zeta \cdot \sigma\left(A_k - \tau \frac{A_k c c^T A_k}{c^T A_k c}\right), \tag{3.3.14}$$

where

$$\rho := \frac{1 - n\beta}{n+1}, \tag{3.3.15}$$

$$\sigma := \frac{n^2(1-\beta^2)}{n^2-1}, \tag{3.3.16}$$

$$\tau := \frac{2(1-n\beta)}{(n+1)(1-\beta)}, \tag{3.3.17}$$

$$\zeta := 1 + \frac{(1-n\beta)^2}{2n^2}. \tag{3.3.18}$$

Again "$\approx$" means that the left hand side is obtained by rounding the right hand side to p digits behind the point. (Note that without rounding and without blowing-up (i. e., with setting $\zeta := 1$) the update formulas above determine the Löwner-John-ellipsoid of $E'_k(A, a, c, \gamma)$ with $\gamma := c^T a_k + \beta\sqrt{c^T A_k c}$ – cf. (3.1.15), (3.1.16), (3.1.17).)

Similarly as in the case of the central-cut ellipsoid method, to establish the correctness of the algorithm we need the following lemmas.

(3.3.19) Lemma. *The matrices $A_0, A_1, \ldots$ are positive definite. Moreover,*

$$\|a_k\| \le R2^k, \ \|A_k\| \le R^2 2^k, \text{and} \|A_k^{-1}\| \le R^{-2}4^k.$$

(3.3.20) Lemma. $K \subseteq E_k$ *for* $k = 0, 1, \ldots$.

(3.3.21) Lemma. $\operatorname{vol}(E_{k+1})/\operatorname{vol}(E_k) \le e^{-(1-n\beta)^2/(5n)}$ *for* $k = 0, 1, \ldots$.

The first two of these lemmas can be proved along the same lines as Lemmas (3.2.8) and (3.2.9). We will prove Lemma (3.3.21), where the crucial condition $\beta < 1/n$ plays a role.

Proof of Lemma (3.3.21). As in the proof of Lemma (3.2.10) we write

$$\frac{\operatorname{vol}(E_{k+1})}{\operatorname{vol}(E_k)} = \sqrt{\frac{\det(A^*_{k+1})}{\det(A_k)}} \sqrt{\frac{\det(A_{k+1})}{\det(A^*_{k+1})}}, \tag{3.3.22}$$

where A_{k+1} is defined in (3.3.14), and we obtain for the first factor in (3.3.22)

$$\text{(3.3.23)}\quad \sqrt{\frac{\det(A_{k+1}^*)}{\det(A_k)}} = \sqrt{(\zeta\sigma)^n(1-\tau)} = \zeta^{n/2}\sigma^{(n-1)/2}\sqrt{\sigma(1-\tau)}$$
$$= \left(1+\frac{(1-n\beta)^2}{2n^2}\right)^{n/2}\left(\frac{n^2(1-\beta^2)}{n^2-1}\right)^{(n-1)/2}\frac{n(1+\beta)}{n+1}.$$

The first of the three factors in (3.3.23) can be easily estimated as follows:

$$\text{(3.3.24)}\quad \left(1+\frac{(1-n\beta)^2}{2n^2}\right)^{n/2} \le e^{(1-n\beta)^2/(4n)}.$$

To derive an upper bound for the last two factors in (3.3.23) take the natural logarithm ln (and recall the power series expansion of $\ln(1+x)$ and $\ln(1-x)$):

$$\text{(3.3.25)}\quad \ln\left(\left(\frac{n^2(1-\beta^2)}{n^2-1}\right)^{(n-1)/2}\cdot\frac{n(1+\beta)}{n+1}\right) =$$
$$= \frac{n-1}{2}\left(\ln(1-\beta^2) - \ln\left(1-\frac{1}{n^2}\right)\right) + \ln(1+\beta) - \ln\left(1+\frac{1}{n}\right)$$
$$= \frac{n-1}{2}\left(\sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{1}{n^{2k}}-\beta^{2k}\right)\right) + \sum_{k=1}^{\infty}\frac{(-1)^k}{k}\left(\frac{1}{n^k}-\beta^k\right)$$
$$= n\left(\sum_{k=1}^{\infty}\frac{1}{2k}\left(\frac{1}{n^{2k}}-\beta^{2k}\right)\right) - \sum_{k=1}^{\infty}\frac{1}{2k-1}\left(\frac{1}{n^{2k-1}}-\beta^{2k-1}\right)$$
$$= \sum_{k=1}^{\infty}\frac{-1}{2k(2k-1)n^{2k-1}}((2k-1)(n\beta)^{2k} - 2k(n\beta)^{2k-1} + 1)$$
$$\le \frac{-(1-n\beta)^2}{2n}$$

The last inequality follows from the observation that each term of the series on the left hand side is negative as $n\beta < 1$. Hence the first term $-(1-n\beta)^2/(2n)$ of this last sum is an upper bound for it.

Thus, from (3.3.24) and (3.3.25) we get

$$\text{(3.3.26)}\quad \sqrt{\frac{\det(A_{k+1}^*)}{\det(A_k)}} \le e^{(1-n\beta)^2/(4n)}e^{-(1-n\beta)^2/(2n)} = e^{-(1-n\beta)^2/(4n)}$$

The second factor in (3.3.22) can be estimated just like in the proof of Lemma (3.2.10), and we obtain

$$\text{(3.3.27)}\quad \sqrt{\frac{\det(A_{k+1})}{\det(A_{k+1}^*)}} \le e^{(1-n\beta)^2/(20n)}.$$

Combining inequalities (3.3.26) and (3.3.27) gives the desired result. This completes the proof of Lemma (3.3.21) and, by the same argument as in the proof of Theorem (3.2.1), also the proof of Theorem (3.3.9). □

As mentioned above, we will only use the shallow β-cut ellipsoid method for $\beta = \frac{1}{n+1}$ in the sequel, and that is what we call the shallow-cut ellipsoid method. Similarly, if $\beta = \frac{1}{n+1}$, a shallow β-separation oracle is called just a **shallow separation oracle**. The parameters used in the shallow-cut ellipsoid method are the following (compare with (3.3.10), ..., (3.3.18) and (3.2.2), ..., (3.2.7)):

$$
\begin{aligned}
N &: \lceil 5n(n+1)^2|\log\varepsilon| + 5n^2(n+1)^2|\log(2R)| + \log(n+1)\rceil,\\
p &:= 8N,\\
\rho &:= \frac{1}{(n+1)^2},\\
\sigma &:= \frac{n^3(n+2)}{(n+1)^3(n-1)},\\
\tau &:= \frac{2}{n(n+1)},\\
\zeta &:= 1 + \frac{1}{2n^2(n+1)^2}.
\end{aligned}
$$

So, in particular one can see that the number N of iterations of the shallow-cut ellipsoid method is about $(n+1)^2$ times as large as the number of iterations of the central-cut ellipsoid method. This, of course, matters for practical purposes, but is of no significance if one is only interested in polynomial time solvability.

Chapter 4

Algorithms for Convex Bodies

We shall now exploit the ellipsoid method (the central-cut and the shallow-cut version) described in Chapter 3. In Sections 4.2, 4.3, and 4.4 we study the algorithmic relations between problems (2.1.10),..., (2.1.14), and we will prove that – under certain assumptions – these problems are equivalent with respect to polynomial time solvability. Section 4.5 serves to show that these assumptions cannot be weakened. In Section 4.6 we investigate various other basic questions of convex geometry from an algorithmic point of view and prove algorithmic analogues of some well-known theorems. Finally, in Section 4.7 we discuss to what extent algorithmic properties of convex bodies are preserved when they are subjected to operations like sum, intersection etc.

As in Chapter 3, we will assume in the proofs that $n \geq 2$, if the one-variable case is trivial.

4.1 Summary of Results

In Chapter 2 we have introduced five fundamental problems concerning convex sets: the weak optimization problem (WOPT) (2.1.10), the weak violation problem (WVIOL) (2.1.11), the weak validity problem (WVAL) (2.1.12), the weak separation problem (WSEP) (2.1.13), and the weak membership problem (WMEM) (2.1.14). Moreover, as a special case of WVIOL, we have mentioned the weak nonemptiness problem (WNEMPT).

In Sections 4.2, 4.3, and 4.4 we study the algorithmic relations between these problems using the oracle concept. The idea behind this is the following. Suppose, for instance, that we have a convex set K for which we have an algorithm that solves the weak separation problem for K. We now want to see whether we can solve the weak optimization problem for K using the separation algorithm as a subroutine (oracle). Of course, what we are looking for is an algorithm solving WOPT for K that calls the separation subroutine only a polynomial number of times and whose further computations can be performed in a number of steps that is bounded by a polynomial function in the encoding length $\langle K \rangle + \langle c \rangle + \langle \varepsilon \rangle$ of the weak optimization problem. If we can achieve this, we say that WOPT for K can be solved in oracle-polynomial time using a separation oracle for K.

Such an algorithm for the weak optimization problem for K is not necessarily polynomial in $\langle K \rangle + \langle c \rangle + \langle \varepsilon \rangle$, since we did not make any assumptions on the running time of the weak separation algorithm for K. But – and this is the

most interesting application of such a result – if the algorithm solving the weak separation problem for K runs in time polynomial in the encoding length of K, then "oracle-polynomial" implies that the weak optimization problem can be solved in time polynomial in $\langle K\rangle + \langle c\rangle + \langle \varepsilon\rangle$ as well.

In Figure 2.2 we indicated the trivial relations between the five problems in question. For instance, if WOPT can be solved for some convex set K in polynomial time it is immediately clear from the definition that WVIOL can also be solved in polynomial time. We displayed this in Figure 2.2 by an arrow from WOPT to WVIOL. In Sections 4.2, 4.3, and 4.4 we show that many more arrows can be added to Figure 2.2 (not only those following from transitivity). From a weak optimization oracle for K we can derive oracle-polynomial time algorithms for all other problems. Some more implications follow if we assume that K is circumscribed, and in some cases additionally, that K is centered. Let us first remark that for uniformity reasons we restrict ourselves to considering convex *bodies* K only. (Most of the results also hold for bounded convex sets, as the proofs show.)

Figure 4.1 summarizes the results of Sections 4.2, 4.3, and 4.4. Unbroken lines correspond to the trivial implications already contained in Figure 2.2. Broken lines represent new implications derived in this chapter.

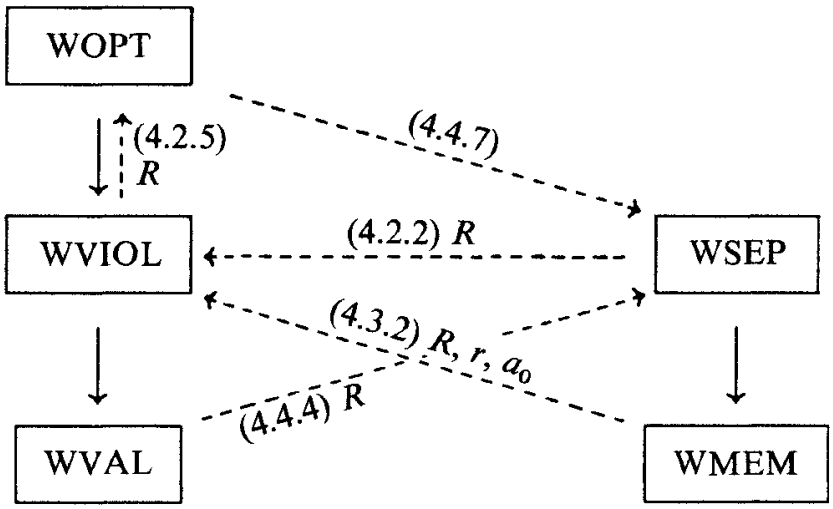


Figure 4.1

The implications represented in Figure 4.1 by unbroken lines are valid for any convex body K. To derive the implications represented by broken lines we assume that we know a radius R so that $K \subseteq S(0,R)$ (lines labeled with R), and, in some cases, that we know an inner radius r and a center a_0 (lines labeled with R, r, a_0). The only exception is the broken line from WOPT to WSEP. We can derive this implication for any convex body. Moreover, the arrows carry the additional information where a proof of the corresponding implication can be found. It is clear that, from the implications given in Figure 4.1, many more implications can be derived by transitivity.

For instance, if a convex body K is given by a weak violation oracle and an outer radius R then the weak optimization problem for K can be solved in oracle-polynomial time. The information along the arrow from WVIOL to WOPT states that this result can be found in Remark (4.2.5). If a circumscribed convex body K is given by a weak separation oracle and an outer radius R then we show in Theorem (4.2.2) that the weak violation problem for K is solvable in oracle-polynomial time. The proof uses the central-cut ellipsoid method. By transitivity it follows that if for a circumscribed convex body K the weak separation problem can be solved in polynomial time, the weak optimization problem can be solved in polynomial time. (We consider this as one of the most important results in this book.) Similarly, if a convex body K is given by a weak optimization oracle then the weak separation problem for K is solvable in oracle-polynomial time. This result is shown in Theorem (4.4.7).

The arrows from WMEM to WVIOL and further to WOPT show that for centered convex bodies given by a weak membership oracle the weak optimization problem can be solved in oracle-polynomial time (see Theorem (4.3.2) and Corollary (4.2.5)). This result, due to Yudin and Nemirovskiĭ, is surprising, because – at first sight – it does not seem apparent that a point almost optimizing a linear function over K can be found just by testing weak membership in K a polynomial number of times. For the derivation of this result the use of the shallow-cut ellipsoid method was necessary.

We do – of course – not claim that the (nontrivial) results shown in Figure 4.1 can only be derived from the ellipsoid method. For instance, Theorem (4.2.2), i. e., the arrow from WSEP to WVIOL can also be shown using the new "simplex method" due to YAMNITSKI and LEVIN (1982). We were not able to prove Theorem (4.3.2), i. e., the arrow from WMEM to WVIOL, using the central-cut ellipsoid method only. Our proof method only worked with the shallow-cut ellipsoid method, and probably, with a shallow-cut version of the new "simplex method" the same result can be obtained. Maybe there is a smart way to get Theorem (4.3.2) from the central-cut ellipsoid method or something even simpler. But we were not able to find such a proof.

In Section 4.5 we give some negative results showing that the assumptions made in the results of Sections 4.2, 4.3, and 4.4 cannot be weakened further. To illustrate the proof ideas we present here an example of this type. We show that we cannot derive an oracle-polynomial time weak optimization algorithm from a weak membership oracle, even if we know an outer radius R and an inner radius r (but not a center a_0).

Suppose that somebody claims he has such an algorithm. We volunteer to play the role of the oracle for a well-bounded convex body $(K; 1, R, 1)$ and challenge him to maximize the linear objective function $1 \cdot x$ over K in the weak sense with $\varepsilon = 1$. To the first R queries, we answer "no x is not in K". Note that this is a valid weak (even strong) membership oracle for some K, since after R questions there will be still a subinterval $K \subseteq [-R, R]$ of length 2 not containing any point asked for so far. So to solve the weak optimization problem he has to ask more than R questions. Since R is exponential in the encoding length of $(K; 1, R, 1)$ and $\varepsilon = 1$, his algorithm has to take exponential time.

In Section 4.6 we show that some other algorithmic problems on a well-

bounded convex body can be solved in oracle-polynomial time, if we have a weak separation oracle for this convex body. Among these problems are those of finding an approximation of the Löwner-John ellipsoid – cf. Theorem (3.1.9) – and of approximating the diameter and width of a convex body. The results use again the shallow-cut ellipsoid method.

Finally, in Section 4.7, we study how algorithmic properties of convex bodies K_1 and K_2 behave under operations like the sum $K_1 + K_2$, the convex hull $\text{conv}(K_1 \cup K_2)$ of the union, and the intersection $K_1 \cap K_2$, and under taking polars, blockers and anti-blockers. Basically, the polynomial time solvability of, say, the optimization problem is preserved by these operations, but the exact side conditions sometimes need a careful study.

*4.2 Optimization from Separation

We shall now show that, under reasonable assumptions, the fundamental problems concerning convex bodies introduced in Chapter 2 are equivalent with respect to polynomial time solvability. A main result of this book is that, for circumscribed convex bodies, we can derive a good weak optimization algorithm from a weak separation oracle. Our basic tool will be the central-cut ellipsoid method described in Chapter 3. The theorem to be described, can be stated in short-hand notation by:

(4.2.1) $$\boxed{\text{WSEP}, R} \longrightarrow \boxed{\text{WVIOL}}\ .$$

Here R means that if we know a radius R such that the convex body is contained in $S(0, R)$, then we can derive an oracle-polynomial time weak violation algorithm from a weak separation oracle. To state the result formally:

(4.2.2) Theorem. *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every circumscribed convex body* $(K; n, R)$ *given by a weak separation oracle.*

Proof. We describe the algorithm. An instance of the weak violation problem is given by the following input: a circumscribed convex body $(K; n, R)$ given by a weak separation oracle SEP_K, a vector $c \in \mathbb{Q}^n$, and rational numbers γ, ε with $\varepsilon > 0$. Without loss of generality we may assume $\|c\|_\infty = 1$. Set

$$K' := K \cap \{x \in \mathbb{R}^n \mid c^T x \geq \gamma\},$$
$$\varepsilon' := \frac{\varepsilon}{2n},$$

We design an oracle $\text{SEP}_{K', S(K', -\varepsilon')}$ in the sense of Remark (3.2.33) as follows.

Suppose $y \in \mathbb{Q}^n$ and a rational number $\delta > 0$ are given. We first check whether

(4.2.3) $$c^T y \geq \gamma + \delta$$

holds. If the answer is no, then we let $\mathrm{SEP}_{K',S(K',-\varepsilon')}$ give $-c$ as its output. If the answer is yes, call SEP_K with the input y and $\delta_1 := \min\{\varepsilon', \delta/n\}$. If SEP_K gives a vector d with $\|d\|_\infty = 1$ and $d^T x \leq d^T y + \delta$ for $x \in S(K, -\delta_1)$, then $\mathrm{SEP}_{K',S(K',-\varepsilon')}$ can give the same output (since $S(K', -\varepsilon') \subseteq S(K, -\delta_1)$). If SEP_K asserts that $y \in S(K, \delta_1)$, then $\mathrm{SEP}_{K',S(K',-\varepsilon')}$ can assert that $y \in S(K', \delta)$ (since, if $y \in S(K, \delta_1)$ satisfies (4.2.3), then $y \in S(K', \delta)$). This describes $\mathrm{SEP}_{K',S(K',-\varepsilon')}$.

We now run the central-cut ellipsoid method with input oracle $\mathrm{SEP}_{K',S(K',-\varepsilon')}$, $\varepsilon_1 := \min\{\varepsilon', (\varepsilon'/n)^n\}$, and R.

The output will be either a point $y \in S(K', \varepsilon_1)$ or an ellipsoid E containing $S(K', -\varepsilon')$ of volume at most ε_1. In the first case, $y \in S(K, \varepsilon)$ and $c^T y \geq \gamma - \|c\|\varepsilon_1 \geq \gamma - \varepsilon$. Thus y satisfies (2.1.11) (ii).

In the second case, we show that $c^T x \leq \gamma + \varepsilon$ for all $x \in S(K, -\varepsilon)$. Suppose to the contrary that $x \in S(K, -\varepsilon)$ and $c^T x > \gamma + \varepsilon$. Then $S(x, \varepsilon/n) \subseteq K'$, and hence $S(x, \varepsilon') \subseteq S(K', -\varepsilon') \subseteq E$. However,

$$\mathrm{vol}(E) \leq \varepsilon_1 < \mathrm{vol}(S(x, \varepsilon')),$$

which is a contradiction. □

Using binary search one can easily see the following implication:

(4.2.4) $$\boxed{\mathrm{WVIOL}, R} \longrightarrow \boxed{\mathrm{WOPT}}.$$

That is:

(4.2.5) Remark. *There exists an oracle-polynomial time algorithm that solves the weak optimization problem for every circumscribed convex body $(K; n, R)$ given by a weak violation oracle.* □

Combining Theorem (4.2.2) and Remark (4.2.5) we get the following implication, which, from the optimization point of view, has a number of significant consequences:

(4.2.6) $$\boxed{\mathrm{WSEP}, R} \longrightarrow \boxed{\mathrm{WOPT}}.$$

That is:

(4.2.7) Corollary. *There exists an oracle-polynomial time algorithm that solves the weak optimization problem for every circumscribed convex body $(K; n, R)$ given by a weak separation oracle.* □

By looking into the details of the ellipsoid method, a more direct algorithm can be obtained to maximize a linear objective function c over a circumscribed convex body $(K; n, R)$ given by a weak separation oracle. Informally, this method does the following.

Given some $\varepsilon > 0$, it constructs a sequence of ellipsoids that do not necessarily contain $S(K,\varepsilon)$ but do include the set of optimal points. If the center of the current ellipsoid is not in $S(K,\varepsilon)$ we cut like in the central-cut ellipsoid method.

Whenever the center of the current ellipsoid is in $S(K,\varepsilon)$, then we use the objective function as a cut. Consider those ellipsoid centers among the first t that are in $S(K,\varepsilon)$ and let c_t be the maximum of their objective function values. If c_{opt} denotes the optimum value of the optimization problem over K, then

$$|c_t - c_{\text{opt}}| \leq \frac{4nR^2}{\varepsilon\|c\|} e^{-t/(5n^2)}.$$

This method is called the **sliding objective function technique** – see BLAND, GOLDFARB and TODD (1981) or GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981) for details.

*4.3 Optimization from Membership

We prove now an important theorem due to YUDIN and NEMIROVSKIĬ (1976b). It states that one can derive an oracle-polynomial time weak violation algorithm from a weak membership oracle for a convex body K, if one knows a_0, r, R such that $S(a_0,r) \subseteq K \subseteq S(0,R)$. In our notation:

(4.3.1) $$\boxed{\text{WMEM}, R, r, a_0} \longrightarrow \boxed{\text{WVIOL}}.$$

That is:

(4.3.2) Theorem. *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every centered convex body* $(K; n, R, r, a_0)$ *given by a weak membership oracle.*

The natural approach to derive such an algorithm would be to derive a weak separation algorithm from the weak membership oracle and then to apply Theorem (4.2.2). There is, however, no direct way known at present to solve the weak separation problem in oracle-polynomial time using a weak membership oracle. As we shall see, Theorem (4.3.2) combined with a polarity argument implies that for a centered convex body given by a weak membership oracle, the weak separation problem can be solved in oracle-polynomial time. But this algorithm would involve the ellipsoid method for solving the weak violation problem and again the ellipsoid method to solve the weak separation problem. It is surprising that this back-and-forth argument has to be used, and it may be that a better understanding of the connection between membership and separation will lead to an algorithm that solves the weak separation problem in oracle-polynomial time using a weak membership oracle but not the ellipsoid method.

We shall, however, derive in Lemma (4.3.4) a "very weak" separation algorithm from a membership oracle (not using the ellipsoid method) which turns out to be the key to applying the shallow cut ellipsoid method to this problem. As a preparation we prove a simple lemma which shows that any weak membership algorithm can be used to strengthen itself.

(4.3.3) Lemma. *There exists an oracle-polynomial time algorithm that, for any centered convex body $(K; n, R, r, a)$ given by a weak membership oracle, for any vector $y \in \mathbb{Q}^n$ and any positive rational number δ, either*

(i) *asserts that $y \in S(K, \delta)$, or*
(ii) *asserts that $y \notin K$.*

Proof. Let $(K; n, R, r, a)$, $y \in \mathbb{Q}^n$, and $\delta \in \mathbb{Q}$, $\delta > 0$ be given as above. If $\|y-a\| \geq 2R$, then $y \notin K$. So we may assume that $\|y-a\| < 2R$. Call the weak membership oracle for K with

$$y' := \left(1 - \frac{\delta}{4R}\right)y + \frac{\delta}{4R}a \quad \text{and} \quad \delta' := \frac{r\delta}{4R}.$$

If it answers that y' belongs to $S(K, \delta')$ then, since

$$\|y - y'\| = \frac{\delta}{4R}\|y - a\| < \frac{\delta}{2},$$

we know $y \in S(K, \delta' + \delta/2) \subseteq S(K, \delta)$.

If the weak membership oracle answers that $y' \notin S(K, -\delta')$, then we assert that $y \notin K$. For, suppose by way of contradiction that $y \in K$. Then, since K is convex and $S(a, r)$ is contained in K, we know that $S(y', \delta r/(4R)) \subseteq K$. But this contradicts the conclusion of the membership oracle. □

The next lemma establishes a further (non-ellipsoidal) separation algorithm.

(4.3.4) Lemma. *There exists an algorithm that, for a point $y \in \mathbb{Q}^n$, for any two rational numbers $0 < \delta$, $\beta < 1$, and for any centered convex body $(K; n, R, r, a)$ given by a weak membership oracle, either*

(i) *asserts that $y \in S(K, \delta)$, or*
(ii) *finds a vector $c \in \mathbb{Q}^n$ such that $c \neq 0$ and for every $x \in K$,*
$c^T x \leq c^T y + (\delta + \beta\|x - y\|)\|c\|$.

The running time of the algorithm is oracle-polynomial in $\lceil 1/\beta \rceil$, $\langle K \rangle$, $\langle y \rangle$, and $\langle \delta \rangle$.

Note that an oracle-polynomial time weak separation algorithm for K could be obtained if the running time of the algorithm of (4.3.4) could be improved to be oracle-polynomial in $\langle \beta \rangle$ rather than in $\lceil 1/\beta \rceil$.

Proof. Let $(K; n, R, r, a)$, y, δ, β, be given as above. If $\|y - a\| \geq 2R$ then $c := (y-a)/\|y-a\|$ satisfies $c^T x \leq c^T y$ for all $x \in K$ and therefore (ii) is satisfied. So we may suppose in the following that $\|y - a\| < 2R$.

Call the algorithm of Lemma (4.3.3) with input K, y, and δ. If the algorithm concludes that $y \in S(K, \delta)$, then we are done.

Assume the algorithm concludes $y \notin K$. Set

$$\alpha := \operatorname{arc\,tg}\left(\frac{\beta}{4n^2}\right),$$

$$\delta_1 := \frac{r\delta}{R+r},$$

$$r_1 := \frac{r\delta_1}{4nR},$$

$$\varepsilon_1 := \frac{\beta^2 r_1}{16n^4}.$$

(To facilitate understanding the idea, we shall do real calculations and ignore the problems arising from rounding. We believe that the reader is convinced by the calculations of Chapter 3 that these roundings can be made precise, but also that he has enough from this.)

By binary search we find two points v and v', say, on the line segment $\overline{ay}$ between a and y such that v' is between a and v, $\|v - v'\| \leq \delta_1/(2n)$, $v \notin K$, but $v' \in S(K, \varepsilon_1)$. Then for the point

$$v'' := (r+\varepsilon_1)^{-1}((r-r_1)v' + (r_1+\varepsilon_1)a)$$

we have $S(v'', r_1) \subseteq K$ (since otherwise there exist $d \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$ such that $\|d\| = 1$, $d^T x \leq \gamma$ for all $x \in K$, and $d^T v'' > \gamma - r_1$; but then $d^T v \leq \gamma + \varepsilon_1$ and $d^T a \leq \gamma - r$, implying $d^T v'' = (r+\varepsilon_1)^{-1}((r-r_1)d^T v' + (r_1+\varepsilon_1)d^T a) \leq \gamma - r_1$).

Note that (using elementary geometry again)

$$\|v - v''\| \leq \frac{\delta_1}{2n} + \|v' - v''\| = \frac{\delta_1}{2n} + \frac{r_1+\varepsilon_1}{r-r_1}\|v'' - a\| < \frac{\delta_1}{n}. \tag{4.3.5}$$

Without loss of generality we may assume that $v'' = 0$.

Let H be the hyperplane perpendicular to v and going through the point $(\cos^2\alpha)v$. Let $v_1, \ldots, v_n$ be the vertices of a regular simplex in H with center in $(\cos^2\alpha)v$ such that the angle between v and v_i equals α, for $i = 1, \ldots, n$. (Note that in real arithmetic such points are easy to construct. The rounding (which we do not consider here) has to be done with some care so that the subsequent estimates still hold.) Thus

$$\|v_i\| = (\cos\alpha)\|v\|.$$

For each of the points v_i, run the algorithm of Lemma (4.3.3) with error parameter ε_1. Assume first that the algorithm concludes that one of the vectors $v_1, \ldots, v_n$ is not in K, say $v_1 \notin K$. Then perform the construction of the regular simplex described above again, now using v_1 as the point v, and repeat the calls of the algorithm of Lemma (4.3.3). Suppose we have iterated this step p times. Since $\|v_1\| = (\cos\alpha)\|v\|$ it follows that after the p-th iteration we have a vector $\bar{v} \notin K$ with $\|\bar{v}\| \leq (\cos\alpha)^p\|v\| < (\cos\alpha)^p R$. But since $S(0, r_1) \subseteq K$, this can only happen as long as

$$(\cos\alpha)^p R > r_1, \quad \text{i. e.,}$$

$$p < \frac{|\ln(R/r_1)|}{|\ln \cos \alpha|},$$

where ln denotes the natural logarithm. Using the inequality $\ln(1+x) \leq x$ for all x, we get

$$\begin{aligned} \ln \cos \alpha &= \tfrac{1}{2} \ln \cos^2 \alpha \\ &= \tfrac{1}{2} \ln(1 - \sin^2 \alpha) \\ &\leq -\tfrac{1}{2} \sin^2 \alpha \\ &= -\frac{(\beta/(4n^2))^2}{2(1 + (\beta/(4n^2))^2)} \\ &\leq \frac{-\beta^2}{34n^4}. \end{aligned}$$

It follows that after a number of iterations that is polynomial in n, $\langle r \rangle$, $\langle R \rangle$ and $\lceil 1/\beta \rceil$ we find that the algorithm of Lemma (4.3.3) concludes with $\bar{v}_1, \ldots, \bar{v}_n \in S(K, \varepsilon_1)$, where the vectors $\bar{v}_1, \ldots, \bar{v}_n$ have been constructed from a vector $\bar{v} \notin K$. We claim now that

$$c := \frac{\bar{v}}{\|\bar{v}\|_\infty}$$

satisfies (ii) of Lemma (4.3.4).

First we show that for all $x \in K$

$$c^T x \leq \beta \|x\| + \delta_1. \tag{4.3.6}$$

Consider any $x \in K$. Set

$$\begin{aligned} v_i' &:= \frac{r_1}{\varepsilon_1 + r_1} \bar{v}_i, \quad i = 1, \ldots, n \text{ and} \\ w &:= \frac{1}{n} \sum_{i=1}^{n} v_i'. \end{aligned}$$

Note that, by construction, $w = \gamma \bar{v}$ for

$$\gamma := (\cos^2 \alpha) r_1 / (\varepsilon_1 + r_1) = \cos^4 \alpha.$$

Similarly as before, since $S(0, r_1) \subseteq K$, it follows that $v_i' \in K$ for $i = 1, \ldots, n$. Let us represent x in the following form

$$x = \lambda \bar{v} + u \tag{4.3.7}$$

where $u^T \bar{v} = 0$.

Case 1. $\lambda \leq 1$. Then by (4.3.5)

$$c^T x = \lambda c^T \bar{v} \leq c^T \bar{v} = \frac{\|\bar{v}\|^2}{\|\bar{v}\|_\infty} \leq n\|\bar{v}\| \leq n\|v\| \leq \delta_1 \leq \beta\|x\| + \delta_1$$

which gives (4.3.6).

Case 2. $\lambda > 1$. Then we transform (4.3.7) into the following form

$$\gamma\bar{v} + \frac{\gamma - 1}{\lambda - 1}u = \frac{\gamma - 1}{\lambda - 1}x + \frac{\lambda - \gamma}{\lambda - 1}\bar{v}. \tag{4.3.8}$$

Let z denote the point determined by either side of (4.3.8). Looking at the right hand side of (4.3.8) we see that $\bar{v}$ is a convex combination of z and x, i. e., $\bar{v}$ is a point on the line segment connecting z to x, and since $x \in K$ and $\bar{v} \notin K$ it follows that $z \notin K$. On the other hand, the left hand side of (4.3.8) tells us that z is on the hyperplane through $v_1', \ldots, v_n'$. Since $z \notin K$ and hence $z \notin \operatorname{conv}\{v_1', \ldots, v_n'\}$ it follows that

$$\|z - w\| \geq \frac{1}{n}\|v_1' - w\| = \frac{1}{n}\operatorname{tg}\alpha\|w\|,$$

where the last equation follows from the fact that the angle between w and v_1' is α. Hence

$$\|u\| = \left|\frac{\lambda - 1}{\gamma - 1}\right| \|z - w\| \geq \frac{\lambda - 1}{1 - \gamma}\frac{\gamma}{n}\operatorname{tg}\alpha\|\bar{v}\|.$$

Using (4.3.7) again we obtain

$$\begin{aligned}\|x\| = \sqrt{\lambda^2\bar{v}^T\bar{v} + u^Tu} \geq \|u\| \geq \frac{\lambda - 1}{1 - \gamma}\cdot\frac{\gamma}{n}\operatorname{tg}\alpha\|\bar{v}\| = \frac{(\lambda - 1)\|\bar{v}\|}{n}\cdot\frac{\gamma\operatorname{tg}\alpha}{1 - \gamma} = \\ = \frac{(\lambda - 1)\|\bar{v}\|}{n}\cdot\frac{\cos^4\alpha\cdot\operatorname{tg}\alpha}{1 - \cos^4\alpha} = \frac{(\lambda - 1)\|\bar{v}\|}{n}\cdot\frac{\cos^4\alpha\cdot\operatorname{tg}\alpha}{\sin^2\alpha(1 + \cos^2\alpha)} \geq \\ \geq \frac{(\lambda - 1)\|\bar{v}\|}{2n}\cdot\frac{\cos^2\alpha}{\operatorname{tg}\alpha} = \frac{(\lambda - 1)\|\bar{v}\|}{2n}\cdot\frac{16n^4 + \beta^2}{8n^2\beta} \geq (\lambda - 1)\|\bar{v}\|\frac{n}{\beta}.\end{aligned}$$

Hence

$$\lambda \leq \frac{\|x\|}{\|\bar{v}\|}\frac{\beta}{n} + 1.$$

Again from (4.3.7) we get

$$\bar{v}^Tx = \lambda\|\bar{v}\|^2 \leq \frac{\beta}{n}\|\bar{v}\|\|x\| + \|\bar{v}\|^2.$$

So, by (4.3.5)

$$\begin{aligned}c^Tx = \frac{\bar{v}^T}{\|\bar{v}\|_\infty}x &\leq \frac{\beta}{n}\frac{\|\bar{v}\|}{\|\bar{v}\|_\infty}\|x\| + \frac{\|\bar{v}\|^2}{\|v\|_\infty} \\ &\leq \beta\|x\| + n\|\bar{v}\| \\ &\leq \beta\|x\| + \delta_1.\end{aligned}$$

This completes the proof of inequality (4.3.6) for all $x \in K$.

Finally, to show that (ii) holds for all $x \in K$, consider

$$x' := \frac{r}{R + r}(x - y).$$

Then $x' \in K$, since x' is the following convex combination

$$x' = \frac{r}{R+r}x + \frac{R}{R+r}\left(\frac{-r}{R}y\right),$$

where x is in K by assumption and $(-r/R)y$ is in K as this is a point on the segment connecting $0 \in K$ and $a - (r/\|y\|)y \in K$.

Substituting x' into (4.3.6) we obtain

$$\frac{r}{R+r}c^T(x-y) = c^T x' \leq \beta\frac{r}{R+r}\|x-y\| + \delta_1,$$

and hence

$$c^T(x-y) \leq \beta\|x-y\| + \delta_1\frac{R+r}{r} = \delta + \beta\|x-y\|.$$

Since $\|c\| \geq 1$ the proof is finished. □

Now we are prepared to prove Theorem (4.3.2).

Proof of Theorem (4.3.2). Let $(K; n, R, r, a_0)$ be a centered convex body given by a weak membership oracle. We design a weak violation algorithm for K. To this end let further a vector $c \in \mathbb{Q}^n$ and rational numbers γ, ε with $\varepsilon > 0$ be given. Let

$$K' = K \cap \{x \in \mathbb{R}^n \mid c^T x \geq \gamma\}.$$

We define a shallow separation oracle for K'. Let $E(A, a)$ be any ellipsoid. First check if $c^T a \leq \gamma$. If so, then the vector $-c$ and the number $-\gamma$ are a valid output for the shallow separation oracle for K'. Suppose now

$$c^T a > \gamma. \tag{4.3.9}$$

Let Q be the positive definite matrix such that $(n+1)^2 A^{-1} = QQ$ and consider the well-bounded convex body $(QK; n, \|Q^{-1}\|^{-1}r, \|Q\|R, Qa_0)$. From the weak membership oracle for K we easily get a weak membership oracle for QK. Now we apply the algorithm of Lemma (4.3.4) for QK to the input $y := Qa \in \mathbb{Q}^n$, $\delta := \min\{(n+2)^{-2}, \varepsilon/\|Q^{-1}\|\}$ and $\beta := (n+2)^{-1}$.

If the algorithm concludes that $Qa \in S(QK, \delta)$ then the shallow separation oracle for K' should declare the ellipsoid $E(A, a)$ tough. Note that in this case $a \in S(K, \delta\|Q^{-1}\|) \subseteq S(K, \varepsilon)$.

If the algorithm of Lemma (4.3.4) finds a vector $d \in \mathbb{Q}^n$ such that $d \neq 0$ and

$$d^T x \leq d^T Qa + (\delta + \beta\|x - Qa\|)\|d\|$$

holds for all $x \in QK$, then this is equivalent to saying that

$$d^T Qz \leq d^T Qa + (\delta + \beta\|Q(z-a)\|)\|d\|$$

holds for all $z \in K$. If $z \in E(A, a)$ then

$$\|Q(z-a)\| = \sqrt{(z-a)^T QQ(z-a)} = (n+1)\sqrt{(z-a)^T A^{-1}(z-a)} \leq n+1$$

by the definition of $E(A,a)$. Hence if $z \in K \cap E(A,a)$ then

$$d^TQz \leq d^TQa + \left((n+2)^{-2} + \frac{n+1}{n+2}\right)\|d\| \tag{4.3.10}$$
$$< d^TQa + \left(1 - \frac{1}{(n+2)^2}\right)\|d\|.$$

So the halfspace $d^TQz \leq d^TQa + (1 - 1/(n+2)^2)\|d\| =: \gamma$ includes $K \cap E(A,a)$. On the other hand, this halfspace does not completely include $E((n+1)^{-2}A, a)$ as the vector

$$a + \frac{Q^{-1}d}{\|d\|}$$

belongs to $E((n+1)^{-2}A,a)$ but violates $d^TQz \leq \gamma$. Hence the vector $d' := Qd$ and the number γ are a valid output of the shallow separation oracle.

To the convex set K' given by the shallow separation algorithm defined above, the rational numbers $\varepsilon_1 := ((r\varepsilon/(2Rn))^n$ and (the given) R we now apply the shallow-cut ellipsoid method (Theorem (3.3.3)). If the shallow-cut ellipsoid method concludes with a tough ellipsoid $E(A,a)$, then by the remark made above, $a \in S(K,\varepsilon)$, and so the vector a is a solution of the weak violation problem for K by (4.3.9).

If the shallow-cut ellipsoid method concludes with an ellipsoid $E(A,a)$ containing K and volume at most ε, then just as in the proof of Theorem (4.2.1) we can assert that $c^Tx \leq \gamma + \varepsilon$ for all $x \in S(K,-\varepsilon)$. □

This proves the Yudin-Nemirovskiĭ theorem. Combining this with Remark (4.2.5) we obtain:

$$\boxed{\text{WMEM}, R, r, a_0} \longrightarrow \boxed{\text{WOPT}}\,. \tag{4.3.11}$$

That is:

(4.3.12) Corollary. *There exists an oracle-polynomial time algorithm that solves the weak optimization problem for every centered convex body $(K;n,R,r,a_0)$ given by a weak membership oracle.* □

As an important application of this result we show that the weak constrained convex function minimization problem (2.1.22) can be solved in oracle-polynomial time.

(4.3.13) Theorem. *There exists an oracle-polynomial time algorithm that solves the following problem:*

Input: *A rational number $\varepsilon > 0$, a centered convex body $(K;n,R,r,a_0)$ given by a weak membership oracle, and a convex function $f : \mathbb{R}^n \to \mathbb{R}$ given by an oracle that, for every $x \in \mathbb{Q}^n$ and $\delta > 0$, returns a rational number t such that $|f(x) - t| \leq \delta$.*

Output: *A vector $y \in S(K,\varepsilon)$ such that $f(y) \leq f(x) + \varepsilon$ for all $x \in S(K,-\varepsilon)$.*

Proof. Consider the following set

$$G(f,K) := \{(x^T,t)^T \in \mathbb{R}^{n+1} \mid x \in K, f(x) \leq t \leq B\},$$

where B is computed as follows:

$$B := r + \max\{|f(a_0)|, |f(a_0 \pm re_i)| (i = 1, \ldots, n)\}.$$

A simple geometric argument shows that $-6RBn/r$ is a lower bound on the values of f in K. Hence

$$\left(G(f,K); n+1, \frac{7RBn}{r}, \frac{r}{n}, \begin{pmatrix} a_0 \\ B - r/2 \end{pmatrix}\right)$$

is a centered convex body. By the definition of $G(f,K)$, one easily derives a weak membership algorithm for this centered convex body from the oracles given in the theorem. So by Theorem (4.3.2), we can solve the weak optimization problem for $G(f,K)$ with input $c = (0^T,-1)^T$. The solution to this problem yields a solution for the weak constrained convex function minimization problem. □

*4.4 Equivalence of the Basic Problems

Now we combine the results of the previous two sections with a polarity argument to prove two further implications indicated in Figure 4.1.

Recall that the polar K^* of a set K is defined as follows:

$$K^* := \{y \in \mathbb{R}^n \mid y^T x \leq 1 \text{ for all } x \in K\}.$$

Note that, if $(K; n, R, r, 0)$ is a 0-centered, convex body, then so is $(K^*; n, 1/r, 1/R, 0)$; and moreover, $(K^*)^* = K$. In addition, a weak membership oracle for K^* is essentially equivalent to a weak validity oracle for K. More precisely,

(4.4.1) Lemma. *There exists an oracle-polynomial time algorithm that solves the weak membership problem for K^*, where K is a 0-centered convex body given by a weak validity oracle.*

Proof. Let $(K; n, R, r, 0)$ be a 0-centered convex body given by a weak validity oracle. We are going to describe a weak membership algorithm for K^*. Let $y \in \mathbb{Q}^n$ and $\delta \in \mathbb{Q}$, $\delta > 0$ be given. If $\|y\| \leq 1/R$, then $y \in K^*$. So we may assume $y \neq 0$. Call the weak validity oracle for K with input $c := y$, $\gamma := 1$ and

$$\varepsilon < \frac{r\delta}{\|y\| + r\|y\| + r\delta}.$$

(i) Suppose the weak violation oracle for K asserts that $c^T x \leq 1+\varepsilon$ for all $x \in S(K,-\varepsilon)$. We claim that this implies $y \in S(K,\delta)$. Namely, for every $x \in K$,

$$\frac{r-\varepsilon}{r} x \in S(K,-\varepsilon)$$

holds by a simple geometric argument. Hence for every $x \in K$

$$\frac{r-\varepsilon}{r} y^T x \leq 1+\varepsilon.$$

Therefore,

$$y_1 := \frac{1}{1+\varepsilon}\frac{r-\varepsilon}{r} y \in K^*.$$

By the choice of ε, $\|y_1 - y\| \leq \delta$ and hence $y \in S(K^*,\delta)$.

(ii) Suppose the weak validity oracle for K asserts that $y^T x \geq 1-\varepsilon$ for some $x \in S(K,\varepsilon)$. We claim that this implies that $y \notin S(K^*,-\delta)$. Assume to the contrary that $y \in S(K^*,-\delta)$. Then

$$\left(1+\frac{\delta}{\|y\|}\right) y \in K^*.$$

Moreover, by a simple geometric argument, we have

$$\frac{r}{r+\varepsilon} x \in K.$$

However,

$$\left(1+\frac{\delta}{\|y\|}\right) y^T \frac{r}{r+\varepsilon} x \geq \left(1+\frac{\delta}{\|y\|}\right)\frac{r}{r+\varepsilon}(1-\varepsilon) > 1.$$

This contradicts the definition of polarity. □

By a similar argument, we obtain the following lemma.

(4.4.2) Lemma. *There exists an oracle-polynomial time algorithm that solves the weak separation problem for K^*, where K is a 0-centered convex body given by a weak violation oracle.* □

The theorem we are going to prove is

(4.4.3) $$\boxed{\text{WVAL}, R} \longrightarrow \boxed{\text{WSEP}}.$$

That is:

(4.4.4) Theorem. *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every circumscribed convex body* $(K; n, R)$ *given by a weak validity oracle.*

Proof. I. First we prove the correctness of the theorem for centered convex bodies. So let $(K; n, R, r, a_0)$ be a centered convex body. Set $\overline{K} := K - a_0$. Then $(\overline{K}; n, 2R, r, 0)$ is a 0-centered convex body, and it is trivial to obtain a weak validity oracle for $\overline{K}$ from one for K. By Lemma (4.4.1), the weak membership problem for the polar $\overline{K}^*$ can be solved in oracle-polynomial time. Now we apply Theorem (4.3.2) and derive that the weak violation problem for $\overline{K}^*$ can be solved in oracle-polynomial time, and therefore by Lemma (4.4.2) the weak separation problem for $\overline{K} = (\overline{K}^*)^*$ can be solved in oracle-polynomial time. Now we only have to translate $\overline{K}$ back to K and adjust the weak separation algorithm for $\overline{K}$ to get one for K.

II. If a circumscribed convex body $(K; n, R)$ is given, we extend K to a convex body $\hat{K} \subseteq \mathbb{R}^{n+1}$ by appending an $(n+1)$-st coordinate and setting

$$\hat{K} := \operatorname{conv}(\{(x^T, 0)^T \in \mathbb{R}^{n+1} \mid x \in K\} \cup S(e_{n+1}, 1/2)),$$

where e_{n+1} denotes the $(n+1)$-st unit vector in $\mathbb{R}^{n+1}$. Clearly, $(\hat{K}, n+1, R+1, 1/2, e_{n+1})$ is a centered convex body in $\mathbb{R}^{n+1}$, and it is trivial to design a weak validity oracle for $\hat{K}$ given such an oracle for K. By Part I, we can solve the weak separation problem for $\hat{K}$ in oracle-polynomial time. Now we show that we can derive from this a weak separation algorithm for K.

Namely, let $y \in \mathbb{Q}^n$ and a rational $\delta > 0$ be given. Set $\hat{y} := (y^T, t)^T$, where $t := \delta/(10(\|y\|_1 + R))$. Call the weak separation algorithm for $\hat{K}$ with input $\hat{y}$ and $\hat{\delta} := t/(4Rn)$. If this concludes that $\hat{y} \in S(\hat{K}, \hat{\delta})$, then a simple geometric argument shows that $y \in S(K, \delta)$. Suppose that the separation algorithm finds a vector $\hat{c} = (c^T, \gamma)^T \in \mathbb{Q}^{n+1}$ such that $\|\hat{c}\|_\infty = 1$ and $\hat{c}^T\hat{x} \leq \hat{c}^T\hat{y} + \hat{\delta}$ for each $\hat{x} \in S(\hat{K}, -\hat{\delta})$. Then by Lemma (3.2.35),

$$\hat{c}^T\hat{x} \leq \hat{c}^T\hat{y} + \hat{\delta} + 4R\hat{\delta}n \tag{4.4.5}$$

for all $\hat{x} \in \hat{K}$. In particular, this holds for all $\hat{x} = (x^T, 0)^T$ where $x \in K$, i. e.,

$$c^T x \leq c^T y + \gamma t + \hat{\delta} + 4R\hat{\delta}n.$$

Inequality (4.4.5) also holds for $\hat{x} = e_{n+1}$, and hence $\gamma \leq c^T y + \gamma t + \hat{\delta} + 4R\hat{\delta}n$. This implies that

$$\|c\|_\infty \geq \min\left\{1, \frac{1}{2\|y\|_1}\right\}.$$

Set $d := c/\|c\|_\infty$. Then for every $x \in K$ we obtain

$$d^T x \leq d^T y + \frac{\gamma t + \hat{\delta} + 4R\hat{\delta}n}{\|c\|_\infty} \leq d^T y + \delta.$$

So d is a valid output of the weak separation algorithm for K. □

The above theorem implies that for a circumscribed convex body, given by a weak optimization oracle, the weak separation problem can be solved in oracle polynomial time. Our next result shows that we do not even need the outer radius for this implication:

(4.4.6) $$\boxed{\text{WOPT}} \longrightarrow \boxed{\text{WSEP}}.$$

That is:

(4.4.7) Theorem. *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every convex body given by a weak optimization oracle.*

Proof. Let $K \subseteq \mathbb{R}^n$ be a convex body given by a weak optimization oracle WOPT, and let the input $y \in \mathbb{Q}^n$ and $\delta > 0$ for the weak separation problem for K be given.

We first call WOPT $2n$ times with input $\varepsilon := \delta/2$ and $c := \pm e_i$, $i = 1, \ldots, n$. If WOPT ever answers that $S(K, -\varepsilon)$ is empty, then any vector with maximum norm 1 will be a valid answer for the weak separation problem. Otherwise we obtain a box that contains $S(K, -\delta/2)$, and $z \in S(K, \varepsilon)$, and hence we obtain an $R' > 0$ such that $S(K, -\delta/2) \subseteq S(0, R')$ and $S(0, R') \cap K \neq \emptyset$. By the General Assumption (1.2.1), the encoding length of R' is bounded by a polynomial in n and $\langle \delta \rangle$. Let $R := 3R'$. By a simple geometric argument one can prove:

Claim. Either $S(K, -\delta)$ is empty or $K \subseteq S(0, R)$. (However, we do not know which case occurs!)

We set $K' := K \cap S(0, R)$. $(K'; n, R)$ is a circumscribed convex body. Now we design a weak optimization subroutine for K'. For any input $c' \in \mathbb{Q}^n$ and $\varepsilon' > 0$ we call WOPT with input $c := c'$ and $\varepsilon := \min\{\delta, \varepsilon'/10\}$. We may assume that WOPT does not give the answer $S(K, -\varepsilon)$ is empty, nor does it give a $y' \in S(K, \varepsilon)$ such that $\|y'\| > R + \varepsilon$, because in both cases we can conclude that $S(K, -\delta) = \emptyset$, and thus the answer to the weak separation problem for K is trivial. So WOPT returns $y' \in S(K, \varepsilon) \cap S(0, R + \varepsilon)$ such that $c^T x \leq c^T y' + \varepsilon$ for all $x \in S(K, -\varepsilon)$. Since $S(K, \varepsilon) \cap S(0, R + \varepsilon) \subseteq S(K', 10\varepsilon)$ and $S(K', -\varepsilon) \subseteq S(K, -\varepsilon)$, y' is a valid output for the weak optimization problem for K'.

By the remark above we can solve the weak separation problem for $(K'; n, R)$ and the given input y, δ in oracle-polynomial time. Note that $S(K', \delta) \subseteq S(K, \delta)$ and by the claim $S(K', -\delta) = S(K, -\delta)$, and hence the output for the weak separation problem for K' is also valid for K. □

We close this section by proving an implication showing that it is justified to assume the knowledge of a center a_0 in Theorem (4.3.2), because such a center can be derived from a weak optimization oracle and an inner radius r:

(4.4.8) $$\boxed{\text{WOPT}, r} \longrightarrow \boxed{r, a_0}.$$

That is:

(4.4.9) Theorem. *There exists an oracle-polynomial time algorithm that, for any given convex body K, specified by a weak optimization oracle, and for any given inner radius r, constructs a vector a_0 and a rational number $r' > 0$ so that $S(a_0, r') \subseteq K$.*

Proof. Let $K \subseteq \mathbb{R}^n$ be a convex body given by a weak optimization oracle and an inner radius r. In the proof of (4.4.7) we saw that we can derive an outer radius R from this. To determine a ball $S(a_0, r')$ contained in K, let

$$\varepsilon := \left(\frac{r}{8nR}\right)^n.$$

First give the input $c := 0$, ε to the oracle, yielding a vector $y_0 \in S(K, \varepsilon)$.

Next determine $y_1, \ldots, y_n$ inductively as follows. If $y_0, \ldots, y_{j-1}$ have been found ($1 \leq j \leq n$), choose any nonzero vector c with $c^T y_0 = c^T y_1 = \ldots = c^T y_{j-1}$ and $\|c\|_\infty = 1$. Give the inputs c, ε and $-c, \varepsilon$ to the oracle. It gives us vectors y' and y'' in $S(K, \varepsilon)$ so that $c^T y'' - \varepsilon \leq c^T x \leq c^T y' + \varepsilon$ for all x in $S(K, -\varepsilon)$. Let $y_j := y'$ if $c^T y' - c^T y_0 \geq c^T y_0 - c^T y''$; let $y_j := y''$ otherwise. Since K contains a ball of radius r, $S(K, -\varepsilon)$ contains a ball of radius $r - \varepsilon$. So

$$d(y_j, \operatorname{aff}\{y_0, \ldots, y_{j-1}\}) \geq r - 2\varepsilon.$$

Therefore, the induction hypothesis implies that the j-dimensional volume of $\operatorname{conv}\{y_0, \ldots, y_j\}$ is at least $(r - \varepsilon)^j / j!$.

On the other hand, for each $j = 0, \ldots, n$, the $(n-1)$-dimensional volume of $\operatorname{conv}\{y_0, \ldots, y_{j-1}, y_{j+1}, \ldots, y_n\}$ is at most $(2(R + r))^n$, since $\|y_0\|, \ldots, \|y_n\| \leq R + \varepsilon \leq R + r$. Hence taking

$$a_0 := \frac{1}{n+1}(y_0 + \ldots + y_n), \text{ and}$$
$$r' := \frac{1}{n!}\left(\frac{r - 2\varepsilon}{2(R + r)}\right)^n - \varepsilon > 0,$$

we have $S(a_0, r' + \varepsilon) \subseteq S(K, \varepsilon)$, and hence $S(a_0, r') \subseteq K$. □

*4.5 Some Negative Results

In this section we give some negative results showing that there are no reductions between the basic algorithmic problems other than, and no reductions under weaker hypotheses than, those following from Figure 4.1.

One negative result was shown already in Section 4.1, namely that one cannot derive an oracle-polynomial time weak optimization algorithm from a weak membership oracle, even if one knows an outer radius R and an inner radius r. In our notation:

(4.5.1) $$\boxed{\text{WMEM}, R, r} \not\rightarrow \boxed{\text{WOPT}}.$$

Most of the other negative results in this section show that knowledge of an outer radius R is quite essential.

Next we show that one cannot derive an oracle-polynomial time weak violation algorithm from a weak validity oracle, even if one knows a ball $S(a_0, r)$ contained in the convex body. In short-hand notation:

(4.5.2) $$\boxed{\text{WVAL}, r, a_0} \rightarrow\!\!\!\!+\ \boxed{\text{WVIOL}}\ .$$

Proof. Suppose, to the contrary, that one can derive such an algorithm. Let $n = 2$, $a_0 = (0,0)^T$, $r = 1$. We give the input $c = (0,1)^T$, $\gamma = 3$, $\varepsilon = 0.2$ to the algorithm. In executing the algorithm, the weak validity oracle turns out to give the following answers, for input $\tilde{c} = (\tilde{c}_1, \tilde{c}_2)^T, \tilde{\gamma}, \tilde{\varepsilon}$:

(i) if $\|\tilde{c}\| \geq \tilde{\gamma}$, or if $\tilde{c}_1 > 0$, or if $\tilde{c}_1 = 0$ and $5\tilde{c}_2 > \tilde{\gamma}$, it asserts that $\tilde{c}^T x \geq \tilde{\gamma} - \tilde{\varepsilon}$ for all x in $S(K, \tilde{\varepsilon})$;
(ii) otherwise, it asserts that $\tilde{c}^T x \leq \tilde{\gamma} + \tilde{\varepsilon}$ for all x in $S(K, \tilde{\varepsilon})$.

Note that these answers are consistent with

$$K = \operatorname{conv}\big(S(a_0, r) \cup \{(\lambda, 5)^T\}\big)$$

for arbitrary large λ. It follows that any output for WVIOL derived from this is unfounded. □

Next we show that one cannot derive an oracle-polynomial time algorithm for the weak optimization problem from an oracle for the weak violation problem, even if one knows a ball $S(a_0, r)$ contained in the convex body. In short-hand notation:

(4.5.3) $$\boxed{\text{WVIOL}, r, a_0} \rightarrow\!\!\!\!+\ \boxed{\text{WOPT}}\ .$$

Proof. Suppose, to the contrary, that one can derive such an algorithm. Let $n = 1$, $a_0 = 0$, $r = 1$. We give the input $c = 1$, $\varepsilon = 1$ to the weak optimization algorithm. In executing the algorithm, the weak violation oracle turns out to give the following answers to the input $\tilde{c}$, $\tilde{\gamma}$, $\tilde{\varepsilon}$:

(i) if $\tilde{c} \neq 0$, it asserts that $x := \tilde{\gamma}/\tilde{c}$ belongs to $S(K, \tilde{\varepsilon})$ (it satisfies $\tilde{c}^T x > \tilde{\gamma} - \tilde{\varepsilon}$);
(ii) if $\tilde{c} = 0$, $\tilde{\gamma} \geq 0$ it asserts that $\tilde{c}^T x \leq \tilde{\gamma} + \tilde{\varepsilon}$ for all x in $S(K, -\tilde{\varepsilon})$;
(iii) if $\tilde{c} = 0$, $\tilde{\gamma} < 0$, it asserts that $x := 0$ belongs to $S(K, \tilde{\varepsilon})$ (with $\tilde{c}^T x > \tilde{\gamma} - \tilde{\varepsilon}$).

It is clear that no conclusion can be drawn with respect to the weak optimization problem. □

One cannot derive an oracle-polynomial time weak membership algorithm from a weak violation oracle, even if one knows a ball $S(a_0, r)$ contained in the convex body:

(4.5.4) $$\boxed{\text{WVIOL}, r, a_0} \rightarrow\!\!\!\!+\ \boxed{\text{WMEM}}\ .$$

Proof. Suppose, to the contrary, that one can derive such an algorithm. Let $n = 2$, $a_0 = (0,0)^T$, $r = 1$. We give the input $y = (8,0)^T$, $\delta = 1$ to the weak membership algorithm. In executing the algorithm, the weak violation oracle turns out to give, for any input $c = (c_1, c_2)^T$, $\gamma \in \mathbb{Q}$, $\varepsilon > 0$, answer (ii), with y' a vector satisfying: $y' \in \mathbb{Z}^2$, $c^T y' > \gamma$, $\|y' - z\|_\infty \leq 1$, where

$$z := \frac{1}{c_1 + c_2\sqrt{2}} \begin{pmatrix} \gamma \\ \gamma\sqrt{2} \end{pmatrix}.$$

(Note that the encoding length of y' is bounded by a polynomial in the encoding lengths of c, γ, and ε, since if $c = \frac{1}{q}d$ with $d \in \mathbb{Z}^2$, $q \in \mathbb{Z}$, then

$$\|y'\|_\infty \leq 1 + \|z\|_\infty = 1 + \frac{|\gamma|\sqrt{2}}{|c_1 + c_2\sqrt{2}|} = 1 + \frac{|\gamma|\sqrt{2}|c_1 - c_2\sqrt{2}|}{|c_1^2 - 2c_2^2|} =$$

$$= 1 + \left|\frac{q^2\gamma\sqrt{2}(c_1 - c_2\sqrt{2})}{d_1^2 - 2d_2^2}\right| \leq 1 + |q^2\gamma\sqrt{2}(c_1 - c_2\sqrt{2})|.)$$

Observe that $c^T z = \gamma$ and that $z \in C := \{(x_1, x_2)^T \mid x_2 = x_1\sqrt{2}\}$. It is clear that no conclusion can be drawn with respect to $(8,0)^T$. □

One cannot derive an oracle-polynomial time weak separation algorithm from a weak membership oracle, even if one knows a ball $S(a_0, r)$ contained in the convex body:

(4.5.5) $$\boxed{\text{WMEM}, r, a_0} \twoheadrightarrow \boxed{\text{WSEP}}.$$

Proof. Suppose, to the contrary, that one can derive such an algorithm. Let $n = 2$, $a_0 = (0,0)^T$, $r = 1$. We give the input $y = (0,3)^T$, $\delta = 0.2$ to the algorithm. In executing the algorithm, the weak membership oracle turns out to give the following answers for input $\tilde{y} = (\tilde{y}_1, \tilde{y}_2)^T, \tilde{\delta}$:

(i) if $\tilde{y}_2 \leq 1$, then it asserts that $\tilde{y} \in S(K, \tilde{\delta})$;
(ii) if $\tilde{y}_2 > 1$, then it asserts that $\tilde{y} \notin S(K, -\tilde{\delta})$.

Clearly, the conclusion that y belongs to $S(K, \delta)$ cannot be drawn. The algorithm neither can give a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ and $c^T x \leq c^T y + \delta$ for all x in $S(K, -\delta)$. Indeed, since the oracle has been asked only a finite number of times, there exists a vector d so that $d^T\tilde{y} \leq 1$ for all $\tilde{y}$ asked to the algorithm with $\tilde{y}_2 \leq 1$, and $d^T\tilde{y} > 1$ for all $\tilde{y}$ asked to the oracle with $\tilde{y}_2 > 1$, and so that $\|d\| = 1$ and $d \neq c$. Hence it can be the case that $K = \{x \mid d^T x \leq 1\} \cap S(0, R)$, for arbitrary large R. If R is large enough, there exists an x in $S(K, -\delta)$ with $c^T x > c^T y + \delta$, contradicting the output of the algorithm. □

One cannot derive an oracle-polynomial time weak validity algorithm from a weak separation oracle, even if one knows a ball $S(a_0, r)$ contained in the convex body:

(4.5.6) $$\boxed{\text{WSEP}, r, a_0} \twoheadrightarrow \boxed{\text{WVAL}}.$$

Proof. Suppose, to the contrary, that one can derive such an algorithm. Let $n = 2$, $a_0 = (0,0)^T$, $r = 1$. We give the input $c = (0,1)^T$, $\gamma = 3$, $\varepsilon = 1$ to the algorithm. In executing the algorithm, the weak separation oracle turns out to give the following answers, for input $y = (y_1, y_2)^T$:

(i) if $\|y\| \leq 1$, or if $|y_2| \leq 1$, $y_1 \leq 0$, it asserts that $y \in S(K, \delta)$;
(ii) otherwise, it gives some $\tilde{c} = (\tilde{c}_1, \tilde{c}_2)^T$ separating y from $S((0,0)^T, 1)$ with $\tilde{c}_1 > 0$.

It is clear that from these answers no conclusion with respect to the validity of $c^T x \leq \gamma + \varepsilon$ can be drawn, since K could be equal to $\{x \in \mathbb{R}^2 \mid \tilde{c}^T x \leq \|\tilde{c}\|$ for each $\tilde{c}$ that has been given as answer by the oracle $\} \cap S(0,R)$ for some large R, or equal to $\text{conv}\{y \in \mathbb{R}^2 \mid y$ has been asked to the oracle leading to output (i)$\}$. □

One cannot derive, in oracle-polynomial time, a vector a_0 and a rational $r' > 0$ such that the convex body K contains $S(a_0, r')$, from a weak separation oracle for K, even if one knows some inner radius r for K in advance:

(4.5.7) $$\boxed{\text{WSEP}, r} \rightarrow\!\!\!\!+ \; \boxed{r, a_0}\,.$$

Proof. Let $n = 1$, $r = 1$, and suppose the weak separation oracle answers, for any $y \in \mathbb{Q}$, $\delta > 0$, that $x < y$ for all x in K. We cannot conclude with any a_0 belonging to K. □

Similarly for a weak violation oracle:

(4.5.8) $$\boxed{\text{WVIOL}, r} \rightarrow\!\!\!\!+ \; \boxed{r, a_0}\,.$$

Proof. Let $n = 2$, $r = 1$, and suppose the weak violation oracle gives, for input $c = (c_1, c_2)^T \in \mathbb{Q}^2$, $\gamma \in \mathbb{Q}$, $\varepsilon > 0$, a vector y in $S(K, \varepsilon)$ with $c^T y > \gamma + \varepsilon$, where y is some rational vector in $S(z, \varepsilon)$, z being the point of intersection of the lines $\{x \mid c^T x = \gamma + 2\varepsilon\}$ and $\{x = (x_1, x_2)^T \mid x_2 = x_1\sqrt{2}\}$. Even if one knows that K contains a ball of radius r, one cannot determine, from a finite number of answers, a ball $S(a_0, r')$ contained in K. □

Note that (4.5.7) and (4.5.8) imply

(4.5.9) $$\boxed{\text{WSEP}, r} \rightarrow\!\!\!\!+ \; \boxed{\text{WOPT}}$$

and

(4.5.10) $$\boxed{\text{WVIOL}, r} \rightarrow\!\!\!\!+ \; \boxed{\text{WOPT}}$$

because of (4.4.8).

Finally, we show that one cannot derive, in oracle polynomial time, a rational number $r > 0$ such that the convex body K contains a ball of radius r, from a weak separation oracle for K, even if one knows a radius R with $K \subseteq S(0,R)$:

(4.5.11) $$\boxed{\text{WSEP}, R} \not\to \boxed{r}\ .$$

Proof. Let $n = 1$, $R = 2$, and suppose the oracle turns out to answer, for any input $y \in \mathbb{Q}$, $\delta > 0$:

(i) if $y < \sqrt{2}$, that $x > y - \delta$ for all x in K,
(ii) if $y > \sqrt{2}$, that $x < y + \delta$ for all x in K.

It is clear that one cannot make any conclusion about the inner radius of K. □

*4.6 Further Algorithmic Problems for Convex Bodies

We shall now investigate some further basic questions of convex geometry from an algorithmic point of view. Our main tool will be an algorithmic version of the Löwner-John theorem (3.1.9). Results of this type have also been obtained by GOFFIN (1984).

(4.6.1) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-bounded convex body $(K; n, R, r)$ given by a weak separation oracle, finds an ellipsoid $E(A,a)$ such that*

$$E\left(\tfrac{1}{n(n+1)^2}A, a\right) \subseteq K \subseteq E(A,a).$$

Proof. We shall only describe the underlying simple geometric idea of the algorithm, supposing that all calculations with real numbers can be carried out exactly. One could take the necessary rounding into account just like in the shallow-cut ellipsoid method and this would not lead to substantial additional difficulties.

We derive a shallow separation oracle from the given weak separation oracle as follows. Let $E = E(A,a)$ be any ellipsoid. Determine the axes of E; more precisely, compute a system of orthonormal eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Then $a \pm \sqrt{\lambda_i} v_i$, $i = 1, \ldots, n$ are the endpoints of the axes. Run the separation oracle for each of the points $a \pm \frac{1}{n+1}\sqrt{\lambda_i}\, e_i$. If we obtain a separating hyperplane for one of these points, this yields a shallow cut for E. If all these points turn out to belong to K then we declare E tough.

Let us remark that for a tough ellipsoid $E(A,a)$, the ellipsoid $E(\frac{1}{(n+1)^2}A, a)$ is not necessarily contained in K. However, the smaller concentrical ellipsoid $E(\frac{1}{n(n+1)^2}A, a)$ is contained in the convex hull of the points $a \pm \frac{1}{n+1}\sqrt{\lambda_i}\, v_i$, $i = 1, \ldots, n$, and therefore in K.

Using this shallow separation oracle we run a shallow-cut ellipsoid algorithm with $\beta = \frac{1}{n+1}$, $\varepsilon = (\frac{r}{n})^n$ and the given R. This gives us an ellipsoid $E(A,a) \supseteq K$. Since $\varepsilon < \text{vol}(K)$, the second possibility of Theorem (3.3.3) is ruled out, and hence $E(A,a)$ is tough. By the remark above, this implies that $E(\frac{1}{n(n+1)^2}A, a) \subseteq K$. □

(4.6.2) Remark. In the proof above we called the weak separation oracle for $2n$ points on the surface of the ellipsoid $E((n+1)^{-2}A, a)$. By calling it for more than $2n$ but still polynomially many points, we could find in polynomial time an ellipsoid $E(A, a)$ such that

$$E(cn^{-3}A, a) \subseteq K \subseteq E(A, a)$$

for every fixed positive c and n large enough. However the degree of the polynomial will depend on c.

By the Löwner-John theorem there exists an ellipsoid $E(A, a)$ with $E(n^{-2}A, a) \subseteq K \subseteq E(A, a)$. We do not know how much the factor n^{-3} obtained above can be improved algorithmically if the convex body is given by a weak separation oracle. □

For special classes of convex bodies Theorem (4.6.1) can be improved. A set $K \subseteq \mathbb{R}^n$ is called **centrally symmetric with respect to** $a \in \mathbb{R}^n$ if $x \in K$ implies $2a - x \in K$. K is called just **centrally symmetric** if K is centrally symmetric with respect to the origin.

(4.6.3) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-bounded, convex body $(K; n, R, r)$ centrally symmetric with respect to $a \in \mathbb{Q}^n$ that is given by a weak separation oracle, finds an ellipsoid $E(A, a)$ with*

$$E\left(\tfrac{1}{n(n+1)}A, a\right) \subseteq K \subseteq E(A, a).$$

□

(4.6.4) Theorem. *There exists a polynomial time algorithm that, for any well-bounded convex body $(P; n, R, r)$ given as the solution set of a system of linear inequalities $Bx \leq b$, where $B \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, finds an ellipsoid $E(A, a)$ with*

$$E\left(\tfrac{1}{(n+1)^2}A, a\right) \subseteq P \subseteq E(A, a).$$

□

(4.6.5) Theorem. *There exists a polynomial time algorithm that, for any well-bounded, centrally symmetric, convex body $(P; n, R, r)$ given as the solution set of a system of linear inequalities $-b \leq Bx \leq b$, $B \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}$, finds an ellipsoid $E(A, 0)$ with*

$$E\left(\tfrac{1}{n+1}A, 0\right) \subseteq P \subseteq E(A, 0).$$

□

(4.6.6) Remark. Jordan proved – cf. JOHN (1948) – that for a convex body K, centrally symmetric with respect to a, there exists an ellipsoid $E(A,a)$ such that

$$E\left(\tfrac{1}{n}A,a\right) \subseteq K \subseteq E(A,a).$$

Again we do not know whether the factor $(n(n+1))^{-1}$ obtained in Theorem (4.6.3) can be improved algorithmically to a factor closer to the theoretically best possible factor $1/n$ in Jordan's theorem.

However, if our convex body is a polytope given as the solution set of a system of linear inequalities, then asymptotically the factors given in the Löwner-John theorem and in Jordan's theorem can be achieved algorithmically, as the last two theorems show. □

We sketch the proof of Theorem (4.6.5) which contains the essential ideas needed in the proofs of Theorems (4.6.3) and (4.6.4).

Proof of Theorem (4.6.5). The idea of the proof is to use parallel cuts instead of shallow cuts. We construct a sequence of ellipsoids $E_k := E(A_k,0)$, including P, as follows. Let E_0 be equal to $S(0,R)$. Suppose in the k-th iteration we have constructed E_k. Then we check whether $E(\frac{1}{\sqrt{n+1}}A_k,0)$ is contained in P – cf. Remark (3.3.4). If not, let $y \in E(\frac{1}{\sqrt{n+1}}A_k,0) \setminus P$, and let $a^Tx \leq \alpha$ be an inequality in the system defining P violated by y. Then $a^Tx \geq -\alpha$ is also a valid inequality for P. Choose as the next ellipsoid E_{k+1} the Löwner-John ellipsoid of $E_k \cap \{x \in \mathbb{R}^n \mid -\alpha \leq a^Tx \leq \alpha\}$ – see (3.1.18) – (3.1.20). Then we have

$$\frac{\mathrm{vol}(E_{k+1})}{\mathrm{vol}(E_k)} \leq e^{-1/(5n^2)}.$$

This inequality guarantees that the algorithm terminates in polynomial time. □

(4.6.7) Remark. In Theorems (4.6.4) and (4.6.5) we could – alternatively – consider polytopes given as the convex hull of finitely many points and reach the same conclusions. In the case of Theorem (4.6.5) this follows easily by polarity. In the case of Theorem (4.6.4) an algorithm by LENSTRA (1983) can be used. □

The algorithms above can be viewed as methods to make a convex body round in the following sense.

(4.6.8) Corollary. *(a) There exists an oracle-polynomial time algorithm that, for any well-bounded, convex body $(K;n,R,r)$ given by a weak separation oracle, finds an affine transformation $Q : x \mapsto Tx+t$ $(T \in \mathbb{Q}^{n\times n}, t \in \mathbb{Q}^n)$ such that*

$$S\left(0,\tfrac{1}{\sqrt{n(n+1)}}\right) \subseteq Q(K) \subseteq S(0,1).$$

If, in addition, K is centrally symmetric then the algorithm achieves

$$S\left(0,\tfrac{1}{\sqrt{n(n+1)}}\right) \subseteq Q(K) \subseteq S(0,1).$$

(b) There exists a polynomial time algorithm that, for any well-bounded, convex body $(P; n, R, r)$ given as the solution set of a system of linear inequalities, finds an affine transformation Q such that

$$S\left(0, \tfrac{1}{n+1}\right) \subseteq Q(K) \subseteq S(0,1).$$

If, in addition, P is a centrally symmetric polytope, the algorithm obtains

$$S\left(0, \tfrac{1}{\sqrt{n+1}}\right) \subseteq Q(K) \subseteq S(0,1).$$

□

These results have a further interpretation for normed linear spaces. If we speak about a given norm on $\mathbb{R}^n$, we shall assume that there is a weak norm oracle (i. e., an oracle that, for every $x \in \mathbb{Q}^n$ and every rational $\varepsilon > 0$, produces a rational number η such that $|N(x) - \eta| < \varepsilon$) and that two rational numbers $c_1, c_2 > 0$ are given such that for all $x \in \mathbb{R}^n$,

$$c_1\|x\| \leq N(x) \leq c_2\|x\|.$$

The next corollary asserts that for every given norm N on $\mathbb{R}^n$ an ellipsoidal norm can be computed in polynomial time that is "close" to N.

(4.6.9) Corollary. *There exists an oracle-polynomial time algorithm that, for any given norm N on $\mathbb{R}^n$, computes a linear transformation Q of $\mathbb{R}^n$ such that for all $x \in \mathbb{R}^n$,*

$$\|Qx\| \leq N(x) \leq \sqrt{n(n+1)}\,\|Qx\|.$$

Proof. Let $K = \{x \in \mathbb{R}^n \mid N(x) \leq 1\}$. Then K is a centrally symmetric convex body and

$$S\left(0, \tfrac{1}{c_2}\right) \subseteq K \subseteq S\left(0, \tfrac{1}{c_1}\right).$$

Further, we have a trivial weak membership algorithm for K. Hence by Theorems (4.3.2) and (4.4.4) we can design a weak separation algorithm for K. So by Theorem (4.6.8) (a), we can compute in oracle-polynomial time a linear transformation Q of $\mathbb{R}^n$ such that

$$S\left(0, \frac{1}{\sqrt{n(n+1)}}\right) \subseteq Q(K) \subseteq S(0,1).$$

Then, if $x \in K$, then $Qx \in Q(K)$ and so $\|Qx\| \leq 1$. Since for any nonzero $x \in \mathbb{R}^n$, $\frac{1}{N(x)}x \in K$, we have $1 \geq \|Q(\frac{1}{N(x)}x)\| = \frac{1}{N(x)}\|Q(x)\|$, and so $\|Qx\| \leq N(x)$. Similarly we obtain that $N(x) \leq \sqrt{n(n+1)}\,\|Q(x)\|$. □

(4.6.10) Remarks.

(a) As described above, the algorithm to compute the ellipsoidal norm $\|Qx\|$ involves three ellipsoid algorithms inside each other (two to obtain separation from membership and one more to calculate Q). With a little care one could combine these into a single shallow-cut ellipsoid method.

(b) The upper bound c_2 for $N(x)/\|x\|$ need not be given in advance. One can easily compute it from the norm oracle. In fact,

$$c_2 := \lceil (N(e_1) + \ldots + N(e_n))^{\frac{1}{2}} \rceil$$

is a suitable choice. On the other hand, if $n \geq 3$ then no lower bound at all can be computed for $N(x)/\|x\|$ from an oracle to evaluate $N(x)$. For $n = 2$,

$$c_1 := \tfrac{1}{2} \max\left\{ N\left(e_1 \pm \tfrac{N(e_1)}{N(e_2)} e_2\right), N\left(e_2 \pm \tfrac{N(e_2)}{N(e_1)} e_1\right) \right\}$$

is a suitable choice. We shall not, however, go into the details of this rather special technical question here. □

Instead of finding the Löwner-John ellipsoid of a convex body K it may be more natural to look for the smallest ball including K and for a largest ball contained in K. These problems are related to two further problems, namely, to determining the width and the diameter of K.

For a set $K \subseteq \mathbb{R}^n$ we denote by $\overline{R}(K)$ and $\overline{r}(K)$ the radii of the smallest circumscribed ball and a largest inscribed ball, respectively. The following two inequalities are well known from classical geometry.

(4.6.11) Theorem. *For every convex body $K \subseteq \mathbb{R}^n$,*

$$\tfrac{1}{2} \operatorname{diam}(K) \leq \overline{R}(K) \leq \frac{1}{\sqrt{2+\frac{2}{n}}} \operatorname{diam}(K),$$

and

$$\tfrac{1}{n+1} \operatorname{width}(K) \leq \overline{r}(K) \leq \tfrac{1}{2} \operatorname{width}(K).$$

□

Algorithmically the following can be achieved.

(4.6.12) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-bounded, convex body $(K; n, R, r)$ given by a weak separation oracle, finds two points $x, y \in K$ and a ball $S(a, \rho)$ including K such that*

$$\rho \leq \tfrac{1}{2}\sqrt{n}\|x - y\|.$$

(Note that obviously $\rho \geq \frac{1}{2}\|x - y\|$.)

Proof. For each $1 \leq i \leq n$, maximize the objective functions $e_i^T x$ and $-e_i^T x$ over K. Let x_i and x_i' be the optimum solutions respectively, and suppose, e. g., that $\|x_1 - x_1'\| \geq \|x_i - x_i'\|$ $(1 \leq i \leq n)$. Set $x := x_1$ and $y := x_1'$. Let

$$a := \sum_{i=1}^{n} e_i^T \left(\frac{x_i + x_i'}{2}\right) e_i ,$$

$$\rho := \tfrac{1}{2}\sqrt{n}\,\|x_1 - x_1'\|.$$

Then $K \subseteq S(a, \rho)$, which is easy to check. □

(4.6.13) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-bounded, convex body $(K; n, R, r)$ given by a weak separation oracle, finds two parallel hyperplanes $c^T x = \gamma_1$, $c^T x = \gamma_2$ and a ball $S(a, \rho')$ contained in K such that for all $x \in K$, $\gamma_1 \leq c^T x \leq \gamma_2$ and*

$$\rho' = \frac{\gamma_2 - \gamma_1}{2(n+1)\sqrt{n}\|c\|}.$$

Proof. By Theorem (4.6.1), we can find in oracle-polynomial time an ellipsoid $E(A, a)$ such that

$$E\left(\tfrac{1}{n(n+1)^2}A, a\right) \subseteq K \subseteq E(A, a).$$

Let c be an eigenvector of A belonging to its smallest eigenvalue λ. (Note that c and λ can be computed in polynomial time using well-known techniques from numerical analysis.) Let

$$\rho' := \frac{\lambda}{(n+1)\sqrt{n}}.$$

Then $S(a, \rho') \subseteq K$. Furthermore, let

$$\gamma_1 = c^T a - \lambda\|c\|$$

$$\gamma_2 = c^T a + \lambda\|c\|.$$

Then obviously the requirements of the theorem are satisfied. □

There are many other geometric parameters of convex bodies that are not well studied from an algorithmic point of view. For example, from Theorem (4.6.1) we get an estimate for the volume of a convex body K with relative error approximately $n^{\frac{3}{2}n}$. This error is not as bad as it looks in the sense that one can prove that no oracle-polynomial time algorithm can compute the volume of every well-bounded, convex body given by (say) a separation oracle with relative error less than $(cn)^{n/2}$ where c is some constant – see ELEKES (1986) and BÁRÁNY and FÜREDI (1986). But it is not known, for example, whether the volume of a polytope given as the convex hull of a set of vectors is computable in polynomial time. Further open algorithmic problems are the determination of the center of gravity and the surface area etc. of a convex body.

*4.7 Operations on Convex Bodies

There are many operations that make new convex bodies from others (intersection, sum, projection, convex hull, polar etc.). In this section we shall study the preservation of algorithmic properties by some of these operations. We restrict our investigation here to well-bounded convex bodies.

The Sum

The **sum** of two convex sets $K_1, K_2 \subseteq \mathbb{R}^n$ is defined as

$$K_1 + K_2 := \{x_1 + x_2 \in \mathbb{R}^n \mid x_1 \in K_1, x_2 \in K_2\}.$$

The difference $K_1 - K_2$ is defined analogously. If $(K_1; n, R_1, r_1)$ and $(K_2; n, R_2, r_2)$ are well-bounded convex bodies then

$$(K_1 + K_2; n, R_1 + R_2, r_1 + r_2)$$

is also a well-bounded convex body. Moreover, if we have weak optimization oracles for K_1 and K_2 then the weak optimization problem for $K_1 + K_2$ can be solved easily. In fact, given $c \in \mathbb{Q}^n$ and a rational $\varepsilon > 0$ (we may assume that $\|c\|_\infty = 1$), then set $\varepsilon_i = \min\{r_i, \varepsilon r_i/(8nR_i)\}$, $i = 1, 2$. We call the weak optimization oracles for K_i with input c and ε_i. This gives us vectors y_1 and y_2 such that $y_i \in S(K_i, \varepsilon_i)$ and for all $x_i \in S(K_i, -\varepsilon_i)$ we have $c^T x_i \leq c^T y_i + \varepsilon_i$. Hence by Lemma (3.2.25) we have that for all $x_i \in K_i$, $c^T x_i \leq c^T y_i + \varepsilon_i + 2R_i\varepsilon_i\sqrt{n}/r_i \leq c^T y_i + \varepsilon/2$. We claim that $y := y_1 + y_2$ solves the weak optimization problem for $K_1 + K_2$ and the given input. For, trivially $y \in S(K_1, \varepsilon_1) + S(K_2, \varepsilon_1) \subseteq S(K, \varepsilon)$, and moreover, letting $x \in K_1 + K_2$, then there are $x_1 \in K_1$ and $x_2 \in K_2$ with $x = x_1 + x_2$ and so

$$c^T x = c^T x_1 + c^T x_2 \leq c^T y_1 + \varepsilon/2 + c^T y_2 + \varepsilon/2 = c^T y + \varepsilon.$$

In view of the results of the previous section, this implies that if we have a weak violation (separation) oracle for K_1 and K_2, then the weak violation (weak separation) problem for $K_1 + K_2$ can be solved in oracle-polynomial time.

It is not difficult to show that weak validity oracles for K_1 and K_2 yield an oracle-polynomial weak validity algorithm for $K_1 + K_2$. Namely, given a rational inequality $c^T x \leq \gamma$ and a rational $\varepsilon > 0$, we can use binary search to find rational γ_1 and γ_2 such that $c^T x_i \leq \gamma_i$ is valid for all $x_i \in K_i$ but there exist $y_i \in S(K_i, \varepsilon/2)$ such that $c^T y_i > \gamma_i - \varepsilon/2$. Now if $\gamma \geq \gamma_1 + \gamma_2$ then $c^T x \leq \gamma$ is valid for all $x \in K$. If $\gamma < \gamma_1 + \gamma_2$ then $y_1 + y_2 \in S(K_1 + K_2, \varepsilon)$ satisfies $c^T(y_1 + y_2) > \gamma_1 + \gamma_2 - \varepsilon \geq \gamma - \varepsilon$.

It is impossible to derive an oracle-polynomial algorithm for the weak membership problem for $K_1 + K_2$, given weak membership oracles for well-bounded convex bodies K_1 and K_2. This follows by the same arguments as given in Section 4.1. For centered convex bodies, however, such a derivation can be made using Yudin and Nemirovskiĭ's theorem.

The Convex Hull of the Union

Now we study $\text{conv}(K_1 \cup K_2)$. The results will be very similar to the results concerning $K_1 + K_2$. Let $(K_i; n, R_i, r_i)$, $i = 1, 2$, be two well-bounded convex bodies. Then

$$(\text{conv}(K_1 \cup K_2); n, \max\{R_1, R_2\}, \max\{r_1, r_2\})$$

is a well-bounded convex body. If we have weak optimization (validity) oracles for K_1 and K_2, then we can solve the weak optimization (validity) problem for $\text{conv}(K_1 \cup K_2)$ trivially. Hence by the results of the previous sections the same holds for weak separation and violation oracles. To solve the weak membership problem for $\text{conv}(K_1 \cup K_2)$ using weak (or even strong) membership oracles for K_1 and K_2 is again impossible in oracle-polynomial time.

The Intersection

The problem with the intersection of two convex bodies K_1 and K_2 is that $K_1 \cap K_2$ need not be a convex body at all, and even if it is, we may not be able to compute a radius of a ball inside $K_1 \cap K_2$ in polynomial time. So let us assume that we also know a rational number r_3 such that $K_1 \cap K_2$ contains a ball with radius r_3. Set $R_3 = \min\{R_1, R_2\}$. Then

$$(K_1 \cap K_2; n, R_3, r_3)$$

is a well-bounded convex body.

First we show that given weak membership oracles for K_1 and K_2 we can solve the weak membership problem for $K_1 \cap K_2$ in oracle-polynomial time quite simply. Let $y \in \mathbb{Q}^n$ and $\delta \in \mathbb{Q}$, $\delta > 0$ be given. Let $\delta' := \delta r_3/(2R_3)$. Call the weak membership oracles for K_1 and K_2 with input y and δ'. If they conclude that $y \notin S(K_i, -\delta')$ for $i = 1$ or 2 then clearly $y \notin S(K_1 \cap K_2, -\delta)$. Suppose the oracles conclude that $y \in S(K_i, \delta')$ for $i = 1, 2$. Then we claim that $y \in S(K_1 \cap K_2, \delta)$. For, let $S(a_0, r_3) \subseteq K_1 \cap K_2$. We are going to show that

$$z := \frac{\delta'}{r_3 + \delta'} a_0 + \frac{r_3}{r_3 + \delta'} y \in K_1 \cap K_2.$$

We first prove that $z \in K_1$. Since $y \in S(K_1, \delta')$ there exists a point $y_1 \in K_1$ such that $\|y - y_1\| \leq \delta'$. Then

$$a_0 + \frac{r_3}{\delta'}(y - y_1) \in S(a_0, r_3) \subseteq K_1.$$

And so by the convexity of K_1

$$z = \frac{\delta'}{r_3 + \delta'}\left(a_0 + \frac{r_3}{\delta'}(y - y_1)\right) + \frac{r_3}{r_3 + \delta'} y_1$$

is in K_1. Similarly $z \in K_2$. So $z \in K_1 \cap K_2$. Since $\|z - y\| = \delta' \|z - a_0\|/r_3 \leq 2\delta' R_3/r_3 = \delta$, this implies that $y \in S(K_1 \cap K_2, \delta)$.

The same argument shows that if we have weak separation oracles for K_1 and K_2 we can solve the weak separation problem for $K_1 \cap K_2$ in oracle-polynomial time. By the results of the previous sections this implies that the same holds for weak optimization, violation and validity.

The somewhat disturbing hypothesis that $K_1 \cap K_2$ contains a ball with radius r_3 can be checked in the following sense.

(4.7.1) Theorem. *There exists an oracle-polynomial time algorithm that, for any two well-bounded, convex bodies $(K_i; n, R_i, r_i)$, $i = 1, 2$, given by weak separation oracles and for any given rational number $r_3 > 0$, either*

(i) *asserts that $K_1 \cap K_2$ contains a ball with radius r_3, or*
(ii) *finds a vector $c \in \mathbb{Q}^n$ with $\|c\|_\infty = 1$ and a number $\gamma \in \mathbb{Q}$ such that $c^T x \leq \gamma$ for all $x \in S(K_1, -\varepsilon)$ and $c^T x \geq \gamma$ for all $x \in S(K_2, -\varepsilon)$ where $\varepsilon = 9nr_3(R_1/r_1 + R_2/r_2)$.*

(I. e., we find a hyperplane almost separating K_1 from K_2.)

Proof. Let us consider the convex body $(K_1 - K_2; n, R_1 + R_2, r_1 + r_2)$. We can solve the weak separation problem for $K_1 - K_2$ in oracle-polynomial time by the first part of this section. Solve the weak separation problem for $K_1 - K_2$ $2n$ times with inputs $y = \pm\frac{\varepsilon}{4}e_i$ and $\delta = r_3$, for $i = 1, \ldots, n$.

Case 1. We find that for all $i = 1, \ldots, n$ and for all choices of the sign the vector $\pm\varepsilon e_i/4$ belongs to $S(K_1 - K_2, r_3)$.

Thus $S(0, \varepsilon/(4\sqrt{n})) \subseteq S(K_1 - K_2, r_3)$. We claim that $S(K_1, -r_3) \cap S(K_2, -r_3) \neq \emptyset$. This is clearly equivalent to the conclusion that $K_1 \cap K_2$ contains a ball with radius r_3.

Suppose by contradiction that $S(K_1, -r_3) \cap S(K_2, -r_3) = \emptyset$. Then $S(K_1, -r_3)$ and $S(K_2, -r_3)$ can be separated by a hyperplane $c_0^T x = \gamma_0$ where $\|c_0\|_\infty = 1$. By Lemma (3.2.35) $c_0^T x \leq \gamma_0 + 2R_1 r_3 \sqrt{n}/r_1$ for all $x \in K_1$ and $c_0^T x \geq \gamma_0 - 2R_2 r_3 \sqrt{n}/r_2$ for all $x \in K_2$. But we know that $\varepsilon/(4\sqrt{n}\|c_0\|)c_0 \in K_1 - K_2$ and so $\varepsilon/(4\sqrt{n}\|c_0\|)c_0 = y_1 - y_2$ for certain $y_i \in K_i$, $i = 1, 2$. Hence $\varepsilon/(4\sqrt{n}\|c_0\|)c_0^T c_0 = c_0^T y_1 - c_0^T y_2 \leq 2r_3\sqrt{n}$ $(R_1/r_1 + R_2/r_2) < \varepsilon/(4\sqrt{n})$. On the other hand:

$$\frac{\varepsilon}{4\sqrt{n}} \frac{c_0^T c_0}{\|c_0\|} = \frac{\varepsilon}{4\sqrt{n}} \|c_0\| \geq \frac{\varepsilon}{4\sqrt{n}},$$

which is a contradiction.

Case 2. For at least one of $i \in \{1, \ldots, n\}$ and at least one choice of the sign we obtain an almost separating hyperplane, i. e., we obtain a vector $c \in \mathbb{Q}^n$, $\|c\|_\infty = 1$ such that (say) $c^T(\varepsilon e_i/4) \geq c^T x - r_3$ for all $x \in S(K_1 - K_2, -r_3)$. Compute a number γ such that $|\gamma - \max\{c^T x \mid x \in K_1\}| \leq \varepsilon/4$. Then trivially $c^T x \leq \gamma + \varepsilon/4 - \varepsilon$ for all $x \in S(K_1, -\varepsilon)$. On the other hand, $c^T x \leq c^T(\varepsilon e_1/4) + r_3 < \varepsilon/2$ for all $x \in S(K_1 - K_2, -r_3)$ and hence by Lemma (3.2.35)

$$c^T x \leq \frac{\varepsilon}{2} + 2\sqrt{n}\,\frac{R_1 + R_2}{r_1 + r_2} r_3 \leq \frac{\varepsilon}{2} + 2n\left(\frac{R_1}{r_2} + \frac{R_2}{r_2}\right) r_3 \leq \frac{3}{4}\varepsilon$$

for all $x \in K_1 - K_2$.

Let $x_2 \in K_2$ be arbitrary and let x_1 be a vector in K_1 maximizing $c^T x$ over K_1. Then $x_1 - x_2 \in K_1 - K_2$ and so

$$c^T(x_1 - x_2) \leq \frac{3}{4}\varepsilon.$$

Hence

$$c^T x_2 \geq c^T x_1 - \frac{3}{4}\varepsilon \geq \gamma - \frac{\varepsilon}{4} - \frac{3}{4}\varepsilon \geq \gamma - \varepsilon.$$

Thus for all $x_2 \in S(K_2, -\varepsilon)$ we have

$$c^T x_2 \geq \gamma - \varepsilon + \varepsilon = \gamma.$$

□

By combining Theorem (4.7.1) with the observations made before, we can conclude that we can either weakly optimize over $K_1 \cap K_2$ or weakly separate K_1 and K_2.

(4.7.2) Corollary. *There exists an oracle-polynomial time algorithm that, for any two well-bounded convex bodies $(K_i; n, R_i, r_i)$, $i = 1, 2$, given by weak separation oracles, for any vector $c \in \mathbb{Q}^n$, and any rational number $\varepsilon > 0$, either*

(i) *finds a vector $y \in \mathbb{Q}^n$ such that $y \in S(K_1 \cap K_2, \varepsilon)$ and $c^T x \leq c^T y + \varepsilon$ for all $x \in S(K_1 \cap K_2, -\varepsilon)$, or*

(ii) *finds a vector $d \in \mathbb{Q}^n$, $\|d\|_\infty = 1$, and a number $\gamma \in \mathbb{Q}$ such that*

$$d^T x \leq \gamma \quad \text{for all } x \in S(K_1, -\varepsilon)$$

and

$$d^T x \geq \gamma \quad \text{for all } x \in S(K_2, -\varepsilon).$$

□

Polars, Blockers, Antiblockers

Recall that in Section 0.1 we have defined the polar, blocker and antiblocker of a set $K \subseteq \mathbb{R}^n$ as follows:

$$\begin{aligned} K^* &:= \{y \in \mathbb{R}^n \mid y^T x \leq 1 \text{ for all } x \in K\}, \\ \mathrm{bl}(K) &:= \{y \in \mathbb{R}^n_+ \mid y^T x \geq 1 \text{ for all } x \in K\}, \\ \mathrm{abl}(K) &:= K^* \cap \mathbb{R}^n_+. \end{aligned}$$

Also recall the notions "up-monotone" and "down-monotone" in $\mathbb{R}^n_+$.

We have seen in Section 4.4 that, if K is a 0-centered convex body, then K^* is also a 0-centered convex body. Furthermore, the separation, optimization etc. problem for such K and K^* are equivalent. We are going to derive similar results for the blocker and antiblocker.

It is easy to see that if $(K; n, R, r)$ is a well-bounded convex body down-monotone in $\mathbb{R}^n_+$ then $(\mathrm{abl}(K); n, 1/r, 1/nR)$ is a well-bounded convex body down-monotone in $\mathbb{R}^n_+$. Furthermore, along the lines of the proof of Lemma (4.4.1) one can show that if K is given by a weak violation oracle, then the weak separation problem for $\mathrm{abl}(K)$ can be solved in oracle-polynomial time. From this we can conclude that the separation, optimization etc. problem for K and $\mathrm{abl}(K)$ are polynomially equivalent.

It is clear that similar results can be worked out for the blocker of a set up-monotone in $\mathbb{R}^n_+$. Of course, the machinery developed so far does not work directly, since nonempty up-monotone sets are not bounded. One could get around this difficulty by intersecting the set and its blocker with large balls. We do not go into the details here. For polyhedra the results are elaborated in Chapter 6.

Chapter 5

Diophantine Approximation and Basis Reduction

As mentioned in Chapter 1, combinatorial optimization problems can usually be formulated as linear programs with integrality constraints. The geometric notion reflecting the main issues in linear programming is convexity, and we have discussed the main algorithmic problems on convex sets in the previous chapters. It turns out that it is also useful to formulate integrality constraints in a geometric way. This leads us to "lattices of points". Such lattices have been studied (mostly from a nonalgorithmic point of view) in the "geometry of numbers"; their main application has been the theory of simultaneous diophantine approximation, i. e., the problem of approximating a set of real numbers by rational numbers with a common small denominator. We offer an algorithmic study of lattices and diophantine approximation.

The main result derived here is an algorithm to approximate a set of numbers by rational numbers with a common small denominator. We have seen previously that the ellipsoid method gives only approximate solutions in various applications; often the exact solutions can be obtained by appropriate "rounding". Thus, the problem of "rounding", that is, of approximating real numbers by rational numbers with small denominator, arises naturally. It is still somewhat surprising that often a straightforward rounding is not sufficient, and quite involved techniques of simultaneous diophantine approximation are needed. Applications of this sort will occur in Chapter 6.

In the first section of this chapter we give a quick survey of the classical algorithm of expanding a number into a continued fraction. This can be used to find, for a given real number, the best rational approximation with a denominator bounded by a prescribed integer (this algorithm was referred to in Section 1.3). In Section 5.2 we consider the problem of simultaneous diophantine approximation. Existence theorems for such rationals and good estimates on their denominators are well known in number theory; the important new point here is that there exists a polynomial time algorithm to solve this problem – at least in a sense (LENSTRA, LENSTRA and LOVÁSZ (1982)).

In Section 5.3 we describe the algorithm of Lenstra, Lenstra, and Lovász. Since Minkowski, a main tool in the study of diophantine approximation problems is the geometry of numbers, the theory that investigates the intersections of point lattices with convex sets. The algorithm described here is based on finding a "reduced basis" in an appropriate lattice.

In Section 5.4 we discuss some algorithmic problems concerning lattices. It seems that the algorithmic theory of lattices is less developed than the algorithmic

theory of convex sets, even though some features are quite analogous. This section is only a first step in this direction. It will not be needed in the rest of the book.

Let us conclude this introduction with a general remark. There are two branches of mathematics dealing with lattice points in various convex bodies: integer linear programming and the geometry of numbers. There is, however, surprisingly little connection between these two fields. It seems that the methods of either one of them have little use for the problems in the other field. We certainly cannot claim to have established a connection; but possibly Lenstra's work and the other algorithms in this book lay the first planks to build a bridge between these two flourishing branches of mathematics.

5.1 Continued Fractions

It may be instructional to survey first what may be considered as a 1- or 2-dimensional version of the basis reduction algorithm to be discussed later – the technique of continued fractions. The classical books on continued fractions are PERRON (1913) and KHINTCHINE (1956) but this method is treated also in many books on number theory (e. g., NIVEN and ZUCKERMAN (1980)) or approximation theory (e. g., NONWEILER (1984)). Its main algorithmic application will be the solution of the following problem.

(5.1.1) Best Approximation Problem. *Given a rational number α and a positive integer N, find a rational number α' with denominator at most N such that $|\alpha - \alpha'|$ is minimum.*

This problem can be formulated as a decision problem as follows.

(5.1.2) Diophantine Approximation Problem. *Given a rational number α, a positive integer N and a positive rational number ε, decide whether there exists a rational number α' with denominator at most N such that $|\alpha - \alpha'| < \varepsilon$.*

The following classical result of DIRICHLET (1842) gives a sufficient condition for the solvability of (5.1.2).

(5.1.3) Theorem. *Given a real number α and $0 < \varepsilon < 1$, there exist integers p and q such that $1 \leq q \leq 1/\varepsilon$ and $|\alpha - p/q| < \varepsilon/q$.*

Proof. Consider a circle with circumference 1. Let $k := \lfloor 1/\varepsilon \rfloor$. Starting from a point a_0 on the circle move clockwise distances $\alpha, 2\alpha, \ldots, k\alpha$ on the circle to get points $a_0, a_1, a_2, \ldots, a_k$. Since we have $k+1$ points, two of these, say a_i and a_j, $i < j$, have distance $d \leq 1/(k+1)$ (measured on the circle). This means that $j\alpha \pm d - i\alpha$ is an integer, say p. Thus for $q := j - i$ we have $|q\alpha - p| = d \leq 1/(k+1) < \varepsilon$ and $q \leq k \leq 1/\varepsilon$. □

This application of the pigeon hole principle is "constructive", but it does not yield a polynomial algorithm to find p and q. In fact, the running time is polynomial in $1/\varepsilon$ but not in $\langle\varepsilon\rangle$, i. e., the pigeon hole principle gives a fully polynomial approximation scheme. Continued fractions, which we describe now, give an algorithm polynomial in the encoding length.

Let $a_0, a_1, a_2, \ldots, a_j$ be integers, all positive, except perhaps a_0. An expression of the following type

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots \; + \cfrac{1}{a_{j-1} + \cfrac{1}{a_j}}}}} \tag{5.1.4}$$

is called a (finite) **continued fraction**. It is usual to abbreviate the expression (5.1.4) by $\langle a_0, a_1, \ldots, a_j\rangle$ or $a_0 + \underset{\frown}{1}\big/a_1 + \underset{\frown}{1}\big/a_2 + \ldots + \underset{\frown}{1}\big/a_j$. Since $\langle\cdot\rangle$ denotes the encoding length in our book we will use the second notation. Clearly, $r = a_0 + \underset{\frown}{1}\big/a_1 + \underset{\frown}{1}\big/a_2 + \ldots + \underset{\frown}{1}\big/a_j$ is a rational number. (The notation $a_0 + \underset{\frown}{1}\big/a_1 + \underset{\frown}{1}\big/a_2 + \ldots + \underset{\frown}{1}\big/a_j$ is also meaningful if $a_0 \in \mathbb{R}$ and $a_1, \ldots, a_j$ are positive real numbers. We shall use this, however, only in one proof.) Conversely, every rational number can be represented by a continued fraction. This can be constructed by the following algorithm:

Let $\alpha = \alpha_0 \in \mathbb{Q}$ be given. Set $a_0 := \lfloor\alpha_0\rfloor$ and $\alpha_1 := 1/(\alpha_0 - a_0)$. Going on similarly, set for $i = 1, 2, \ldots$

$$a_i := \lfloor\alpha_i\rfloor.$$

If $a_i = \alpha_i$ we stop; otherwise, we let

$$\alpha_{i+1} := \frac{1}{\alpha_i - a_i}.$$

The sequence $(a_0, a_1, a_2, \ldots)$ defined this way is called the **continued fraction expansion** of the number α.

It follows immediately from the definition that $a_i \leq \alpha_i < a_i + 1$ and so $\alpha_{i+1} = 1/(\alpha_i - a_i) > 1$. Thus $a_{i+1} > 0$ for $i \geq 0$. By induction, $\alpha = a_0 + \underset{\frown}{1}\big/a_1 + \ldots + \underset{\frown}{1}\big/a_{i-1} + \underset{\frown}{1}\big/\alpha_i$. Hence, if the procedure terminates after the i-th step then $\alpha = a_0 + \underset{\frown}{1}\big/a_1 + \ldots + \underset{\frown}{1}\big/a_i$. Conversely, if, say, $\alpha = p/q$ with p and q coprime, $q > 0$, then $\alpha_1, \alpha_2, \ldots$ are also rationals, and it follows trivially from the algorithm that if $\alpha_i = p_i/q_i > 0$ and $a_i \neq 0$ then $p_i + q_i > p_{i+1} + q_{i+1} > 0$. So in this case the algorithm must terminate with $a_m = \alpha_m$ for some $m \geq 0$.

Let α be an arbitrary rational number and $a_0 + \underset{\frown}{1}\big/a_1 + \ldots + \underset{\frown}{1}\big/a_m$ its continued fraction expansion. The rational numbers $\beta_k := a_0 + \underset{\frown}{1}\big/a_1 + \ldots + \underset{\frown}{1}\big/a_k$ $(0 \leq k \leq m)$ are called the **convergents** of α.

In practice, when determining the continued fraction expansion of a number α, it is convenient to compute the numerators and denominators of its convergents

β_k along with a_k and α_k. In fact, we can compute, along with the a_i's, two auxiliary sequences g_k and h_k by the recurrence

$$\begin{aligned} &g_{-2} := 0,\ g_{-1} := 1,\ h_{-2} := 1,\ h_{-1} := 0\,, \\ &g_i := a_i g_{i-1} + g_{i-2} \quad (i = 0, 1, \ldots, m)\,, \\ &h_i := a_i h_{i-1} + h_{i-2} \quad (i = 0, 1, \ldots, m)\,. \end{aligned} \tag{5.1.5}$$

It is obvious from the definition of h_i that $1 = h_0 \leq h_1 < h_2 < \ldots < h_m$, and also that $h_k \geq F_k$, where F_k is the k-th Fibonacci number (i. e., $F_0 := 0$, $F_1 := 1$, $F_k := F_{k-1} + F_{k-2}, k \geq 2$). So h_k grows exponentially.

The following identities are easily proved by induction on k:

(5.1.6) Lemma.

(a) $\beta_k = \dfrac{g_k}{h_k}$;

(b) $g_{k+1}h_k - g_k h_{k+1} = (-1)^k$. □

A consequence of Lemma (5.1.6) is that the integers g_k and h_k defined in (5.1.5) are coprime. So for $\alpha = p/q$, p and q coprime, we have $\alpha = \beta_m$ for some m, which implies $p = g_m$ and $q = h_m$. From $q = h_m \geq F_m = \frac{1}{\sqrt{5}}((\frac{1+\sqrt{5}}{2})^m - (\frac{1-\sqrt{5}}{2})^m)$ we can derive $m = O(\log q)$. These considerations yield:

(5.1.7) Theorem. *The continued fraction expansion of a rational number can be found in polynomial time.* □

Next we discuss how well the convergents β_k of a rational number α approximate α. First we show that they are alternately smaller and larger than α.

(5.1.8) Lemma. *Let $0 \leq k < m$. Then $\beta_k < \alpha$ if k is even and $\beta_k > \alpha$ if k is odd.*

Proof. Consider $f(x) := a_0 + \underline{1}\lceil a_1 + \ldots + \underline{1}\lceil a_{k-1} + \underline{1}\lceil x$. Then f is a monotone function of x for $x > 0$ (increasing if k is even and decreasing if k is odd). By definition, $f(a_k) = \beta_k$ and $f(\alpha_k) = \alpha$. Since $0 < a_k < \alpha_k$, the assertion follows. □

We are now able to show how continued fractions can be used to find p and q in Dirichlet's theorem (5.1.3). Let k be the largest index such that $h_k \leq 1/\varepsilon$. From Lemmas (5.1.6) and (5.1.8) we get:

$$|\alpha - \beta_k| < |\beta_{k+1} - \beta_k| = \frac{|g_{k+1}h_k - g_k h_{k+1}|}{h_k h_{k+1}} = \frac{1}{h_k h_{k+1}} < \frac{\varepsilon}{h_k}.$$

We need a little more work to find a best aproximation. This algorithm is due to KHINTCHINE (1956).

(5.1.9) Theorem. *The Best Approximation Problem* (5.1.1) *is solvable in time polynomial in* $\langle N \rangle$ *and* $\langle \alpha \rangle$.

Proof. Let us compute the continued fraction expansion $a_0 + 1\lfloor a_1 + \ldots + 1\lfloor a_m$ of α. Consider the largest subscript i with $h_i \leq N$. Since $h_i \geq F_i$, we have $i = O(\log N)$. Next, let $j = \lfloor (N - h_{i-1})/h_i \rfloor$; so

$$h_{i-1} + jh_i \leq N < h_{i-1} + (j+1)h_i. \tag{5.1.10}$$

Clearly, $j < a_{i+1}$, as $h_{i-1} + a_{i+1}h_i = h_{i+1} > N$.

We claim that

$$r_1 = \frac{g_i}{h_i} \quad \text{or} \quad r_2 = \frac{g_{i-1} + jg_i}{h_{i-1} + jh_i}$$

is the solution of the Best Approximation Problem (5.1.1).

Suppose i is even. Then by Lemma (5.1.8) $r_1 < \alpha$ and, moreover, it follows from the definition that a_{i+1} is the biggest integer t such that $(tg_i + g_{i-1})/(th_i + h_{i-1}) \geq \alpha$. Since $j < a_{i+1}$ we have $\alpha < r_2$, thus

$$r_1 < \alpha < r_2.$$

(Note that in case i is odd we similarly have $r_2 < \alpha < r_1$.) Let p/q be any rational number with $r_1 < p/q < r_2$. Then

$$\frac{p}{q} - r_1 = \frac{ph_i - g_i q}{qh_i} \geq \frac{1}{qh_i}$$

and similarly

$$r_2 - \frac{p}{q} \geq \frac{1}{q(h_{i-1} + jh_i)}.$$

Hence

$$r_2 - r_1 \geq \frac{1}{q}\left(\frac{1}{h_i} + \frac{1}{h_{i-1} + jh_i}\right) = \frac{h_{i-1} + (j+1)h_i}{qh_i(h_{i-1} + jh_i)}. \tag{5.1.11}$$

On the other hand, we have by Lemma (5.1.6),

$$r_2 - r_1 = \frac{g_{i-1}h_i - g_i h_{i-1}}{h_i(h_{i-1} + jh_i)} = \frac{1}{h_i(h_{i-1} + jh_i)}. \tag{5.1.12}$$

From (5.1.11), (5.1.12) and (5.1.10) we obtain that

$$q \geq h_{i-1} + (j+1)h_i > N.$$

The result follows similarly for i odd. So indeed any rational number closer to α than r_1 and r_2 has denominator larger than N. □

5.2 Simultaneous Diophantine Approximation: Formulation of the Problems

In some applications, we shall have to approximate certain numbers $\alpha_1, \ldots, \alpha_n$ by rationals with a common denominator q. Note that this is trivial if we allow an error of $1/(2q)$ in each approximation; but one can do better if only an upper bound on q is prescribed.

The following classical theorem of DIRICHLET (1842) guarantees that the simultaneous approximation problem has a solution in which q, and hence $p_1, \ldots, p_n$, are not too large.

(5.2.1) Theorem. *Given any real numbers $\alpha_1, \ldots, \alpha_n$ and $0 < \varepsilon < 1$, there exist integers $p_1, \ldots, p_n, q$ such that $|\alpha_i - p_i/q| < \varepsilon/q$ for $i = 1, \ldots, n$ and $1 \leq q \leq \varepsilon^{-n}$.*

Proof. Let $\alpha := (\alpha_1, \ldots, \alpha_n)^T$ and $m := \lfloor \varepsilon^{-n} \rfloor$. Consider the set S of all points in $\mathbb{R}^n$ of the form

$$k\alpha + z, \; z \in \mathbb{Z}^n, \; 0 \leq k \leq m.$$

Draw about each point $v \in S$ the open cube

$$\{x \in \mathbb{R}^n \mid \|x - v\|_\infty < \varepsilon/2\}.$$

If these cubes were disjoint then the density of their union would be $(m+1)\varepsilon^{-n} > 1$, which is impossible. So there are two such cubes which intersect, i. e., there are two different points in S, say $k_1\alpha + z_1$ and $k_2\alpha + z_2$, such that $\|k_1\alpha + z_1 - k_2\alpha - z_2\|_\infty < \varepsilon$. As $\varepsilon < 1$, we have $k_1 \neq k_2$, say $k_1 < k_2$. Then $q := k_2 - k_1$ and $p := z_1 - z_2 = (p_1, \ldots, p_n)^T$ satisfy the requirements of the theorem. □

Unfortunately, no polynomial time algorithm is known to find such $p_1, \ldots, p_n$ and q with denominator $q \leq \varepsilon^{-n}$ (even though we know by Dirichlet's theorem that such solutions do exist). In the next section we shall describe an algorithm which does find a simultaneous approximation in polynomial time, but the upper bound on the denominator is $2^{n(n+1)/4}\varepsilon^{-n}$.

Similarly as in the case of $n = 1$, we can ask for the best possible approximation of several given numbers by rationals with a common small denominator. Formulated as a decision problem, we can state this as follows.

(5.2.2) Simultaneous Diophantine Approximation Problem. *Given rationals $\alpha_1, \ldots, \alpha_n$, $\varepsilon > 0$, and an integer $N > 0$, decide if there exist integers $p_1, \ldots, p_n$ and an integer $q > 0$ such that $q \leq N$ and $|q\alpha_i - p_i| < \varepsilon$ for $i = 1, \ldots, n$.*

This problem is $\mathcal{NP}$-complete (LAGARIAS (1982)). However, a weaker problem will be solved in the next section. (Note that Dirichlet's theorem says that the answer to this problem is "yes" if $N \geq \varepsilon^{-n}$).

A closely related number theoretical problem is the following.

(5.2.3) Small Linear Form Problem. *Given rationals* $\alpha_1, \ldots, \alpha_n$, $\varepsilon > 0$, *and an integer* $N > 0$, *find integers* $p_1, \ldots, p_n$, *not all* 0, *such that* $|p_i| \leq N$ *for* $i = 1, \ldots, n$ *and* $|\alpha_1 p_1 + \ldots + \alpha_n p_n| < \varepsilon$.

(It would be easy to formulate a common generalization of problems (5.2.2) and (5.2.3) by asking for several linear forms to be simultaneously small, but we postpone this to the next section, where point lattices provide a convenient framework for these algorithmic problems.)

Again we may want to find $p_1, \ldots, p_n$ here that are not too large. DIRICHLET (1842) proved the following analogue of Theorem (5.2.1):

(5.2.4) Theorem. *Given any real numbers* $\alpha_1, \ldots, \alpha_n$, *and* $0 < \varepsilon < 1$, *there exist integers* $p_0, p_1, \ldots, p_n$, *not all* 0, *such that* $|p_i| \leq \varepsilon^{-1/n}$ *for* $i = 1, \ldots, n$ *and* $|p_0 + \alpha_1 p_1 + \ldots + \alpha_n p_n| < \varepsilon$. □

However, problem (5.2.3) is $\mathcal{NP}$-complete (even if we restrict (5.2.3) to $N = 1$, $\varepsilon = 1/2$ and α_i integer, as the so-called subset sum problem can be reduced to this – see Section 7.6). But in the next section we formulate an algorithm which solves a somewhat weaker problem which is, however, still sufficient for most number-theoretic applications. (Again, note that by Theorem (5.2.4) the answer to (5.2.3) is always yes if N is large enough.)

Let us conclude with the remark that in problems (5.2.2) and (5.2.3) we could replace the hypothesis that $\alpha_1, \ldots, \alpha_n$ are rational by the hypothesis that $(\alpha_1, \ldots, \alpha_n)^T$ is a polynomially computable vector. This observation brings these problems closer to the number-theoretic applications (where the α_i are algebraic numbers or numbers like e, π, etc.), but it does not add to the algorithmic aspects. Namely, if $\alpha_1, \ldots, \alpha_n$ are irrational then we compute a rational approximation first (not necessarily with the same denominator), and then compute a simultaneous approximation or small linear form of this approximation.

5.3 Basis Reduction in Lattices

A **lattice** in $\mathbb{R}^n$ is any set of the form

$$L = L(b_1, \ldots, b_n) = \left\{ \sum_{i=1}^{n} \lambda_i b_i \mid \lambda_i \in \mathbb{Z},\ i = 1, \ldots, n \right\} \tag{5.3.1}$$

where $\{b_1, \ldots, b_n\}$ is a basis of $\mathbb{R}^n$. We say that $\{b_1, \ldots, b_n\}$ is a **basis of** L.

In this book we do not go into the detailed study of lattices; we shall only address the problem of finding a "decent" basis in this section, and discuss a few other elementary problems in the next. For more information on lattice geometry, consult CASSELS (1959) or LEKKERKERKER (1969).

Clearly, a lattice L may have several bases. If $\{b_1, \ldots, b_n\}$ is a basis of L and if $A = (a_{ij})_{i,j=1}^n$ is any integral matrix with determinant ± 1 then $\{b'_1, \ldots, b'_n\}$

defined by

$$b_i' := \sum_{j=1}^{n} a_{ij} b_j \quad (i = 1, \ldots, n) \tag{5.3.2}$$

is another basis of L. It is well known that all bases of the lattice (5.3.1) arise by such a transformation. Hence $|\det(b_1, \ldots, b_n)|$ is independent of the choice of the basis $\{b_1, \ldots, b_n\}$ for the lattice L. We set

$$\det L := |\det(b_1, \ldots, b_n)|.$$

Geometrically speaking, $\det L$ is the (common) volume of each parallelepiped that has its vertices in L but no other point in common with L.

Of the many possible bases of a lattice, one would like to select one that is in some sense nice or simple, or "reduced", which is the term used in this area. There are many different notions of "reduced" bases in the literature. One of them consists of requiring that all of its vectors be short in the following sense.

(5.3.3) Minimum Basis Problem. *Given linearly independent vectors $a_1, \ldots, a_n \in \mathbb{Q}^n$, find a basis $\{b_1, \ldots, b_n\}$ of the lattice $L(a_1, \ldots, a_n)$ such that $\|b_1\| \cdot \ldots \cdot \|b_n\|$ is minimal.*

This problem is $\mathcal{NP}$-hard (LOVÁSZ, unpublished).

One may ask why do we consider the product, and not, say, the sum of the norms of basis vectors. An explanation is that the product is closely related to the determinant of the lattice. In fact, it follows from Hadamard's inequality (0.1.27) that for any basis $b_1, \ldots, b_n$,

$$\|b_1\| \cdot \ldots \cdot \|b_n\| \geq \det L.$$

On the other hand, HERMITE (1850) proved the following.

(5.3.4) Theorem. *Every lattice L in $\mathbb{R}^n$ has a basis $\{b_1, \ldots, b_n\}$ such that*

$$\|b_1\| \cdot \ldots \cdot \|b_n\| \leq c_n \det L,$$

where c_n is a constant that depends only on n. □

The best known value of c_n, for large enough n, is about $1.43(0.97n)^n n^{1/4}$.

One might try to find short linearly independent vectors one after one. This leads to the **theory of successive minima**, developed by Minkowski. We do not want to go into the details of this, but we formulate the first problem suggested by this approach:

(5.3.5) Shortest Lattice Vector Problem. *Given linearly independent vectors $a_1, \ldots, a_n \in \mathbb{Q}^n$, find a nonzero vector $v \in L(a_1, \ldots, a_n)$ with $\|v\|$ minimal.*

The following classical theorem of Minkowski gives essential information about the shortest vector in different norms.

(5.3.6) Minkowski's Theorem. *If $K \subseteq \mathbb{R}^n$ is a convex body centrally symmetric with respect to the origin, and $L \subseteq \mathbb{R}^n$ is a lattice such that*

$$\text{vol}(K) \geq 2^n \det L,$$

then K contains a lattice point different from the origin. □

In other words, the shortest nonzero lattice vector in any vector norm N is not larger than $2(\det L/\text{vol}(S))^{1/n}$, where $S = \{x \in \mathbb{R}^n \mid N(x) \leq 1\}$ is the "unit ball" of the norm N.

(5.3.7) Exercise. *Derive Dirichlet's theorems* (5.2.1) *and* (5.2.4) *from Minkowski's theorem.* □

Applying Minkowski's theorem to the Euclidean norm we obtain:

(5.3.8) Corollary. *In every lattice L in $\mathbb{R}^n$, there exists a vector $v \neq 0$ such that*

$$\|v\| < c\sqrt{n}\sqrt[n]{\det L},$$

where c is a constant. □

The best value of c known is about 0.3196, for large enough n.

It is not known whether the Shortest Lattice Vector Problem (5.3.5) is $\mathcal{NP}$-hard or not; quite probably it is. It is known to be $\mathcal{NP}$-hard if the Euclidean norm is replaced by the maximum norm (VAN EMDE BOAS (1981)).

There is a related, but essentially different problem:

(5.3.9) Nearest Lattice Vector Problem. *Given n linearly independent vectors $a_1, \ldots, a_n \in \mathbb{Q}^n$, and a further vector $b \in \mathbb{Q}^n$, find a vector $v \in L(a_1, \ldots, a_n)$ with $\|b - v\|$ minimal.*

This problem is known to be $\mathcal{NP}$-hard for any norm (VAN EMDE BOAS (1981)).

Unfortunately, none of the known proofs of the above mentioned results (5.3.4), (5.3.6), (5.3.8) is constructive in the sense of a polynomial time algorithm. We shall develop an algorithm which will construct short vectors in polynomial time – but the bounds we obtain will be worse. It is an outstanding open problem to find a polynomial time algorithm to construct a basis with the property in Theorem (5.3.4) and a vector v with the property in Corollary (5.3.8), with the best possible values of c_n and c. However, if we allow larger values (depending on n) instead of c_n and $c\sqrt{n}$, then these problems are polynomially solvable. This result, due to A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász is the main topic of this section.

We shall define a rather technical notion of "reducedness" and then show that a reduced basis in this sense, on the one hand, is polynomially constructible and at the same time, satisfies Theorem (5.3.4) (with a somewhat poor c_n).

Recall from Section 1.4 that the Gram-Schmidt orthogonalization $(b_1^*, \ldots, b_n^*)$ of an ordered basis $(b_1, \ldots, b_n)$ of $\mathbb{R}^n$ satisfies

$$b_j = \sum_{i=1}^{j} \mu_{ji} b_i^* \quad (j = 1, \ldots, n), \tag{5.3.10}$$

and that in this formula $\mu_{jj} = 1$. A connection between the Gram-Schmidt orthogonalization and our problems is shown by the following lemma.

(5.3.11) Lemma. *Let L be a lattice in $\mathbb{R}^n$, $(b_1, \ldots, b_n)$ an ordered basis of L, and $(b_1^*, \ldots, b_n^*)$ the Gram-Schmidt orthogonalization of $(b_1, \ldots, b_n)$. Then for any non-zero vector b in L,*

$$\|b\| \geq \min\{\|b_1^*\|, \ldots, \|b_n^*\|\}.$$

Proof. By definition,

$$b = \sum_{i=1}^{n} \lambda_i b_i,$$

where $\lambda_i \in \mathbb{Z}$. Let k be the last subscript with $\lambda_k \neq 0$. Substituting for b_i by (5.3.10) we obtain

$$b = \sum_{i=1}^{k} \lambda_i^* b_i^*,$$

where $\lambda_k^* = \lambda_k$ is a nonzero integer. Thus

$$\|b\|^2 = \sum_{i=1}^{k} (\lambda_i^*)^2 \|b_i^*\|^2 \geq \lambda_k^2 \|b_k^*\|^2 \geq \|b_k^*\|^2,$$

which proves the Lemma. □

The key of our algorithmic approach is the following technical definition. Let L be a lattice, $(b_1, \ldots, b_n)$ an ordered basis of L, and $(b_1^*, \ldots, b_n^*)$ its Gram-Schmidt orthogonalization; let the number μ_{ij} be defined by (5.3.10). We say that the basis $(b_1, \ldots, b_n)$ is **reduced**, if the following two conditions hold:

(5.3.12) (i) $|\mu_{ji}| \leq \frac{1}{2}$ for every $1 \leq i < j \leq n$;

(ii) $\|b_{j+1}^* + \mu_{j+1,j} b_j^*\|^2 \geq \frac{3}{4} \|b_j^*\|^2$ for $j = 1, \ldots, n-1$.

The first of these conditions is rather natural – it means that the basis is reasonably "almost orthogonal". The mysterious looking second condition can be illuminated as follows. Since the Gram-Schmidt orthogonalization depends on the order of the basis elements, its vectors change if b_j and b_{j+1} are interchanged; in fact b_j^* and b_{j+1}^* will change. Now the new b_j^* is the vector $b_{j+1}^* + \mu_{j+1,j} b_j^*$, so (5.3.12) (ii) says that by interchanging b_j and b_{j+1}, the length of b_j^* does not drop too much. The choice of 3/4 as the allowed factor of decrease is arbitrary: any number between 1/4 and 1 would do just as well. The most natural choice (giving the best values in Theorem (5.3.13) below) would be 1 instead of 3/4; but we could not guarantee the polynomiality of the resulting algorithm.

(5.3.13) Theorem. *Let L be a lattice in $\mathbb{R}^n$ and $(b_1, \ldots, b_n)$ a reduced basis of L. Then the following are true:*

(a) $\|b_1\| \leq 2^{(n-1)/4}\sqrt[n]{\det L}$;

(b) $\|b_1\| \leq 2^{(n-1)/2} \min\{\|b\| : b \in L, b \neq 0\}$;

(c) $\|b_1\| \cdot \ldots \cdot \|b_n\| \leq 2^{\binom{n}{2}/2} \det L$.

Proof. By property (ii) of (5.3.12) in the definition of a reduced basis, we have

$$\tfrac{3}{4}\|b_j^*\|^2 \leq \|b_{j+1}^* + \mu_{j+1,j} b_j^*\|^2 = \|b_{j+1}^*\|^2 + \mu_{j+1,j}^2 \|b_j^*\|^2.$$

By property (i), we have $\mu_{j+1,j}^2 \leq 1/4$, and thus

$$\|b_{j+1}^*\|^2 \geq \tfrac{1}{2}\|b_j^*\|^2 \quad (1 \leq j \leq n-1).$$

Hence it follows by induction that

$$\|b_j^*\|^2 \geq 2^{i-j}\|b_i^*\|^2 \quad (1 \leq i < j \leq n). \tag{5.3.14}$$

In particular,

$$\|b_j^*\|^2 \geq 2^{1-j}\|b_1^*\|^2 = 2^{1-j}\|b_1\|^2.$$

Multiplying this inequality for $i = 1, \ldots, n$, we obtain

$$\|b_1^*\|^2 \cdot \ldots \cdot \|b_n^*\|^2 \geq 2^{-\binom{n}{2}}\|b_1\|^{2n}.$$

But the left hand side is just $(\det L)^2$, from which (a) follows.

To prove (b), notice that (5.3.14) implies

$$\min_{1 \leq j \leq n} \|b_j^*\| \geq 2^{-(n-1)/2}\|b_1\|$$

and since by Lemma (5.3.11) the left hand side is a lower bound on any $\|b\|$ $(b \in L, b \neq 0)$, (b) follows.

Finally, using property (5.3.12) (i) of reduced bases,

$$\|b_j\|^2 = \sum_{i=1}^{j} \mu_{ji}^2 \|b_i^*\|^2 \leq \|b_j^*\|^2 + \sum_{i=1}^{j-1} \tfrac{1}{4}\|b_i^*\|^2.$$

So by (5.3.14),

$$\|b_j\|^2 \leq (1 + \sum_{i=1}^{j-1} \tfrac{1}{4} \cdot 2^{j-i})\|b_j^*\|^2 \leq 2^{j-1}\|b_j^*\|^2. \tag{5.3.15}$$

Multiplying this for $j = 1, 2, \ldots, n$, we obtain

$$\|b_1\|^2 \cdot \ldots \cdot \|b_n\|^2 \leq 2^{\binom{n}{2}}\|b_1^*\|^2 \cdot \ldots \cdot \|b_n^*\|^2 = 2^{\binom{n}{2}}(\det L)^2,$$

from where (c) follows. □

(5.3.16) Theorem. *There is a polynomial time algorithm that, for any given linearly independent vectors $a_1, \ldots, a_n$ in $\mathbb{Q}^n$, finds a reduced basis of the lattice* $L(a_1, \ldots, a_n)$.

Let us remark that, in contrast to the ellipsoid method, the algorithm given below is reasonably efficient in practice.

Proof. Without loss of generality, assume that $a_1, \ldots, a_n$ are integral. We shall transform the given basis until both conditions in the definition of reduced bases are satisfied. So start with an ordered basis $(b_1, \ldots, b_n) := (a_1, \ldots, a_n)$. Let $(b_1^*, \ldots, b_n^*)$ be the Gram-Schmidt orthogonalization of this basis, and let the coefficients μ_{ji} be defined by (5.3.10).

Step (I): *For $j = 1, 2, \ldots, n$, and, given j, for $i = 1, 2, \ldots, j-1$, replace b_j by $b_j - \lceil \mu_{ji} \rfloor b_i$, where $\lceil \mu_{ji} \rfloor$ is the integer nearest to μ_{ji}.*

It is clear that this procedure does not change the Gram-Schmidt orthogonalization $(b_1^*, \ldots, b_n^*)$ of the basis. Hence, after executing step (I) for a certain j, the new μ_{ji} will satisfy $|\mu_{ji}| \leq 1/2$ for all $i = 1, \ldots, j-1$, and executing the step for higher j will not spoil this property. If all pairs (j, i) are treated, (i) of (5.3.12) is achieved and we go to Step (II).

Step (II): *If there is a subscript j violating (ii) of* (5.3.12) *then interchange b_j and b_{j+1} and return to Step (I).*

To analyse this step, let $(c_1^*, \ldots, c_n^*)$ be the Gram-Schmidt orthogonalization of the resulting basis. As remarked before, $c_i^* = b_i^*$ unless $i = j$ or $j+1$. Further, $c_j^* = b_{j+1}^* + \mu_{j+1,j} b_j^*$ and the fact that (ii) fails for j means just that

$$\|c_j^*\|^2 < \tfrac{3}{4}\|b_j^*\|^2.$$

The formula for c_{j+1}^* is somewhat more complicated and we do not need it explicitly. But from

$$\|c_1^*\|^2 \cdot \ldots \cdot \|c_n^*\|^2 = \|b_1^*\|^2 \cdot \ldots \cdot \|b_n^*\|^2 = (\det L)^2$$

it follows that $\|c_j^*\|^2 \cdot \|c_{j+1}^*\|^2 = \|b_j^*\|^2 \cdot \|b_{j+1}^*\|^2$. So $\|c_j^*\|^{2(n-j+1)}\|c_{j+1}^*\|^{2(n-j)} < \tfrac{3}{4}\|b_j^*\|^{2(n-j+1)}\|b_{j+1}^*\|^{2(n-j)}$. To estimate how many steps have to be carried out before a reduced basis is obtained (if at all), let us study the quantity

$$D := \|b_1^*\|^{2n}\|b_2^*\|^{2(n-1)} \cdot \ldots \cdot \|b_n^*\|^2.$$

By the remarks above, Step (I) does not change the value of D, while Step (II) decreases it by a factor $< 3/4$. We get by elementary linear algebra that

$$\|b_1^*\|^2 \cdot \ldots \cdot \|b_j^*\|^2 = \det((b_\nu^T b_\mu)_{\nu,\mu=1}^j)$$

and hence

$$D = \prod_{j=1}^{n} \det((b_\nu^T b_\mu)_{\nu,\mu=1}^j). \tag{5.3.17}$$

Since the vectors $b_1, \ldots, b_n$ clearly remain integral during the procedure, D is an integer and hence $D \geq 1$.

At the beginning, we have

$$D_0 := \|a_1^*\|^{2n}\|a_2^*\|^{2(n-1)} \cdot \ldots \cdot \|a_n^*\| \leq \|a_1\|^{2n}\|a_2\|^{2(n-1)} \cdot \ldots \cdot \|a_n\|^2$$
$$\leq (\|a_1\| \cdot \ldots \cdot \|a_n\|)^{2n}.$$

So the number of times Step (II) has to be carried out is at most

$$\frac{\log D_0}{\log \frac{4}{3}} \leq \frac{2n}{\log 4 - \log 3}(\log \|a_1\| + \ldots + \log \|a_n\|),$$

which is polynomial in the encoding length of the input. Between two executions of Steps (II) there is only one execution of Step (I), with $O(n^3)$ arithmetic operations, so we obtain that the total number of arithmetic operations in this algorithm is polynomial.

We shall have to show that the numerators and denominators of the numbers occuring do not grow too large, i. e., their encoding lengths remain polynomial. First, it follows by (5.3.17) and by the definition of b_j^* that Db_j^* is an integral vector. In Step (II),

$$\max\{\|c_i^*\| \mid 1 \leq i \leq n\} \leq \max\{\|b_i^*\| \mid 1 \leq i \leq n\},$$

since $c_j^* = b_j^*$ if $i \neq j, j+1$, $\|c_j^*\| < \|b_j^*\|$ by the condition on this step and $\|c_{j+1}^*\| \leq \|b_j^*\|$ since c_{j+1}^* is the "component" of b_j^* perpendicular to c_j^*. Hence in any step,

$$\max\{\|b_j^*\| \mid 1 \leq j \leq n\} \leq \max\{\|a_j^*\| \mid 1 \leq j \leq n\}$$
$$\leq \max\{\|a_j\| \mid 1 \leq j \leq n\} =: A_0.$$

So every entry of b_j^* is a rational number whose denominator is a divisor of D and whose numerator is at most $A_0 D$. Thus the vectors b_j^* remain "decent".

We still need to estimate the vectors b_j. These are integral. After one execution of Step (I) is finished, we have

$$\|b_j\|^2 = \sum_{i=1}^{j} \mu_{ji}^2 \|b_i^*\|^2 \leq \sum_{i=1}^{j} \|b_i^*\|^2 \leq n \cdot A_0^2. \tag{5.3.18}$$

Step (II) does not change the vectors b_j (only permutes them). Finally, if during Step (I) b_j is replaced by $b_j - \lceil \mu_{ji} \rfloor b_i$, then we have

$$|\mu_{ji}| = \frac{\|b_j^T b_i^*\|}{\|b_i^*\|^2} \leq \frac{\|b_j\|}{\|b_i^*\|} \leq D\|b_j\|,$$

$$|\lceil \mu_{ji} \rfloor| \leq 2|\mu_{ji}| \leq 2D\|b_j\|,$$

and so

$$\|b_j - \lceil \mu_{ji} \rfloor b_i\| \leq \|b_j\| + |\lceil \mu_{ji} \rfloor| \, \|b_i\| \leq \|b_j\| + 2D\|b_i\| \, \|b_j\|.$$

Now by the time we are changing b_j, the vector b_i is "settled" (i. e., it will not be changed until the end of this step (I)), so we already have $\|b_i\| \leq \sqrt{n}A_0$ from (5.3.18). Hence

$$\|b_j - \lceil \mu_{ji} \rfloor b_i\| \leq \|b_j\|(1 + 2\sqrt{n}DA_0) < 2nDA_0\|b_j\|,$$

and since b_j is changed at most $j - 1 < n$ times, it follows that $\|b_j\|$ never exceeds

$$(2nDA_0)^n \cdot \sqrt{n} \cdot A_0.$$

This proves that no entry of any b_i ever exceeds this bound, and so the number of its digits remains polynomial.

This completes the proof of Theorem (5.3.16). □

If we replace the "technical factor" 3/4 in the definition of reduced basis (5.3.12) (ii) by a number close to 1, we could improve the exponential factors $2^{(n-1)/4}$ etc. in Theorem (5.3.13) to $(4/3 + \varepsilon)^{(n-1)/4}$ etc. Moreover, a reduced basis could still be found in time polynomial in $\langle a_1 \rangle + \ldots + \langle a_n \rangle + 1/\varepsilon$. So this is still polynomial for every fixed ε. A more substantial improvement was found by SCHNORR (1985). He showed, using a technically quite a bit more involved refinement of the above method, that the exponential factors $2^{(n-1)/4}$ etc. can be replaced by $(1 + \varepsilon)^{(n-1)/4}$ for every fixed $\varepsilon > 0$.

We give some applications of Theorems (5.3.13) and (5.3.16) to diophantine approximation problems.

(5.3.19) Theorem. *There exists a polynomial time algorithm that, given rational numbers $\alpha_1, \ldots, \alpha_n$, and $0 < \varepsilon < 1$, computes integers $p_1, \ldots, p_n$, and an integer q such that*

$$1 \leq q \leq 2^{n(n+1)/4}\varepsilon^{-n}$$

and

$$|\alpha_i q - p_i| < \varepsilon \quad (i = 1, \ldots, n).$$

In view of Dirichlet's theorem (5.2.1), this result is quite unsatisfactory because of the large factor $2^{n(n+1)/4}$ in the upper bound for q. Note, however, that for n fixed and $\varepsilon \to 0$ the order of magnitude of q is best possible – see SCHMIDT (1980). Also, this result will suffice for the applications in this book.

Proof of (5.3.19). Let $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{R}^{n+1}$ $(i = 1, \ldots, n)$ and $a = (\alpha_1, \ldots, \alpha_n, 2^{-n(n+1)/4}\varepsilon^{n+1})^T$. Let $L \subseteq \mathbb{R}^{n+1}$ be the lattice generated by $\{e_1, \ldots, e_n, a\}$. Then trivially

$$\det L = 2^{-n(n+1)/4}\varepsilon^{n+1}.$$

By Theorem (5.3.16), we can find a reduced basis in L in polynomial time, and by Theorem (5.3.13) (a), the first vector b_1 of this basis satisfies

$$\|b_1\| \leq 2^{n/4} \sqrt[n+1]{\det L} = \varepsilon. \tag{5.3.20}$$

Let us write $b_1 = p_1e_1 + \ldots + p_ne_n - qa$ with $p_1, \ldots, p_n, q \in \mathbb{Z}$. Since $\varepsilon < 1$, q is nonzero, so we may assume that $q > 0$. Then the i-th coordinate of b_1 is $p_i - q\alpha_i$ for $i = 1, \ldots, n$ and is $2^{-n(n+1)/4}\varepsilon^{n+1}q$ for $i = n+1$. Thus we have from (5.3.20)

$$|p_i - q\alpha_i| < \varepsilon \quad (i = 1, \ldots, n)$$

and

$$2^{-n(n+1)/4}\varepsilon^{n+1}q \leq \varepsilon, \text{ i. e., } q \leq 2^{n(n+1)/4}\varepsilon^{-n}.$$

□

It may be worthwhile to specialize property (5.3.13) (b) of reduced bases as well:

(5.3.21) Theorem. *There exists a polynomial time algorithm that, for any given rational numbers $\alpha_1, \ldots, \alpha_n$, $\varepsilon > 0$, and any given integer $N > 0$, either*

(a) *asserts that the Simultaneous Diophantine Approximation Problem* (5.2.2) *has no solution for the given input, or*
(b) *finds integers $p_1, \ldots, p_n$ and an integer q such that*

$$|\alpha_i q - p_i| < 2^{n/2}\varepsilon\sqrt{n+1}, \; i = 1, \ldots, n \text{ and } 0 < q \leq 2^{n/2}N\sqrt{n+1}.$$

Proof. Let $e_1, \ldots, e_n \in \mathbb{R}^{n+1}$ be as in the proof of Theorem (5.3.19), and let $a := (\alpha, \ldots, \alpha_n, \varepsilon/N)^T$. Let L be the lattice generated by $\{e_1, \ldots, e_n, a\}$. If $p_1', \ldots, p_n', q'$ is a solution of the simultaneous diophantine approximation problem, then $p_1'e_1 + \ldots + p_n'e_n - q'a$ is a nonzero vector in L of length less than $\varepsilon\sqrt{n+1}$. So with the basis reduction algorithm we can find in L a vector $b_1 \neq 0$ with $\|b_1\| < 2^{n/2}\varepsilon\sqrt{n+1}$. Write $b_1 = p_1e_1 + \ldots + p_ne_n - qa$ with $p_1, \ldots, p_n, q \in \mathbb{Z}$. If $q \neq 0$, then $p_1, \ldots, p_n, q$ is a valid output for (b). If $q = 0$, then we must have $2^{n/2}\varepsilon\sqrt{n+1} \geq 1$ and so $q = 1$ and $p_i = \lfloor \alpha_i \rfloor$, $i = 1, \ldots, n$ is a valid output for (b). □

In a similar way we obtain the following results.

(5.3.22) Theorem. *There exists a polynomial time algorithm that, for any given rational numbers $\alpha_1, \ldots, \alpha_n$, and $0 < \varepsilon < 1$, computes integers $p_1, \ldots, p_n$, not all 0, such that $|\alpha_1p_1 + \ldots + \alpha_np_n| < \varepsilon$ and $|p_i| \leq 2^{(n+1)/4}\varepsilon^{-1/n}$.* □

(5.3.23) Theorem. *There exists a polynomial time algorithm that, for any given rational numbers $\alpha_1, \ldots, \alpha_n$, $\varepsilon > 0$, and any given integer $N > 0$, either*

(a) *asserts that the Small Linear Form Problem* (5.2.3) *has no solution for the given input, or*
(b) *finds integers $p_1, \ldots, p_n$, not all 0, such that*

$$\left|\sum \alpha_i p_i\right| < 2^{n/2}\varepsilon\sqrt{n+1}, \; |p_i| \leq 2^{n/2}N\sqrt{n+1} \; (i = 1, \ldots, n).$$

□

It is easy to modify the basis reduction algorithm described above, replacing the Euclidean norm $\|\cdot\|$ by any other ellipsoidal norm $\|x\|_A = \sqrt{x^T A^{-1} x}$ defined by a positive definite matrix A. But in fact the algorithm can be extended to an arbitrary norm with the help of the results in Section 4.6.

(5.3.24) Theorem. *There exists an oracle-polynomial time algorithm that, for any given set of n linearly independent vectors $\{a_1, \ldots, a_n\} \subseteq \mathbb{Q}^n$ and for any given norm N on $\mathbb{R}^n$ (specified as usual by a weak norm oracle and two rational numbers $r, R > 0$ such that $r\|x\| \leq N(x) \leq R\|x\|$ for all $x \in \mathbb{R}^n$), finds a basis $b_1, \ldots, b_n$ of the lattice $L = L(a_1, \ldots, a_n)$ with the following properties:*

(a) $N(b_1) \leq 2^{(n-1)/4} C_N^{1/n} (n+1) \sqrt[n]{\det L}$,

(b) $N(b_1) \leq 2^{(n-1)/2} \cdot (n+1) \cdot \min\{N(b) \mid b \in L, b \neq 0\}$,

(c) $N(b_1) \cdot \ldots \cdot N(b_n) \leq (n+1)^n 2^{n(n-1)/4} C_N \det L$.

(Here C_N is the ratio of the volumes of the unit balls with respect to the Euclidean norm and the norm N.)

Proof. By Corollary (4.6.9) we can construct a linear transformation Φ in oracle-polynomial time such that

$$\|\Phi x\| \leq N(x) \leq (n+1)\|\Phi x\|.$$

Applying Theorems (5.3.13) and (5.3.16) to the lattice $L' = \Phi L$ we obtain, in oracle-polynomial time, a basis $\{b'_1, \ldots, b'_n\}$ of ΦL such that

(a′) $\|b'_1\| \leq 2^{(n-1)/4} \sqrt[n]{\det L'} = 2^{(n-1)/4} \sqrt[n]{|\det \Phi|} \cdot \sqrt[n]{\det L}$,

(b′) $\|b'_1\| \leq 2^{(n-1)/2} \min\{\|b'\| \mid b' \in L', b' \neq 0\}$,

(c′) $\|b'_1\| \cdot \ldots \cdot \|b'_n\| \leq 2^{\binom{n}{2}/2} |\det \Phi| \cdot \det L$.

Write $b_i := \Phi^{-1} b'_i$. Then $\{b_1, \ldots, b_n\}$ is a basis of L and has the following properties:

(a″) $N(b_1) \leq (n+1)\|\Phi b_1\| = (n+1)\|b'_1\| \leq 2^{(n-1)/4}(n+1)\sqrt[n]{|\det \Phi|} \cdot \sqrt[n]{\det L}$,

(b″) $N(b_1) \leq (n+1)\|b'_1\| \leq 2^{(n-1)/2}(n+1) \min\{\|b'\| \mid b' \in L', b' \neq 0\}$
$\leq 2^{(n-1)/2}(n+1) \min\{N(b) \mid b \in L, b \neq 0\}$,

(c″) $N(b_1) \cdot \ldots \cdot N(b_n) \leq (n+1)^n \|b'_1\| \ldots \|b'_n\|$
$\leq (n+1)^n 2^{\binom{n}{2}/2} |\det \Phi| \cdot \det L$.

To conclude, we only have to estimate $\det \Phi$. From the definition of Φ we know that the unit ball K of N satisfies $\Phi(K) \subseteq S(0,1)$ and hence $|\det \Phi| \leq C_N$. This proves the theorem. □

The fact that a vector b_1 with property (5.3.24) (a) can be found may be viewed as a polynomially constructive form of Minkowski's theorem (5.3.6):

(5.3.25) Corollary. *There exists an oracle-polynomial time algorithm that, for any lattice L, specified by n linearly independent vectors $a_1, \ldots, a_n \in \mathbb{Q}^n$, and for any well-bounded convex body $(K; n, R, r)$, specified by a weak membership oracle, such that K is centrally symmetric with respect to 0 and*

$$\mathrm{vol}(K) \geq \frac{2^{n(n-1)/4}\pi^{n/2}}{\Gamma(n/2+1)}(n+1)^n \det L,$$

finds a non-zero lattice point in K.

Proof. Consider K as the unit ball of a vector norm N. The lattice vector b_1 found in Theorem (5.3.24) satisfies $N(b_1) \leq 1$ by (5.3.24) (a). □

Basis reduction, in particular in the form of the diophantine approximation procedure, has other applications in number theory. For instance, it can be used to obtain the factorization of a polynomial $f \in \mathbb{Q}[x]$ into irreducible factors (LENSTRA, LENSTRA and LOVÁSZ (1982)). ODLYZKO and TE RIELE (1985) used the basis reduction algorithm to disprove a long-standing conjecture in number theory, the so-called Mertens conjecture. Their approach involved basis reduction in a 70-dimensional space. The basis reduction algorithm was also successfully used for breaking certain encryption schemes in cryptology – cf. Section 7.6. The next section contains further applications of this algorithm, mostly in conjunction with the ellipsoid method.

The basis reduction algorithm described above can also be used to find an "approximately nearest" lattice point (cf. problem (5.3.9)). The following result is due to BABAI (1986):

(5.3.26) Theorem. *There exists a polynomial time algorithm that, for any n given linearly independent vectors $a_1, \ldots, a_n \in \mathbb{Q}^n$ and for any given further vector $b \in \mathbb{Q}^n$, finds $w \in \mathrm{L}(a_1, \ldots, a_n)$ such that*

$$\|b - w\| \leq 2^{(n/2)-1} \min\{\|b - v\| \mid v \in L(a_1, \ldots, a_n)\}.$$

Proof. We start with finding a reduced basis $(b_1, \ldots, b_n)$ in $\mathrm{L}(a_1, \ldots, a_n)$, by Theorem (5.3.16). Let $(b_1^*, \ldots, b_n^*)$ be the Gram-Schmidt orthogonalization of $(b_1, \ldots, b_n)$. Next we find a lattice vector $w \in \mathrm{L}(a_1, \ldots, a_n)$ such that we have

$$b - w = \sum_{i=1}^{n} \lambda_i b_i^*, \quad |\lambda_i| \leq \frac{1}{2} \quad (i = 1, \ldots, n). \tag{5.3.27}$$

This can be achieved by working "down" on the indices $n, n-1, \ldots$: First, we write

$$b = \sum_{i=1}^{n} \lambda_i^0 b_i^*.$$

Then we subtract $\lceil \lambda_n^0 \rfloor b_n$ to get a representation

$$b - \lceil \lambda_n^0 \rfloor b_n = \sum_{i=1}^{n} \lambda_i^1 b_i^*$$

where $|\lambda_n^1| \leq 1/2$. Next we subtract $\lceil \lambda_{n-1}^1 \rfloor b_{n-1}$ etc. (The procedure is essentially the same as Step (I) of the proof of Theorem (5.3.16).)

Now we claim that if w satisfies (5.3.27) then it is an "approximately nearest" lattice vector in the sense formulated in the theorem. For, let v be any other lattice vector and write

$$b - v = \sum_{i=1}^{n} \mu_i b_i^*.$$

Let k be the largest index such that $\mu_k \neq \lambda_k$. Then

$$v - w = \sum_{i=1}^{k} (\lambda_i - \mu_i) b_i^* \in L$$

and hence it follows that $\lambda_k - \mu_k$ is a nonzero integer, in particular $|\lambda_k - \mu_k| \geq 1$ and hence $|\mu_k| \geq 1/2$. So

$$\|v - b\|^2 \geq \sum_{i=k+1}^{n} \mu_i^2 \|b_i^*\|^2 + \frac{1}{4}\|b_k^*\|^2 = \sum_{i=k+1}^{n} \lambda_i^2 \|b_i^*\|^2 + \frac{1}{4}\|b_k^*\|^2$$

and so

$$\begin{aligned} \|w - b\|^2 &\leq \sum_{i=k+1}^{n} \lambda_i^2 \|b_i^*\|^2 + \frac{1}{4}\sum_{i=1}^{k} \|b_i^*\|^2 \\ &\leq \sum_{i=k+1}^{n} \lambda_i^2 \|b_i^*\|^2 + \frac{1}{4}\|b_k^*\|^2 \left(\sum_{i=1}^{k} 2^{k-i}\right) \\ &< 2^{k-2}\|v - b\|^2 \leq 2^{n-2}\|v - b\|^2. \end{aligned}$$

□

*5.4 More on Lattice Algorithms

It appears that the algorithmic theory of lattices in $\mathbb{Q}^n$ is less developed and understood than the algorithmic theory of convex bodies. We discuss a few questions that can be answered by the methods developed above. The questions are lattice analogues of the oracle results for convex sets in Chapter 4.

Let L be a lattice in $\mathbb{R}^n$. Its **dual lattice** L^* is defined as follows:

$$L^* = \{x \in \mathbb{R}^n \mid x^T y \in \mathbb{Z} \text{ for all } y \in L\}. \tag{5.4.1}$$

It is easy to see that L^* is a lattice. Furthermore, if $(b_1, \ldots, b_n)$ is any basis of L then the vectors $c_1, \ldots, c_n$ defined by

$$c_i^T b_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5.4.2}$$

form a basis of L^*. We shall call $(c_1, \ldots, c_n)$ the **dual basis** of $(b_1, \ldots, b_n)$. It follows that $L^{**} = L$ and $\det L \cdot \det L^* = 1$.

A lattice may be viewed as "nicely described" if a basis of it is known. Often, however, we have other descriptions. A general form of a definition of a lattice L is an oracle which checks, for any $b \in \mathbb{Q}^n$, whether or not $b \in L$, and if not, gives a vector $c \in \mathbb{Q}^n$ such that $c \in L^*$ and $b^T c \notin \mathbb{Z}$. Such an oracle will be called a **separation oracle** for L.

One may also want to define an analogue of a violation oracle for lattices; this is, however, just a separation oracle for the dual lattice.

Similarly as in the case of convex bodies, we need a technical hypothesis about the lattices we consider. A **well-described lattice** is a triple $(L; n, T)$ where L is a lattice in $\mathbb{Q}^n$, and $T \in \mathbb{N}$ is an upper bound for all denominators of entries of vectors in L and L^*. If L is given by n linearly independent vectors, then an integer T can be easily computed in polynomial time such that $(L; n, T)$ is a well-described lattice. But if the lattice is given by a separation oracle, then it is necessary to make some assumption on its arithmetical nature. The **encoding length** of $(L; n, T)$ is $n + \langle T \rangle$.

(5.4.3) Theorem. *A basis of a well-described lattice $(L; n, T)$ given by a separation oracle can be found in oracle-polynomial time.*

Proof. We use the following notation. Let $a_1, \ldots, a_k \in \mathbb{R}^n$. Then we set

$$\Delta(a_1, \ldots, a_k) := (\det((a_i^T a_j)_{i,j=1}^k))^{1/2}. \tag{5.4.4}$$

Geometrically, $\Delta(a_1, \ldots, a_k)$ is the k-dimensional Lebesque measure of the parallelepiped spanned by $a_1, \ldots, a_k$. Note that in the case when $k = n$, we have

$$\Delta(a_1, \ldots, a_n) = |\det(a_1, \ldots, a_n)|. \tag{5.4.5}$$

In the algorithm that follows, many of the complications arise from the fact that we have to control the encoding lengths of the numbers that occur. A similar situation comes up in the related algorithm of KANNAN and BACHEM (1979).

We shall maintain a list $C = \{c_1, \ldots, c_k\}$ of linearly independent vectors in L^*. At each "step" we either increase $|C|$ or keep $|C|$ invariant but decrease $\Delta(c_1, \ldots, c_k)$. There is some difference between the cases $k < n$ and $k = n$, and accordingly, we distinguish two phases.

Phase 1. $k < n$. Compute a vector $b = (\beta_1, \ldots, \beta_n)^T \in \mathbb{Q}^n$ such that $b^T c_1 = \ldots = b^T c_k = 0$ but $b \neq 0$. We may assume without loss of generality that the largest denominator M of the entries $\beta_1, \ldots, \beta_n$ satisfies $T < M \leq 2T$ (find first an integral vector b with the desired properties and then divide by an appropriate power of 2).

Let Q be a common denominator of the entries of $c_1, \ldots, c_k$, and define $\delta_i := \beta_i - Q\lfloor \beta_i / Q \rfloor$, and $d := (\delta_1, \ldots, \delta_n)^T$. Then we have

$$d^T c_j = b^T c_j - \sum_{i=1}^n Q c_{ij} \left\lfloor \frac{\beta_i}{Q} \right\rfloor \in \mathbb{Z},$$

denoting $c_j =: (c_{1j}, \ldots, c_{nj})^T$. Furthermore, since the largest denominator in d (which is the same as the largest denominator in b) is larger than T, we have that $d \notin L$. So if we call the separation oracle with input d, it gives us a vector $c \in L^*$ such that $d^T c \notin \mathbb{Z}$.

We now check whether c is linearly independent from $c_1, \ldots, c_k$.

Case 1. The vector c is linearly independent from $c_1, \ldots, c_k$. Then we add $c_{k+1} := c$ to the list C. This step is called an **extension step**.

Case 2. The vector c is linearly dependent from $c_1, \ldots, c_k$. Then we can write

$$c = \sum_{i=1}^{k} \lambda_i c_i, \ \lambda_i \in \mathbb{Q}.$$

Consider

$$c' := \sum_{i=1}^{k} (\lambda_i - \lceil \lambda_i \rfloor) c_i = c - \sum_{i=1}^{k} \lceil \lambda_i \rfloor c_i.$$

Then $c' \in L^*$ and furthermore,

$$d^T c - d^T c' = \sum_{i=1}^{k} \lceil \lambda_i \rfloor d^T c_i \in \mathbb{Z}$$

and so $d^T c' \notin \mathbb{Z}$. Hence it follows that for at least one i $(1 \leq i \leq k)$, $\lambda_i - \lceil \lambda_i \rfloor \neq 0$. Consider the largest such i. Then $c_1, \ldots, c_{i-1}, c', c_{i+1}, \ldots, c_k$ are linearly independent and for all $i \leq j \leq k$

$$\begin{aligned} \Delta(c_1, \ldots, c_{i-1}, c', c_{i+1}, \ldots, c_j) &= |\lambda_i - \lceil \lambda_i \rfloor| \Delta(c_1, \ldots, c_j) \\ &\leq \frac{1}{2} \Delta(c_1, \ldots, c_j). \end{aligned}$$

Now replace c_i by c'. This step is called an **exchange step**.

Thus in the case $k < n$, we could either extend C to a system of $k+1$ linearly independent vectors in L^* or else exchange a vector so that $\Delta(c_1, \ldots, c_k)$ decreases by a factor $< 1/2$.

Phase 2. Suppose that $k = n$. Then consider the vectors b_i defined by $b_i^T c_j = \delta_{ij}$ (i. e., the basis of $\mathbb{Q}^n$ dual to $\{c_1, \ldots, c_n\}$). Check whether $b_i \in L$ for $i = 1, \ldots, n$.

Case 1. Suppose that $b_i \in L$ for $i = 1, \ldots, n$. Then $b_1, \ldots, b_n$ is a basis of L. To show this, let $a \in L$ and write

$$a = \sum_{i=1}^{n} \alpha_i b_i \quad (\text{with } \alpha_i \in \mathbb{Q}, \ i = 1, \ldots, n).$$

Then $\alpha_i = a^T c_i \in \mathbb{Z}$ and so indeed every vector of L is an integer linear combination of $b_1, \ldots, b_n$. So in this case we stop.

Case 2. Suppose that $b_i \notin L$ for some i. Then the separation oracle gives a vector $c \in L^*$ such that $b_i^T c \notin \mathbb{Z}$. From here, we go on like in phase 1 and obtain a new list C such that $\Delta(c_1, \ldots, c_n)$ decreases by a factor $< 1/2$. We call this operation a **phase 2 exchange step**.

This completes the description of the algorithm.

We have to prove that it terminates in a polynomial number of steps and also that the encoding lengths of the numbers occurring are bounded by a polynomial in n and $\langle T \rangle$. The proofs of these two facts are somewhat intertwined.

We know by the General Assumption (1.2.1) about oracles that there exists a polynomial Φ such that for any vector $b \in \mathbb{Q}^n$ with $b \notin L$ the separation oracle gives a vector $c \in L^*$ such that $c^T b \notin \mathbb{Z}$ and $\langle c \rangle \leq \Phi(\langle b \rangle)$. Without loss of generality, $\Phi(x) \geq x$ for all $x \geq 0$.

First we estimate the vectors d used in phase 1. The numbers δ_i satisfy $|\delta_i| \leq Q \leq T^{n^2}$ and the denominator of δ_i is at most $2T$. Hence

$$\langle \delta_i \rangle \leq \langle 2T \rangle + \langle 2T \cdot T^{n^2} \rangle < 2n^2 \langle T \rangle,$$

and so

$$\langle d \rangle \leq 2n^3 \langle T \rangle.$$

Note that this bound holds uniformly for all vectors d that occur. Hence by the definition of Φ, we have for any output c of the separation oracle

$$\langle c \rangle \leq \Phi(\langle d \rangle) \leq \Phi(2n^3 \langle T \rangle) =: \Psi,$$

and so

$$\|c\| \leq 2^{\Psi}.$$

Thus at an extension step, we get

$$\langle c_{k+1} \rangle \leq \Psi$$

and so

$$\|c_{k+1}\| \leq 2^{\Psi}.$$

Next we show by induction on k that, at any stage of the algorithm,

$$\Delta(c_1, \ldots, c_k) \leq 2^{k\Psi}. \tag{5.4.6}$$

Suppose first that c_k was just added. Then c_k is an output of the separation oracle and so $\|c_k\| \leq 2^{\Psi}$. Thus by induction,

$$\begin{aligned} \Delta(c_1, \ldots, c_k) &\leq \Delta(c_1, \ldots, c_{k-1}) \cdot \|c_k\| \\ &\leq 2^{(k-1)\Psi} \cdot 2^{\Psi} = 2^{k\Psi}. \end{aligned}$$

Further extension steps or exchange steps of either phase cannot increase $\Delta(c_1, \ldots, c_k)$.

We have, from the definition of Q, that $\Delta(Qc_1, \ldots, Qc_k) \in \mathbb{Z}$ and hence

$$Q^k \cdot \Delta(c_1, \ldots, c_k) = \Delta(Qc_1, \ldots, Qc_k) \geq 1.$$

Therefore

$$\Delta(c_1, \ldots, c_k) \geq Q^{-k} \geq T^{-n^2k}. \tag{5.4.7}$$

It follows from inequalities (5.4.6) and (5.4.7) that between two extension steps, the number of exchange steps is at most

$$\log_2(2^{k\Psi}/T^{-n^2k}) = k\Psi + n^2k \log_2 T < 2k\Psi \leq 2n\Psi.$$

A similar bound holds for the number of exchange steps in phase 2. Hence it follows that the total number of steps is at most $2n^2\Psi$.

Finally, we have to show that the encoding length for the vectors in C remains bounded by a polynomial in n and $\langle T \rangle$. This is clear for those vectors which are included in C as the result of an extension step. To handle exchange steps, note the following: at an exchange step, the value $\max\{\|c\| \mid c \in C\}$ increases by a factor $< n$. In fact, the new vector c' to be included in C can be written as

$$c' = \sum_{i=1}^{k} (\lambda_i - \lceil \lambda_i \rfloor) c_i$$

and thus

$$\begin{aligned} \|c'\| &\leq \sum_{i=1}^{k} |\lambda_i - \lceil \lambda_i \rfloor| \cdot \|c_i\| \\ &\leq k \cdot \max\{\|c_i\| \mid 1 \leq i \leq k\} \\ &\leq n \cdot \max\{\|c_i\| \mid 1 \leq i \leq k\}. \end{aligned}$$

Hence it follows by induction that any vector c' inserted in C after t steps satisfies

$$\|c'\| \leq n^t 2^{\Psi} \leq n^{2n^2\Psi} 2^{\Psi}.$$

Since $c' \in L^*$, the denominators of the entries of c' are bounded by T, and hence

$$\begin{aligned} \langle c' \rangle &\leq n(\langle T \rangle + \langle T \cdot n^{2n^2\Psi} 2^{\Psi} \rangle) \\ &\leq 2n\langle T \rangle + 4n^3\Psi. \end{aligned}$$

□

(5.4.8) Corollary. *For any given $a_1, \ldots, a_m \in \mathbb{Q}^n$, a basis of the lattice $L(a_1, \ldots, a_m)$ can be found in polynomial time.*

Proof. We may assume without loss of generality that $L = L(a_1, \ldots, a_m)$ is a full-dimensional lattice. Then a separation algorithm for the dual lattice L^* can be designed easily. In fact, to check whether $b \in L^*$ it suffices to evaluate the

numbers $b^T a_1, \ldots, b^T a_m$. If all these numbers are integral, then $b \in L^*$. If $b^T a_i$ is not integral for some $i \in \{1, \ldots, m\}$ then $a_i \in L$ is a valid output for the separation algorithm.

Moreover, if Q is the least common denominator of the entries of $a_1, \ldots, a_m$ then the denominator of any entry of any vector in L is a divisor of Q. Moreover, if $a_1, \ldots, a_n$ are linearly independent (say) then the denominator of any entry of any vector in L^* is a divisor of $Q^n \det(a_1, \ldots, a_n)$. So if we take

$$T := \max\{Q, Q^n \det(a_1, \ldots, a_n)\}$$

then $(L^*; n, T)$ is a well-described lattice.

Hence Theorem (5.4.3) implies that bases of L and L^* can be constructed in polynomial time. □

As an application of this, we obtain that a system of linear diophantine equations can be solved in polynomial time (VON ZUR GATHEN and SIEVEKING (1976), FRUMKIN (1976a,b)).

(5.4.9) Corollary. *There exists a polynomial time algorithm that, for any given vectors $a_1, \ldots, a_m, c \in \mathbb{Q}^n$, decides whether $c \in \mathrm{L}(a_1, \ldots, a_m)$, i. e., whether the system of linear diophantine equations*

$$x_1 a_1 + \ldots + x_m a_m = c$$

is solvable in integers, and that finds a solution of this equation if one exists. □

(5.4.10) Corollary. *For any matrix $A \in \mathbb{Q}^{m \times n}$, a basis for the integral solutions of the system $Ax = 0$ can be found in polynomial time.* □

As an application of the basis reduction algorithm, we give an algorithm to transform a matrix into Hermite normal form. The first polynomial time algorithm to achieve this has been designed by FRUMKIN (1976c).

Let A be a nonsingular $m \times m$-matrix. Recall from Section 1.4 that the **Hermite normal form** of A is an $m \times m$-matrix $H = (h_{ij})$ obtained from A by elementary column operations (i. e., by adding a column to another, or multiplying a column by -1) with the following properties:

$$h_{ij} = 0 \quad \text{if } j > i, \tag{5.4.11}$$

$$0 \leq h_{ij} < h_{ii} \quad \text{if } j < i, \tag{5.4.12}$$

and that the Hermite normal form of A is unique.

(5.4.13) Theorem. *The Hermite normal form of a nonsingular rational matrix A can be obtained in polynomial time.*

Proof. Without loss of generality assume that A is integral. Let

$$M := 2^{\binom{n}{2}} |\det A|.$$

Let us multiply the i-th row of A by M^{n-i}, $i = 1, \ldots, n$, to get a matrix A'. Let L be the lattice generated by the columns of A'.

Construct a basis $\{b_1, \ldots, b_n\}$ of L such that

$$\|b_1\| \cdot \ldots \cdot \|b_n\| \leq 2^{\frac{1}{2}\binom{n}{2}} \det L = 2^{\frac{1}{2}\binom{n}{2}} M^{\binom{n}{2}} |\det A| \tag{5.4.14}$$

using the algorithm of Theorem (5.3.16). We claim that $\{b_1, \ldots, b_n\}$ can be reordered so that we get a lower triangular matrix.

Since

$$|\det(b_1, \ldots, b_n)| = \det L \neq 0,$$

there exists a reordering of this basis so that $b_{ii} \neq 0$, for $i = 1, \ldots, n$. We claim that if $1 \leq i < j \leq n$ then $b_{ij} = 0$. Suppose there are i,j with $1 \leq i < j \leq n$ such that $b_{ij} \neq 0$; then

$$\begin{aligned} \|b_1\| \cdot \ldots \cdot \|b_n\| &= \|b_j\| \prod_{k \neq j} \|b_k\| > |b_{ij}| \cdot \prod_{k \neq j} |b_{kk}| \geq M^{n-i} \prod_{k \neq j} M^{n-k} \\ &\geq M \cdot \prod_{k=1}^{n} M^{n-k} \\ &= M^{\binom{n}{2}+1}. \end{aligned}$$

This contradicts (5.4.14).

So we have transformed A', and of course also A, into a lower triangular form. Condition (5.4.12) can be easily achieved for $i = n, n-1, \ldots, 1$. □

Chapter 6

Rational Polyhedra

In most combinatorial (and real world) applications the convex sets one encounters are polyhedra. Often these polyhedra have "simple" vertices and facets. It turns out that the knowledge of such additional information on the convex sets in question extends the power of the ellipsoid method considerably. In particular, optimum solutions can be calculated exactly, boundedness and full-dimensionality assumptions can be dropped, and dual solutions can be obtained. In the case of explicitly given linear programs this was the main contribution of Khachiyan to the ellipsoid method. If the linear programs are given by some oracle – which is often the case in combinatorial optimization – then these additional goals can still be achieved, albeit with more involved techniques. In particular, we have to make use of the simultaneous diophantine approximation algorithm described in Chapter 5.

As we did before, in the proofs we assume that $n \geq 2$, if the one-variable case is trivial.

6.1 Optimization over Polyhedra: A Preview

In Chapters 3 and 4 we have seen how the ellipsoid method can be used to solve various algorithmic problems concerning convex sets. These results, however, had some limitations. First, we could only solve the problems in an approximate sense, that is, we could only treat the weak versions of the problems. Second, we had to make various restrictions on the convex sets in question: they had to be bounded and, for many results, also full-dimensional in a very explicit sense. We showed that these restrictions cannot be removed in general.

Most (although not all) applications of the ellipsoid method in combinatorial optimization concern, however, polyhedra which have integral or half-integral vertices, or at least vertices whose entries are rational numbers with relatively small encoding length. On the other hand, boundedness and full-dimensionality are not always natural conditions (although often they can be guaranteed by *ad hoc* methods). The aim of this chapter is to discuss versions of the previous results which show that if we know an upper bound on the encoding lengths of the vertices or facets of a polyhedron then the solvability of the weak problems implies the solvability of the strong problems in polynomial time. We shall also see that the conditions on boundedness and full-dimensionality can be dropped.

Let us preview some of the results of this chapter concerning optimization problems. The most important one is:

– *Linear programs can be solved in polynomial time.*

This theorem, due to KHACHIYAN (1979), is stated in (6.4.12). In fact, an optimum vertex solution can be found in polynomial time, if such a solution exists, as is shown in Remark (6.5.2). These results will be strengthened in various ways. One of them was developed by TARDOS (1986) who showed (see (6.6.3)):

– *Any linear program* $\max\{c^T x \mid Ax \leq b\}$ *can be solved by a polynomial time algorithm, which only performs a number of elementary arithmetic operations bounded by a polynomial in* $\langle A \rangle$.

This result sharpens Khachiyan's since here the number of elementary arithmetic operations is independent of the encoding lengths of b and c. Tardos' result, for instance, implies that network flow problems can be solved in strongly polynomial time.

Most of the effort in this chapter is spent on extending Khachiyan's result in another direction. Roughly speaking, one of the main outcomes is the following:

– *There exists a polynomial time algorithm that, for any polyhedron* $P \subseteq \mathbb{R}^n$ *and for any* $c \in \mathbb{Q}^n$, *solves* $\max\{c^T x \mid x \in P\}$, *provided the following two conditions are satisfied:*

(i) *the strong separation problem for* P *can be solved in polynomial time;*
(ii) *a number* $\varphi \in \mathbb{N}$ *is given such that* P *can be defined by linear inequalities each having encoding length at most* φ.

(Again, this result can be sharpened in the way Tardos' result sharpens Khachiyan's – see (6.6.5).) This result follows from Theorem (6.4.9), one of the central theorems of this book, which reads as follows:

– *Any one of the following three problems:*
  – *strong separation,*
  – *strong violation,*
  – *strong optimization,*

  can be solved in oracle-polynomial time for any "well-described" polyhedron given by an oracle for any of the other two problems.

The term "well-described" will be defined in (6.2.2) – it amounts to knowing an upper bound φ as in (ii) above.

Moreover, we will show that for a linear program $\max\{c^T x \mid x \in P\}$ optimum dual solutions can be found in oracle-polynomial time if conditions (i) and (ii) above are satisfied. For precise statements see (6.5.14), (6.5.17), and (6.6.5).

In the remaining part of this first section we illustrate how the results mentioned above can be achieved by combining the methods of Chapter 5 with the general results on the ellipsoid method. The subsequent sections contain the details which are often quite elaborate. So let us make some remarks about the additional goals and difficulties one by one.

(6.1.1) *How can we solve the strong problems if we know how to solve the weak ones, i. e., how can we obtain exact solutions from approximations?*

If we know how to solve, say, the weak optimization problem over a polyhedron P, i. e., if we know how to find a vector "almost in P" that "almost" maximizes a given linear objective function over P, then it is a natural idea to obtain the maximizing vertex by rounding. This is indeed the approach we shall take; in fact, this is the path Khachiyan took when he applied the ellipsoid method to linear programming. The rounding has to be carried out with some care though. Namely, if there is a vertex that is not optimal, but nearly optimal, then the solution to the weak optimization problem found by the ellipsoid method may be very close to this "near-optimal" vertex and rounding may not take us to the true optimum. This difficulty could be overcome by perturbing the objective function, and this method was used, e. g., in GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981). But a more elegant way is the use of simultaneous diophantine approximation for the rounding. Since this method is also used in the handling of non-full-dimensionality, where it seems to be vital, we shall discuss it in connection with the next question:

(6.1.2) *How can we handle polyhedra that are not full-dimensional?*

This question is harder than passing from weak to strong solutions, and no "elementary" trick is known to solve it. The method which we follow depends on how the polytope is given. For the purposes of this introduction, we shall consider just two oracles: separation and violation.

Assume first that P is a polytope in $\mathbb{R}^n$ given by a strong separation oracle and assume that we know that all the vertices of P are $\{0,1\}$-vectors. Let us consider the first "bad" case, i. e., when P is $(n-1)$-dimensional. Then P is contained in a unique hyperplane H. It can easily happen that the separation oracle returns separating hyperplanes parallel to H unless a vector in H is fed to it. So we must determine this hyperplane along the lines in order to make sure that we get nontrivial information concerning P.

Conversely, if we can determine H in polynomial time then the problem is reduced to a full-dimensional instance and we can use the methods of the previous chapters. Thus our goal is to find the hyperplane H in polynomial time. (More generally if we do not make any assumption on the dimension of P, then our goal is to determine the affine hull of P. But to give the idea of our approach, we will stick to the case of $(n-1)$-dimensional polyhedra.)

The algorithm which achieves this was described in GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1984a). It consists of a combination of the ellipsoid method with diophantine approximation. First, let us run the ellipsoid method as if we were trying to find a vector in P. So we obtain a sequence of ellipsoids including the polytope whose volume decreases by a fixed factor at each step. If P is full-dimensional then this sequence is finite. In fact it is not too difficult to show that $\mathrm{vol}(P) \geq 1/n!$, and from this, one can see that the procedure terminates in polynomial time. But if P is not full-dimensional, the centers of the ellipsoids will never belong to P unless by "accident", and so the sequence generally will

not terminate. (If this "accident" happens then we perturb the center a little to move it out of P.)

Now the sequence of ellipsoids whose volumes tend to 0 but which all include P must become very "flat" in the direction perpendicular to H; that is, for any of these ellipsoids, the symmetry hyperplane belonging to the shortest axis must be very close to H. So we could try to find H by rounding the equation of this symmetry hyperplane. It turns out that just straightforward rounding would not be enough, but if we apply the more involved techniques of Chapter 5 (simultaneous diophantine approximation), then we can indeed recover H.

The argument runs as follows. Assume that the hyperplane H is described by the equation $a^T x = \alpha$. We want to determine α and a. One thing we do know a priori is that the vertices of P span H and these vertices are $\{0,1\}$-vectors.

Now we compute an ellipsoid E of volume less than $2^{-n^2}(3n)^{-2n}$ including P by the ellipsoid method and determine its shortest axis. Let v be the center of E and let w be the unit vector parallel to the shortest axis of E. The length of this shortest axis is less than $\varepsilon := 2^{-n^2-1}(3n)^{-n-1}$ by a simple computation. The hyperplane H' defined by the equation $w^T x = w^T v$ will be a good approximation of H. More exactly, if u is any vector in E (in particular, if u is any vector in P) then we have

$$|w^T u - w^T v| = |w^T(u-v)| \leq \varepsilon.$$

To recover H from this hyperplane, let us approximate the vector $(w^T, w^T v)^T$ using simultaneous diophantine approximation, and find an integral vector $(p^T, \pi)^T$ $(p \in \mathbb{R}^n, \pi \in \mathbb{R})$ and a positive integer q such that

$$\|qw - p\|_\infty < \frac{1}{3n}, \quad |qw^T v - \pi| < \frac{1}{3n}$$

and

$$0 < q < 2^{n^2}(3n)^{n+1}.$$

We claim that the hyperplane defined by the equation $p^T x = \pi$ contains the polytope P. For, let z be any vertex of P. Then z is a $\{0,1\}$-vector and so

$$\begin{aligned} |p^T z - \pi| &\leq |qw^T z - qw^T v| + \frac{1}{3n}(\|z\|_1 + 1) \\ &\leq q|w^T z - w^T v| + \frac{1}{3n}(n+1) \\ &< q2^{-n^2-1}(3n)^{-n-1} + \frac{1}{3n}(n+1) \leq 1. \end{aligned}$$

Since the left hand side is an integer, this proves the assertion. So we have found the hyperplane containing P.

The case when P is given by a strong violation oracle can be reduced to the case when it is given by a strong separation oracle by a certain polarity construction. But in this section we describe another way to get around non-full-dimensionality in this case, which is much easier.

Again, it suffices to show how to find the affine hull of a polyhedron P given by a strong violation oracle. The algorithm to achieve this was described (in a somewhat different context) by EDMONDS, LOVÁSZ and PULLEYBLANK (1982).

At any stage of the algorithm, we shall have a list of affinely independent vectors $a_1, \ldots, a_k$ and a list of linearly independent hyperplanes $H_1, \ldots, H_l$ such that $a_1, \ldots, a_k \in P$ and $H_1, \ldots, H_l \supseteq P$. First, suppose that $k + l \leq n$. Then we compute a hyperplane $H = \{x \in \mathbb{R}^n \mid c^T x = \gamma\}$ that contains all the vectors $a_1, \ldots, a_r$ and is linearly independent of the hyperplanes $H_1, \ldots, H_l$. (This is elementary linear algebra.) Next, we call the strong violation oracle for both inequalities

$$c^T x \leq \gamma \quad \text{and} \quad c^T x \geq \gamma.$$

If both are satisfied for all $x \in P$, then we have $H \supseteq P$ and so we can append H to our list of hyperplanes as H_{l+1}. If, on the other hand, the oracle returns a vector $a \in P$ that violates one of these inequalities, then this vector can be appended to the list of vectors in P as a_{k+1}. So in each case we were able to increase $k + l$ by 1.

Eventually we reach $k + l = n + 1$. Then we have the affine hull of P (in fact, we have both a spanning set and a system of equations describing it).

(6.1.3) *How can we handle unbounded polyhedra?*

This is usually easy: since we know an upper bound on the encoding length of the vertices, all the vertices lie inside a cube which we can calculate. In the "combinatorial" case, when all vertices are $\{0, 1\}$-vectors, this could be the cube $Q := \{x \in \mathbb{R}^n \mid -1 \leq x_i \leq 2,\ i = 1, \ldots, n\}$. Then we replace the polyhedron P by its intersection with this cube, and this reduces the problem to the bounded case.

There is a technical point here: the polyhedron in $\mathbb{R}^n$, $n \geq 2$, defined by, say $x_1 \geq 10^{100}$ has $\{0, 1\}$-vertices (since it has no vertices at all), but by intersecting it with the cube Q we get the empty set; and we loose all information about the polyhedron itself. So, if we say that a polyhedron has $\{0, 1\}$-vertices, we tacitly assume that it intersects the unit cube (this is for the purposes of this preview; the precise definition of vertex-complexity, to be given in the next section, takes care of this difficulty).

To be more specific, assume that P is a polyhedron with $\{0, 1\}$-vertices given by a strong separation oracle and that we want to maximize a linear objective function $c^T x$ over P. Let P' be the intersection of P with the cube Q. Then it is trivial to design a strong separation algorithm for P'. Now let us optimize $c^T x$ over P'. By perturbing c a little, if necessary, we may assume that the optimum is attained at a unique vertex v of P'. Now if v is a vertex of P then it also maximizes our objective function over P. If, on the other hand, v is not a vertex of P, then it lies on the boundary of Q and from this it follows that $c^T x$ is unbounded on P.

(6.1.4) *Can we solve other problems with these methods?*

The remarks above indicate the methods that are used to exploit our two fundamental geometric algorithms, the ellipsoid method and diophantine approximation, for the solution of some basic questions concerning polyhedra. We employ these techniques in Sections 6.2, ..., 6.7 to solve a number of further interesting algorithmic problems concerning polyhedra. To mention a few, we

prove that the affine hull and the lineality space of a polyhedron can be computed in polynomial time, and we design a strong separation algorithm for the recession cone. We can find a vertex of a polyhedron P, if there is any, and decide whether or not two vertices are adjacent, in polynomial time. One can do Carathéodory's theorem algorithmically: given a polytope P by a strong separation oracle, a bound on its vertex-complexity, and a vector $y \in P$, we can find a representation of y as a convex combination of at most $\dim P + 1$ vertices of P.

The polar of this last result is of particularly great importance in combinatorial optimization. Suppose that the polyhedron P is given by a strong separation oracle. Then we may view this as a system of linear inequalities (all valid inequalities ever returned by the separation oracle), which is, however, not explicitly given. So optimizing a linear objective function over P is a linear program with this implicit constraint set. From the above we know that we can find an optimal primal solution for this problem in polynomial time. But using the above-mentioned algorithmic version of Carathéodory's theorem and polarity, we obtain a polynomial-time procedure to find an optimal dual solution as well. This result can be strengthened as follows. Suppose that $\gamma := \max\{c^T x \mid x \in P\}$ is finite. Then we can find inequalities $a_1^T x \leq \alpha_1, \ldots, a_k^T x \leq \alpha_k$ and positive rationals $\lambda_1, \ldots, \lambda_k$ in polynomial time such that $c = \lambda_1 a_1 + \ldots + \lambda_k a_k$, $\gamma = \lambda_1 \alpha_1 + \ldots + \lambda_k a_k$ and the inequalities $a_i^T x \leq \alpha_i$ are either implicit equations or define facets of P.

In Section 6.6 we show that a combination of simultaneous diophantine approximation and the ellipsoid method runs in strongly polynomial time for a number of interesting linear programs. In particular, the results of Section 6.6 yield that these techniques can be used to design strongly polynomial time algorithms for almost all of the combinatorial optimization problems discussed in Chapters 7 – 10. Finally, in Section 6.7 we derive from our two basic geometric algorithms the result of H. W. Lenstra, Jr., that, for any fixed number of variables, integer linear programs can be solved in polynomial time.

*6.2 Complexity of Rational Polyhedra

In Chapter 3 we restricted ourselves to bounded sets. This assumption will be removed in the case of polyhedra. (We could remove the boundedness assumption also for general convex sets but too many pathologies and technical difficulties would creep in; for instance a linear objective function could have a finite supremum but no optimum.) To cover the unbounded case we have to modify the definition of the strong optimization problem.

(6.2.1) The Strong Optimization Problem for Polyhedra. *Given a polyhedron $P \subseteq \mathbb{R}^n$ and a vector $c \in \mathbb{Q}^n$, either*

(i) *assert that P is empty, or*
(ii) *find a vector $y \in P$ maximizing $c^T x$ over P, or*
(iii) *find a vector $z \in \text{rec}(P)$ such that $c^T z \geq 1$* (i. e., a direction in which $c^T x$ is unbounded over P).

The strong versions of the other problems do not need any modification for the unbounded case. The weak versions will be discussed later.

Analogous to our definitions of various convex bodies (2.1.16) we now introduce the notion of "well-describedness" of polyhedra.

(6.2.2) Definition. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let φ and ν be positive integers.*

(a) *We say that P has* **facet-complexity at most** *φ if there exists a system of inequalities with rational coefficients that has solution set P and such that the encoding length of each inequality of the system is at most φ. In case $P = \mathbb{R}^n$ we require $\varphi \geq n+1$.*

(b) *We say that P has* **vertex-complexity at most** *ν if there exist finite sets V, E of rational vectors such that $P = \operatorname{conv}(V) + \operatorname{cone}(E)$ and such that each of the vectors in V and E has encoding length at most ν. In case $P = \emptyset$ we require $\nu \geq n$.*

(c) *A* **well-described polyhedron** *is a triple $(P; n, \varphi)$ where $P \subseteq \mathbb{R}^n$ is a polyhedron with facet-complexity at most φ. The* **encoding length** *$\langle P \rangle$ of a well-described polyhedron $(P; n, \varphi)$ is $\varphi + n$.* □

It is obvious that if a polyhedron $P \subseteq \mathbb{R}^n$ has facet-complexity at most φ and vertex-complexity at most ν then

$$\varphi \geq n+1 \quad \text{and} \quad \nu \geq n.$$

In the sequel we will frequently encounter polyhedra P for which we know an upper bound φ on the facet-complexity but for which we do not have explicitly a system $Ax \leq b$ defining P. Even if we know a system defining P, its inequalities may not necessarily have encoding length at most φ. But if we turn this system into a standard representation of P – as defined in Section 0.1 – we obtain a system with small encoding length of each inequality resp. equation.

(6.2.3) Lemma. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron with facet-complexity at most φ. Then each inequality in any standard description of P has encoding length at most $35n^2\varphi$.*

Proof. We may assume that $P = \{x \in \mathbb{R}^n \mid Cx = d, Ax \leq b\}$ is a representation of P such that each equation and inequality has encoding length at most φ. We may also assume that $\operatorname{aff}(P) = \{x \mid Cx = d\}$, C has full row rank m, the submatrix F of C consisting of the first m columns is nonsingular and each inequality in the system $Ax \leq b$ defines a facet of P.

Let $P = \{x \in \mathbb{R}^n \mid (I, C')x = d', A'x \leq b'\}$ be any standard representation of P. Then

$$(I, C') = F^{-1}C, \; d' = F^{-1}d,$$
$$A' = A - AC^T(CC^T)^{-1}C, \; b' = b - AC^T(CC^T)^{-1}d.$$

Using the formulas for the encoding length in (1.3.5) the result follows. □

The reader may wonder why we do not use vertex-complexity in the definition of well-described polyhedra. The next lemma shows that from the point of view of polynomial algorithms both measures of complexity are equivalent.

(6.2.4) Lemma. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron.*

(a) *If P has facet-complexity at most φ, then P has vertex-complexity at most $4n^2\varphi$.*

(b) *If P has vertex-complexity at most ν, then P has facet-complexity at most $3n^2\nu$.*

Proof. (a) Suppose that P has facet-complexity at most φ, i. e., there are inequalities $a_i^T x \leq b_i$ ($a_i \in \mathbb{Q}^n$, $b_i \in \mathbb{Q}$, $i = 1, \ldots, m$) defining P such that each inequality has encoding length at most φ.

It is well known that there exist finite sets $V, E \subseteq \mathbb{Q}^n$ with $P = \text{conv}(V) + \text{cone}(E)$ such that every nonzero entry of every vector in $V \cup E$ is the quotient of two subdeterminants, of order at most n, of the matrix

$$C := \begin{pmatrix} a_1^T, b_1 \\ \vdots \\ a_m^T, b_m \end{pmatrix}.$$

Let $D = \begin{pmatrix} d_1 \\ \vdots \\ d_k \end{pmatrix}$ be a square submatrix of C, $k \leq n$. Then by Lemma (1.3.4) (b)

$$\langle \det D \rangle \leq 2\langle D \rangle \leq 2n\varphi.$$

Thus every entry of every vector in $V \cup E$ has encoding length at most $4n\varphi$ and hence every vector in $V \cup E$ has encoding length at most $4n^2\varphi$.

(b) Suppose that P has vertex-complexity at most ν. If $P = \emptyset$ or $P = \{0\}$ the assertion is trivial. In any other case $\nu \geq n + 1$. Let us assume first that P is full-dimensional. Choose finite sets of vectors $V, E \subseteq \mathbb{Q}^n$ such that $P = \text{conv}(V) + \text{cone}(E)$ and every vector in $V \cup E$ has encoding length at most ν. Every facet of P determines a hyperplane given by a linear equation of the form

$$\det \begin{pmatrix} 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \\ x_1 & & & & & & \\ \vdots & & & & & & \\ & v_1 & \ldots & v_k & e_1 & \ldots & e_{n-k} \\ x_n & & & & & & \end{pmatrix} = 0,$$

where $v_1, \ldots, v_k \in V$ and $e_1, \ldots, e_{n-k} \in E$. Expanding the determinant by the first column, we get

$$\sum_{i=1}^{n} (-1)^i \det(D_i) x_i = -\det D_0$$

where D_i is the matrix obtained by deleting the $(i+1)$-st row from the matrix

$$D = \begin{pmatrix} 1 & \ldots & 1 & 0 & \ldots & 0 \\ v_1 & \ldots & v_k & e_1 & \ldots & e_{n-k} \end{pmatrix}.$$

Using Lemma (1.3.4), we estimate the encoding length of this equation as follows:

$$\begin{aligned}\sum_{i=0}^{n}\langle \det D_i\rangle &\leq \sum_{i=0}^{n}(2\langle D_i\rangle - n^2)\\ &\leq 2n\langle D\rangle - (n+1)n^2\\ &\leq 2n(nv+2n) - (n+1)n^2\\ &= 2n^2v - n^2((n+1)-4).\end{aligned}$$

The last expression is clearly at most $3n^2v$.

Second, if P is not full dimensional, then consider the polyhedra $P + S_i$ where $S_1, \ldots, S_{2^n}$ are the 2^n simplices of $\mathbb{R}^n$ spanned by the zero vector and n linearly independent vectors of the vectors $(0, \ldots, \pm 1, \ldots, 0)^T$. Obviously for every i, $P + S_i$ is a full-dimensional polyhedron with vertex-complexity at most $v + 1$. By the previous part of the proof, $P + S_i$ has facet-complexity at most $2n^2(v+1) - n^2((n+1)-4) = 2n^2v - n^2((n+1)-6)$ and as $v \geq n+1 \geq 3$, this is at most $3n^2v$. Since

$$P = \bigcap_{i=1}^{2^n}(P + S_i)$$

it follows that P has facet-complexity at most $3n^2v$. □

The following two lemmas show that bounded full-dimensional well-described polyhedra may be viewed as well-bounded convex bodies.

(6.2.5) Lemma. *Let P be a polyhedron with vertex-complexity at most v. Then all vertices of P are contained in the ball $S(0, 2^v)$. If P is bounded then $P \subseteq S(0, 2^v)$.*

Proof. Trivial. □

(6.2.6) Lemma. *Let $P \subseteq \mathbb{R}^n$ be a full-dimensional polyhedron with facet-complexity at most φ. Then P contains a ball with radius $2^{-7n^3\varphi}$. Moreover, this ball is contained in $S(0, 2^{5n^2\varphi})$.*

Proof. By Lemma (6.2.4) (a), P has vertex-complexity at most $v := 4n^2\varphi$. So there are (minimal) sets $V, E \subseteq \mathbb{Q}^n$ such that $P = \mathrm{conv}(V) + \mathrm{cone}(E)$ and each vector in $V \cup E$ has encoding length at most v.

By the full-dimensionality of P there exist an affinely independent subset $\{v_1, \ldots, v_k\}$ of V and a linearly independent subset $\{e_1, \ldots, e_t\}$ of E such that $k + t = n + 1$ and the vectors $v_1, \ldots, v_k, v_1 + e_1, \ldots, v_1 + e_t$ span an n-dimensional simplex.

Consider the barycenter c of this simplex, i. e.,

$$c := \frac{1}{n+1}\left(\sum_{i=1}^{k} v_i + \sum_{j=1}^{t}(v_1 + e_j)\right).$$

Clearly, c is an interior point of P. We claim that the ball $S(c, 2^{-7n^3\varphi})$ is contained in P. To prove this we show that the distance of c from every facet of P is at least $2^{-7n^3\varphi}$. So, let $\bar{a}^T x \leq \bar{b}$ define a facet of P which we may assume to have encoding length at most φ. Multiplying this inequality with the product of the denominators of the coefficients of $\bar{a}$ and $\bar{b}$ gives an inequality $a^T x \leq b$ with integral coefficients and encoding length at most $n\varphi$. The distance of c from this facet is

$$\frac{b - a^T c}{\|a\|}.$$

Here $b - a^T c$ is a positive rational number whose denominator is at most the least common denominator of the coefficients of c (as b and a are integral), which in turn is at most $(n+1)$ times the least common denominator of the entries of $v_1, \ldots, v_k, e_1, \ldots, e_t$. This denominator is at most $2^{(n+1)\nu}$. Therefore,

$$b - a^T c \geq 2^{-(n+1)\nu}/(n+1).$$

Since furthermore

$$\|a\| \leq 2^{\langle a \rangle} \leq 2^{n\varphi},$$

we get that the distance of c from the facet defined by $a^T x \leq b$ is at least $2^{-(n+1)\nu - n\varphi}/(n+1) = 2^{-4(n+1)n^2\varphi - n\varphi}/(n+1) \geq 2^{-7n^3\varphi}$.

To prove the second assertion of the Lemma, observe that

$$\|c\| \leq 2 \max\{\|v_i\|, \|e_j\| \mid 1 \leq i \leq k, 1 \leq j \leq t\} \leq 2 \cdot 2^{\nu}.$$

So, $S(c, 2^{-7n^3\varphi})$ is contained in $S(0, 2^{\nu+1} + 2^{-7n^3\varphi}) \subseteq S(0, 2^{5n^2\varphi})$. □

The above estimates can be improved considerably, but we have chosen a quick and rather straightforward way to derive polynomial bounds. The main trick to make use of the rationality hypothesis is summarized in the following lemma.

(6.2.7) Lemma. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron with facet-complexity at most φ and let $y \in \mathbb{Q}^n$ be a vector whose entries have common denominator at most q. Suppose that*

$$y \in S(P, \frac{1}{q} 2^{-2\varphi}).$$

Then $y \in P$.

Proof. Let $a^T x \leq b$ be any valid inequality for P with encoding length at most φ. Then the hypothesis that $y \in S(P, 2^{-2\varphi}/q)$ and Lemma (1.3.3) imply

$$a^T y - b \leq \frac{1}{q} 2^{-2\varphi} \|a\| < \frac{1}{q} 2^{-\varphi}.$$

But $a^T y - b$ is a rational number with denominator at most $q 2^{\varphi}$. Hence $a^T y \leq b$. □

The following polar form of Lemma (6.2.7) is proved in the same way.

(6.2.8) Lemma. *Let P be a polyhedron with vertex-complexity at most v and let $a^T x \leq b$ be an integral inequality such that*

$$a^T x \leq b + 2^{-v-1}$$

is valid for P. Then $a^T x \leq b$ is also valid for P. □

Lemma (6.2.7) says that if a point close to a polyhedron has small enough encoding length, then it is in fact contained in the polyhedron. What we describe next is that, if a point is close to a polyhedron, then we try to round it to a nearby point with sufficiently small encoding length, which would therefore belong to the polyhedron. This can be achieved with the method of simultaneous diophantine approximation derived in Chapter 5.

It is interesting to notice that this rounding procedure is independent of the polyhedron (it only depends on the complexity of the polyhedron). Similarly, almost valid inequalities can be rounded to valid ones.

(6.2.9) Lemma.

(a) *Let $v \in \mathbb{R}^n$ and φ be a positive integer. Suppose $q \in \mathbb{Z}$ and $w \in \mathbb{Z}^n$ satisfy*

$$\|qv - w\| < 2^{-3\varphi} \quad \text{and}$$

$$0 < q < 2^{4n\varphi}.$$

Then $\frac{1}{q}w$ is contained in every polyhedron P of facet-complexity at most φ for which $v \in S(P, 2^{-6n\varphi})$.

(b) *Let $a \in \mathbb{R}^n$, $b \in \mathbb{R}$ and let v be a positive integer. Suppose that $q \in \mathbb{Z}$, $c \in \mathbb{Z}^n$ and $d \in \mathbb{Z}$ satisfy*

$$\|qa - c\| + |qb - d| < 2^{-3v}$$

and

$$0 < q < 2^{4nv}.$$

Then the inequality $c^T x \leq d$ is valid for every polyhedron P of vertex-complexity at most v for which $a^T x \leq b + 2^{-6nv}$ is a valid inequality.

Proof. (a) By Lemma (6.2.7) it suffices to show that $\frac{1}{q}w \in S(P, \frac{1}{q}2^{-2\varphi})$. This is clear since

$$\begin{aligned} d(\frac{1}{q}w, P) &\leq \|\frac{1}{q}w - v\| + d(v, P) \\ &\leq \frac{1}{q}2^{-3\varphi} + 2^{-6n\varphi} \\ &\leq \frac{1}{q}2^{-2\varphi}. \end{aligned}$$

(b) Follows similarly from Lemma (6.2.8). □

(6.2.10) Remark. In part (a) of Lemma (6.2.9), for every rational vector v and every positive integer φ, the number q and the vector w can be found in time polynomial in φ and $\langle v \rangle$ by the algorithm of Theorem (5.3.19). A similar remark applies to (b) of (6.2.9). □

The previous lemma gives a "rounding" procedure that assigns to each vector v a "rounded" vector $\frac{1}{q}w$, $w \in \mathbb{Z}^n$, with "small" denominator q such that, if v lies in, or near, any polyhedron with low facet complexity, then the vector $\frac{1}{q}w$ belongs to the polyhedron.

Sometimes it is more desirable to round so that points outside the polyhedron should stay outside. Such a procedure was given by FRANK and TARDOS (1987). It solves the following two problems.

(6.2.11) *Given a vector $y \in \mathbb{Q}^n$ and a natural number φ, find a vector $z \in \mathbb{Q}^n$ with the following properties:*

(i) *for each inequality $c^T x \leq \gamma$, with encoding length at most φ, we have $c^T y \leq \gamma$ if and only if $c^T z \leq \gamma$, and*
(ii) *the encoding length of z is at most $44n(n+1)^2\varphi$.*

(6.2.12) *Given a rational inequality $c^T x \leq \gamma$ in n variables, and a natural number ν, find a rational inequality $a^T x \leq \alpha$, in n variables, with the following properties:*

(i) *for each y in $\mathbb{Q}^n$ with encoding length at most ν, we have $c^T y \leq \gamma$ if and only if $a^T y \leq \alpha$;*
(ii) *the encoding length of $a^T x \leq \alpha$ is at most $22(n+1)^3(\nu+4)$.*

Frank and Tardos showed that these two problems are polynomially solvable.

(6.2.13) Theorem. *There exist polynomial time algorithms to solve problems* (6.2.11) *and* (6.2.12). *Moreover, the number of elementary arithmetic operations performed by these algorithms is polynomial in φ (ν, respectively).*

Proof. I. We first consider problem (6.2.11) for the case where we restrict (i) to inequalities $c^T x \leq \gamma$ with $\gamma = 0$. Let a vector $y \in \mathbb{Q}^n$ and $\varphi \in \mathbb{N}$ be given. Define

$$\kappa := 2^{7n\varphi}.$$

Determine vectors $z_1, z_2, \ldots$ successively as follows. First set $z_1 := y$. Suppose $z_1, \ldots, z_i$ have been found. If $z_i \neq 0$, let

$$v := \frac{1}{\kappa} \cdot \left\lfloor \frac{\kappa}{\|z_i\|_\infty} z_i \right\rfloor. \tag{6.2.14}$$

(Here $\lfloor\ \rfloor$ means the componentwise round-down. Note that v can be determined by $14n^2\varphi$ comparisons through binary search.)

By the algorithm of Theorem (5.3.19) we can find $u_i \in \mathbb{Z}^n$, $q_i \in \mathbb{Z}$ so that

$$\|v - \frac{1}{q_i} \cdot u_i\|_\infty < \frac{1}{q_i} \cdot 2^{-3\varphi} \tag{6.2.15}$$
$$1 \leq q_i < 2^{4n\varphi}.$$

(Note that u_i, q_i can be found in time polynomially bounded by $\langle v \rangle$ and φ, and that $\langle v \rangle$ is polynomially bounded by φ.)

Set

$$z_{i+1} := z_i - \frac{\|z_i\|_\infty}{q_i} \cdot u_i. \tag{6.2.16}$$

If $z_i = 0$, stop. Otherwise repeat with $i := i + 1$.

Since z_{i+1} has at least one more 0-component than z_i has, as one easily derives from (6.2.14), (6.2.15), and (6.2.16), the algorithm stops after $k \leq n$ iterations. Let $z_1, \ldots, z_k$ be the sequence generated. Note that by (6.2.16),

$$y = \frac{\|z_1\|_\infty}{q_1} u_1 + \frac{\|z_2\|_\infty}{q_2} u_2 + \ldots + \frac{\|z_k\|_\infty}{q_k} u_k. \tag{6.2.17}$$

We claim that

$$z := 2^{-8n\varphi} u_1 + 2^{-16n\varphi} u_2 + \ldots + 2^{-8kn\varphi} u_k \tag{6.2.18}$$

is a vector as required.

To show this, choose $c \in \mathbb{Q}^n$ with encoding length at most φ, so that $c^T y \leq 0$. If $c^T u_1 = \ldots = c^T u_k = 0$, then (6.2.18) implies $c^T z \leq 0$.

If $c^T u_i \neq 0$ for some i, choose the smallest such i. We show that $c^T u_i < 0$. Indeed, since by (6.2.16)

$$z_i = y - \frac{\|z_1\|_\infty}{q_1} u_1 - \frac{\|z_2\|_\infty}{q_2} u_2 - \ldots - \frac{\|z_{i-1}\|_\infty}{q_{i-1}} u_{i-1},$$

and since $c^T y \leq 0$, $c^T u_1 = \ldots = c^T u_{i-1} = 0$, we know $c^T z_i \leq 0$. Then

$$c^T u_i = c^T (u_i - q_i v) + q_i c^T (v - z_i) + q_i c^T z_i \leq$$
$$\leq \|c\| \cdot \|u_i - q_i v\| + q_i \cdot \|c\| \cdot \|v - z_i\| \leq 2^\varphi \cdot 2^{-3\varphi} + 2^{4n\varphi} \cdot 2^\varphi \cdot \frac{n}{\kappa} < 2^{-\varphi}.$$

Since u_i is integral and $\langle c \rangle \leq \varphi$, it follows that $c^T u_i \leq 0$. As $c^T u_i \neq 0$ we know $c^T u_i < 0$. Hence even

$$c^T u_i \leq -2^{-\varphi}.$$

Therefore, by (6.2.18),

$$c^T z = 2^{-8in\varphi} c^T u_i + 2^{-8(i+1)n\varphi} c^T u_{i+1} + \ldots + 2^{-8kn\varphi} c^T u_k$$
$$\leq -2^{-8in\varphi} 2^{-\varphi} + n \cdot 2^{-8(i+1)n\varphi} 2^{5n\varphi} < 0$$

(using $|c^T u_j| \leq \|c\|_1 \cdot \|u_j\|_\infty \leq 2^\varphi \cdot q_j \leq 2^\varphi \cdot 2^{4n\varphi} \leq 2^{5n\varphi}$).

So, if $c^T y \leq 0$ then $c^T z \leq 0$; and if $c^T y < 0$ then $c^T z < 0$. Similarly, if $c^T y > 0$ then $c^T z > 0$. Hence $c^T y \leq 0$ iff $c^T z \leq 0$.

One easily checks that each component of z has encoding length at most $22n^2\varphi$.

II. We next consider problem (6.2.11) for general γ. Let $y \in \mathbb{Q}^n$ and $\varphi \in \mathbb{N}$ be given. With I. applied to $(y^T, 1)^T \in \mathbb{Q}^{n+1}$, we can find a vector $(\tilde{z}^T, \lambda)^T \in \mathbb{Q}^{n+1}$ so that for each linear inequality $c^T x \leq \gamma$ of encoding length at most φ we have: $c^T y - \gamma \leq 0$ iff $c^T \tilde{z} - \lambda\gamma \leq 0$, and so that each component of $(\tilde{z}^T, \lambda)^T$ has encoding length at most $22(n+1)^2\varphi$. Since $0^T y - 1 < 0$, this implies $0^T \tilde{z} - \lambda < 0$, and thus $\lambda > 0$. Then $z := \lambda^{-1}\tilde{z}$ has the following property: for each inequality $c^T x \leq \gamma$ of encoding length at most φ, $c^T y \leq \gamma$ holds if and only if $c^T z \leq \gamma$ holds.

Moreover, for each component of z the encoding length is at most $44(n+1)^2\varphi$. Hence z has encoding length at most $44n(n+1)^2\varphi$.

III. Finally we consider problem (6.2.12). Let be given an inequality $c^T x \leq \gamma$ in n variables and $\nu \in \mathbb{N}$. We apply I. to the vector $(c^T, \gamma)^T \in \mathbb{Q}^{n+1}$ and $\varphi = \nu + 4$. We obtain $(a^T, \alpha)^T \in \mathbb{Q}^{n+1}$ such that

(i) for each vector $(z^T, \lambda)^T \in \mathbb{Q}^{n+1}$ of encoding length at most $\nu + 4$ we have $c^T z \leq \lambda\gamma$ if and only if $a^T z \leq \lambda\alpha$;
(ii) each component of $(a^T, \alpha)^T$ has encoding length at most $22(n+1)^2(\nu+4)$.

Property (i) gives that for each vector $z \in \mathbb{Q}^n$ with encoding length at most ν, we have $c^T z \leq \gamma$ if and only if $a^T z \leq \alpha$. Property (ii) gives that the inequality $a^T x \leq \alpha$ has encoding length at most $22(n+1)^3(\nu+4)$. □

One of the main applications of the technique described above is the following consequence of the algorithm for problem (6.2.12).

(6.2.19) Corollary. *There exists a polynomial time algorithm that, given a vector $c \in \mathbb{Q}^n$ and a natural number ν, computes a vector $a \in \mathbb{Q}^n$ with the following properties:*

(i) *for each polyhedron $P \subseteq \mathbb{R}^n$ with vertex-complexity at most ν, the objective functions $c^T x$ and $a^T x$ are optimized by the same vectors in P, and*
(ii) $\langle a \rangle \leq 88(n+1)^3(\nu+1)$.

Moreover, the number of elementary arithmetic operations the algorithm performs is bounded by a polynomial in ν. □

*6.3 Weak and Strong Problems

We shall now show that for bounded, full-dimensional, well-described polyhedra the weak and strong versions of the five basic problems introduced in Section 2.1 are polynomially equivalent – with the exception of the membership problem, where some additional information is needed. Remark (6.2.10) gives the following direct corollary.

(6.3.1) Corollary. *There exists an oracle-polynomial time algorithm that, for any well-described, full-dimensional polyhedron $(P; n, \varphi)$ given by a weak nonemptiness oracle, solves the strong nonemptiness problem, i. e., for every such polyhedron P a point in P can be found in oracle-polynomial time.* □

The next theorem is a major result. It shows that, for well-described, full-dimensional polytopes, oracle-polynomial time algorithms for the strong problems (2.1.1),..., (2.1.5) can be derived from oracles for the weak problems (2.1.10),..., (2.1.14).

(6.3.2) Theorem.

(a) *There exists an oracle-polynomial time algorithm that, for any bounded, full-dimensional, well-described polyhedron $(P; n, \varphi)$ given by a weak optimization (violation, validity, separation) oracle, solves the strong optimization (violation, validity, separation) problem.*

(b) *There exists an oracle-polynomial time algorithm that, given any bounded, full-dimensional, well-described polyhedron $(P; n, \varphi)$ specified by a weak membership oracle, and given any interior point $a_0 \in P$, solves the strong membership problem.*

Proof. (a) 1. First we prove that weak optimization yields strong optimization.

Let $c \in \mathbb{Q}^n$ with $c \neq 0$ be given. Call the weak optimization oracle for P with input c and ε, where

$$\varepsilon := \frac{2^{-18n\langle c\rangle - 24n^4\varphi}}{\|c\|_\infty}.$$

By the choice of ε, $S(P, -\varepsilon) \neq \emptyset$, and so the oracle gives an "almost optimal" point v. Using the simultaneous diophantine approximation algorithm described in Chapter 5, find an integer q and an integer vector w such that

$$\|qv - w\| < 2^{-9\langle c\rangle - 24n^2\varphi}$$

$$0 < q < 2^{9n\langle c\rangle + 24n^3\varphi + n^2}.$$

Then v, q, w and P satisfy the hypothesis of Lemma (6.2.9) (a), and hence $\frac{1}{q}w$ is in P. Moreover, let $M := \max\{c^T x \mid x \in P\}$. Then v, q, w and the hyperplane $c^T x = M$ also satisfy the hypothesis of this lemma and hence $c^T(\frac{1}{q}w) = M$, i. e., $\frac{1}{q}w$ is a solution of the strong optimization problem.

2. Second, we show that weak validity yields strong validity. Let $c \in \mathbb{Q}^n$ and $\gamma \in \mathbb{Q}$ be given. Call the weak validity oracle with c and

$$\gamma' := \gamma + \varepsilon(2 + \|c\|_1),$$
$$\varepsilon := 2^{-\langle c\rangle - \langle \gamma\rangle - 4n^2\varphi}(3 + \|c\|_1 2^{4n^3\varphi})^{-1}.$$

If it concludes that $c^T x \geq \gamma' - \varepsilon$ for some $x \in S(P, \varepsilon)$ then there exists an $x' \in P$ such that $\|x - x'\| \leq \varepsilon$. So $c^T x' \geq c^T x - \|c\|_1 \varepsilon \geq \gamma' - \varepsilon - \|c\|_1 \varepsilon > \gamma$. Hence $c^T x \leq \gamma$ is invalid for P.

If the oracle concludes that $c^T x \leq \gamma' + \varepsilon$ for all $x \in S(P, -\varepsilon)$, then consider any vertex v of P. By Lemma (6.2.6), P contains a ball $S(a_0, 2^{-7n^3\varphi})$, and by Lemma (6.2.5), $\|a_0 - v\| \leq 2^{5n^2\varphi+1}$. Let

$$x := v + \varepsilon 2^{7n^3\varphi}(a_0 - v).$$

Then $S(x, \varepsilon)$ is contained in P, and hence x is in $S(P, -\varepsilon)$. Therefore

$$\begin{aligned} c^T v - \gamma &= c^T(v - x) + (c^T x - \gamma') + (\gamma' - \gamma) \\ &\leq \|c\| \, \|v - x\| + \varepsilon + \varepsilon(2 + \|c\|_1) \\ &= \|c\| \varepsilon 2^{7n^3\varphi} \|a_0 - v\| + \varepsilon(3 + \|c\|_1) \\ &\leq \varepsilon(2^{7n^3\varphi+5n^2\varphi+1} \|c\| + 3 + \|c\|_1) \\ &< 2^{-\langle c\rangle-\langle\gamma\rangle-\langle v\rangle}. \end{aligned}$$

Since the left hand side is a rational number with denominator at most $2^{\langle c\rangle+\langle\gamma\rangle+4n^2\varphi}$, it follows that $c^T v \leq \gamma$. As the vertex v is arbitrarily chosen, the inequality $c^T x \leq \gamma$ is valid for P.

3. It follows by similar arguments that weak violation yields strong violation.

4. To show that weak separation yields strong separation turns out to be more difficult in the sense that we have to use a more complex tool. Let $y \in \mathbb{Q}^n$ be given. First we use Corollary (4.2.7) and Theorem (4.4.9) to find a point a_0 in the interior of P. Without loss of generality assume that $a_0 = 0$. Let P^* be the polar polyhedron of P. The weak separation oracle for P yields a weak violation oracle for P^*. By part 2 of this proof this yields a strong violation algorithm for P^* which in turn yields a strong separation algorithm for P.

(b) The proof could be done analogous to part 4 above. A more direct proof is the following.

Let $y \in \mathbb{Q}^n$ be given. We may assume $y \neq a_0$. Define

$$\varepsilon := 2^{-\langle y\rangle-2\varphi}.$$

Choose any rational point $y' \neq y$ on the line segment connecting a_0 and y such that $\|y - y'\| \leq \varepsilon/2$. Set

$$\varepsilon' := \min\{\frac{\varepsilon}{2}, 2^{-\langle y'\rangle-2\varphi}\}.$$

Call the weak membership oracle for P with input y' and ε'. If the oracle concludes that $y' \in S(P, \varepsilon')$ then $y \in S(P, \varepsilon)$ and so by Lemma (6.2.7) $y \in P$. Suppose the oracle concludes that $y' \notin S(P, -\varepsilon')$. Then there exists an inequality $a^T x \leq b$ valid for P with encoding length at most φ such that y' has distance at most ε' from the halfspace $H = \{x \mid a^T x \geq b\}$. Since H has facet-complexity at most φ, by Lemma (6.2.7), $y' \in H$. Hence y' is not in the interior of P which implies that $y \notin P$. □

(6.3.3) Remark. The hypothesis that P be full-dimensional is essential. It is easy to verify that if P is not full-dimensional then for any of the weak problems (2.1.10),...,(2.1.14) we can design an oracle that does not distinguish P from the empty set, since $S(P, -\varepsilon)$ is empty. □

(6.3.4) Remark. The hypothesis that P be bounded is, on the other hand, not always essential if we extend the definition of the weak versions of our problems appropriately. We illustrate this by the example of the optimization problem.

The **weak optimization problem for a (not necessarily bounded) polyhedron** P is as follows. Given a vector $c \in \mathbb{Q}^n$ and a rational number $\varepsilon > 0$, either

(6.3.5)
- (i) assert that $S(P, -\varepsilon)$ is empty, or
- (ii) find a vector $y \in S(P, \varepsilon)$ such that $c^T x \leq c^T y + \varepsilon$ for all $x \in S(P, -\varepsilon) \cap S(0, 1/\varepsilon)$, or
- (iii) find a vector z such that $\frac{1}{\|z\|_\infty} z \in S(\text{rec}(P), \varepsilon)$ and $c^T z \geq 1$.

To motivate the occurence of the ball $S(0, 1/\varepsilon)$ in (ii) of (6.3.5) let us point out that the condition $x \in S(P, -\varepsilon) \cap S(0, 1/\varepsilon)$ can be interpreted as "x is deep in P" in the sense that x is also separated from infinity. It may be instructive to verify that the use of the ball $S(0, 1/\varepsilon)$ is necessary to maintain the "continuity" of the output as sketched in Chapter 2.

One can now show similarly as in Theorem (6.3.2) that for full-dimensional, well-described polyhedra weak and strong optimization are polynomially equivalent. One can modify the definitions of weak violation, weak validity, and weak separation in a similar spirit and obtain analogous results.

Dropping the boundedness assumption for the membership problem with given interior point no modification is needed in the definition, nor in the statement and the proof of part (b) of Theorem (6.3.2). □

(6.3.6) Remark. The hypothesis in part (b) of (6.3.2) that an interior point of P is given cannot be dropped. Suppose we have a weak membership oracle for a polyhedron $P \subseteq \mathbb{R}^n$ (which we do not know) and we know that P has vertex-complexity $2n$. We wish to decide whether $0 \in P$. Suppose the oracle gives a negative answer whatever we ask. If we ask less than 2^n questions, then there exists a unit cube H with $\{0, +1, -1\}$-vertices containing zero as a vertex whose interior has never been hit by our questions and therefore all the answers given by the oracle have been legitimate for $P = H$. But they were also legitimate for $P = \text{conv}\{x \in H \mid x \text{ a vertex of } H, x \neq 0\}$. Thus it is impossible to decide whether $0 \in P$ in less than 2^n calls on the oracle.

A similar construction can be given to show that even in dimension two it may be necessary to call the oracle a number of times exponential in the given vertex-complexity of the polyhedron. □

(6.3.7) Remark. In part 4 of the proof of Theorem (6.3.2) showing the equivalence of weak and strong separation the ellipsoid method was used. One may wonder whether a proof with a more elementary rounding trick could be given (as in the other parts of the proof). We will now show that it is just as difficult to derive strong separation from weak separation as it is to derive strong nonemptiness from weak separation. (Note that the latter includes solving a set of linear inequalities and so also linear programming.)

More precisely, we shall show by an elementary construction that if we have an oracle-polynomial algorithm to solve the strong separation problem for full-dimensional well-described polytopes given by some weak separation oracle, then this can be used to solve the strong nonemptiness problem for such convex bodies.

Let a full-dimensional well-described polytope $(P; n, \varphi)$ be given by a weak separation oracle. By hypothesis we may as well assume that we have a strong separation oracle for P. Consider the polytope $P' \subseteq \mathbb{R}^{n+1}$ defined by

$$P' = \{(\lambda x^T, \lambda)^T \in \mathbb{R}^{n+1} \mid 0 \leq \lambda \leq 1, x \in P\}.$$

Then the weak separation problem for P' can be solved in oracle-polynomial time as follows.

Let a point $(y^T, \mu)^T \in \mathbb{Q}^{n+1}$ and a rational number $\delta > 0$ be given. If $\mu \leq 0$ then $x_{n+1} \geq 0$ is valid for P' and weakly separates y from P'. If $\mu \geq 1$ then the same holds for the inequality $x_{n+1} \leq 1$. If $0 < \mu < 1$ then we call the strong separation oracle for P with input $\frac{1}{\mu}y$. If the oracle concludes that $\frac{1}{\mu}y \in P$ then we conclude that $(y^T, \mu)^T \in S(P', \delta)$. If the oracle gives a vector $c \in \mathbb{Q}^n$ such that $\|c\|_\infty = 1$ and $c^T x < c^T \frac{1}{\mu} y$ for all $x \in P$ then the vector $(c^T, -\frac{1}{\mu}c^T y)/\max\{1, |\frac{1}{\mu}c^T y|\} =: c'$ is a valid output for the weak separation problem for P'.

Apply now the hypothesized strong separation algorithm to test whether the origin is in P'. It of course concludes with "yes" since $P \neq \emptyset$. Consider those points for which the weak separation oracle for P' has been called. We claim that for at least one of these points the oracle must have concluded that it was "near to P'". For, otherwise all answers of the oracle would have been legitimate also for the polyhedron

$$P'' := \{(\lambda x^T, \lambda)^T \in \mathbb{R}^{n+1} \mid \frac{1}{2} \leq \lambda \leq 1, x \in P\}$$

which however does not contain zero. But, if the answer is that $(y^T, \mu)^T$ is "near to P'" then $\frac{1}{\mu}y \in P$ and so the strong nonemptiness problem for P can be solved. □

*6.4 Equivalence of Strong Optimization and Separation

In this section we shall prove one of the fundamental results of this book, namely that for well-described polyhedra (boundedness and full-dimensionality are not assumed) the strong optimization problem and the strong separation problem are equivalent with respect to polynomial time solvability. As a first step, we show that, for well-described polyhedra, strong separation yields strong nonemptiness.

(6.4.1) Theorem. *The strong nonemptiness problem for well-described polyhedra given by a strong separation oracle, can be solved in oracle-polynomial time.*

Proof. I. We first show the existence of a subroutine.

(6.4.2) *There exists an oracle-polynomial time algorithm* ELL″ *that, for any well-described bounded polyhedron* $(P; n, \varphi)$ *given by a strong separation oracle, either*

(i) *gives a nonzero vector* $p \in \mathbb{Z}^n$ *and a natural number* d *such that* $P \subseteq \{x \in \mathbb{Q}^n \mid p^T x = d\}$, *or*

(ii) *gives a vector* x *in* P.

Indeed, let $(P; n, \varphi)$ be a well-described bounded polyhedron, given by the strong separation oracle SEP. Let $v := 4n^2\varphi$, which is an upper bound on the vertex-complexity of P. Apply the central-cut ellipsoid method (3.2.1) to P, with

$$R := 2^v,$$
$$\varepsilon := 2^{-12n^3v},$$

and the given strong separation oracle SEP. If the central-cut ellipsoid method concludes with a vector y such that $y \in S(P, \varepsilon)$, then we next obtain a vector x in P by applying the simultaneous diophantine approximation algorithm to y. With this algorithm we find an integer vector $p \in \mathbb{Z}^n$ and an integer q such that

$$\|y - \tfrac{1}{q} \cdot p\| \leq \tfrac{1}{q} 2^{-7n^2v}, \text{ and } 1 \leq q \leq 2^{n^2+7n^3v}.$$

The vector $x := \frac{1}{q} \cdot p$ belongs to P by Lemma (6.2.7). (In fact, if the central-cut ellipsoid method concludes that $y \in S(P, \varepsilon)$, then y itself is in P, since the separation oracle is strong.)

The other possible outcome is an ellipsoid $E = E(A, a)$, with volume at most ε, and containing P. Hence

$$\det A = \left(\frac{\operatorname{vol} E}{V_n}\right)^2 \leq n^{2n}\varepsilon^2 \leq 2^{-22n^2v}, \tag{6.4.3}$$

where V_n denotes the volume of the n-dimensional unit ball. (Recall that $V_n \geq n^{-n}$.)

Now we can find a vector u with $u^T A u \leq 2^{-20nv}$, $\|u\|_\infty = 1$ as follows. (Comment: we could find such a vector by computing an eigenvector belonging to the smallest eigenvalue of A. However, to avoid discussions on polynomiality and rounding, we describe a different method.) By (6.4.3),

$$\det(A^{-1}) \geq 2^{22n^2v}$$

and hence the largest eigenvalue of A^{-1} is at least 2^{22nv}. Therefore, $\operatorname{tr}(A^{-1}) \geq 2^{22nv}$ (as A^{-1} is positive definite). We may assume without loss of generality that the maximum element of the main diagonal of A^{-1} is in position $(1, 1)$. Then

$$\lambda := (A^{-1})_{11} \geq n^{-1} \cdot \operatorname{tr}(A^{-1}) \geq n^{-1} \cdot 2^{22nv} \geq 2^{20nv}.$$

Let v be the first column of A^{-1}, and let $u := \frac{1}{\lambda} \cdot v$. Then

$$u^T A u = \frac{1}{\lambda^2} \cdot v^T A v = \frac{1}{\lambda^2} \cdot (A^{-1})_{11} = \lambda^{-1} \leq 2^{-20nv}. \tag{6.4.4}$$

Moreover, $u_1 = (A^{-1})_{11}/\lambda = 1$, and hence $\|u\|_\infty = 1$ (as A^{-1} is positive definite). Thus we have found the desired u, with, without loss of generality, $u_1 = 1$.

Let $\beta := u^T a$. Then for each x in P (using the Cauchy-Schwarz inequality (6.4.4), and the fact that $P \subseteq E$):

$$(6.4.5) \qquad |u^T x - \beta| = |u^T(x-a)| \leq (u^T A u)^{1/2}((x-a)^T A^{-1}(x-a))^{1/2} \leq 2^{-10nv}.$$

With the simultaneous diophantine approximation algorithm (5.3.19) applied to $u_1, \ldots, u_n, \beta$ we can find an integral vector $p = (p_1, \ldots, p_n)^T$ and integers d and q such that

$$\begin{aligned} \|p - qu\| &\leq 2^{-4v}, \\ |d - q\beta| &\leq 2^{-4v}, \\ 1 \leq q &\leq 2^{(n+1)(n+2)/4} \cdot 2^{4(n+1)v} \leq 2^{7nv}. \end{aligned}$$

Then $p \neq 0$, as $p_1 \geq 1$, since

$$p_1 \geq qu_1 - |p_1 - qu_1| \geq q - 2^{-4v} > 0,$$

as $u_1 = 1$. For each vertex x in P (cf. (6.4.5)), we have

$$\begin{aligned} |p^T x - d| &\leq |(p-qu)^T x| + |q(u^T x - \beta)| + |q\beta - d| \\ &\leq \|p - qu\| \cdot \|x\| + |q(u^T x - \beta)| + |q\beta - d| \\ &\leq 2^{-4v} \cdot 2^v + 2^{7nv} \cdot 2^{-10nv} + 2^{-4v} \\ &< 2^{-v}. \end{aligned}$$

However, as p and d are integer, and x has encoding length at most v, $p^T x - d$ is a rational number with denominator at most 2^v. Hence $p^T x = d$ for all vertices x of P, and hence for all vectors x in P.

It is easily checked that the above procedure is an oracle-polynomial time algorithm.

II. The algorithm ELL″ is a subroutine for the following algorithm, which is almost as required by the theorem, except for the restriction to bounded polyhedra.

(6.4.6) *There exists an oracle-polynomial time algorithm* ELL′ *that solves the strong nonemptiness problem for well-described polyhedra given by a strong separation oracle.*

Indeed, let $(P; n, \varphi)$ be a well-described polyhedron, given by a strong separation oracle SEP. Let $v := 4n^2\varphi$. We apply the following iteration step repeatedly, starting with $k = 0$. Suppose we have found linearly independent integer vectors $c_1, \ldots, c_k$ and integers $d_1, \ldots, d_k$ such that

$$(6.4.7) \qquad P \subseteq \{x \mid c_1^T x = d_1, \ldots, c_k^T x = d_k\}.$$

Without loss of generality, assume that we can write

$$\begin{pmatrix} c_1^T \\ \vdots \\ c_k^T \end{pmatrix} = (C_1, C_2),$$

where C_1 is a nonsingular matrix of order k. Let

$$d := \begin{pmatrix} d_1 \\ \vdots \\ d_k \end{pmatrix}.$$

If $k = n$, then $C_1^{-1}d$ is the only possible vector in P. By giving this vector to the separation oracle, we see whether P is empty. So let $k < n$.

Let the polyhedron P_k be the projection of P arising by forgetting the first k coordinates, i. e.,

$$P_k = \{y \in \mathbb{Q}^{n-k} \mid \begin{pmatrix} \tilde{y} \\ y \end{pmatrix} \in P \text{ for some } \tilde{y} \in \mathbb{Q}^k\}.$$

By (6.4.7), $y \in P_k$ if and only if $((C_1^{-1}d - C_1^{-1}C_2y)^T, y^T)^T \in P$. Moreover, all vertices of P_k have encoding length at most v.

We describe a strong separation oracle SEP_k for P_k. If $w \in \mathbb{Q}^{n-k}$ is given, apply SEP to the vector

$$z := ((C_1^{-1}d - C_1^{-1}C_2w)^T, w^T)^T.$$

If SEP answers that z is in P, then w belongs to P_k. If SEP gives a vector a such that $a^Tx < a^Tz$ for all x in P, decompose $a^T = (a_1^T, a_2^T)$, with a_1 and a_2 vectors of length k and $n-k$, respectively. Then

$$(a_2^T - a_1^TC_1^{-1}C_2)y < (a_2^T - a_1^TC_1^{-1}C_2)w$$

for all y in P_k, and SEP_k gives the vector $a_2 - (a_1^TC_1^{-1}C_2)^T$ as answer.

Now apply the algorithm ELL″ of Part I to the well-described polytope $(P_k; n-k, 3n^2v)$, with the strong separation oracle SEP_k. Then ELL″ will produce one of the following two answers (i), (ii).

(i) ELL″ gives a nonzero integral vector $\tilde{c}_{k+1} \in \mathbb{Z}^{n-k}$ and an integer d_{k+1} such that $P_k \subseteq \{y \in \mathbb{Q}^{n-k} \mid \tilde{c}_{k+1}^T y = d_{k+1}\}$.
(Defining $c_{k+1}^T := (0, \ldots, 0, \tilde{c}_{k+1}^T) \in \mathbb{Z}^n$, we have $P \subseteq \{x \in \mathbb{Q}^n \mid c_{k+1}^T x = d_{k+1}\}$, and therefore we can start the iteration anew, with k replaced by $k+1$.)

(ii) ELL″ gives a vector y in P_k.
(Then the algorithm ELL′ will give the vector $((C_1^{-1}d - C_1^{-1}C_2y)^T, y^T)^T$ as an element of P.)

This describes the algorithm ELL′.

Note that throughout the iterations the vertex-complexities of the polyhedra P_k remain uniformly bounded by v, and thus, the encoding lengths of the c_i and d_i are uniformly bounded by a polynomial in v. This implies that the running time of the whole algorithm is oracle-polynomially bounded.

III. Finally, the unbounded case is easily reduced to the bounded case. If a well-described polyhedron $(P; n, \varphi)$ is given let

$$P' := P \cap \{x \in \mathbb{Q}^n \mid -2^{4n^2\varphi} \leq x_i \leq 2^{4n^2\varphi} \text{ for } i = 1, \ldots, n\}.$$

Then $P \neq \emptyset$ if and only if $P' \neq \emptyset$, by Lemma (6.2.4). Moreover, a separation oracle for P' is easily derived from a separation oracle for P. So applying ELL$'$ to the well-described bounded polyhedron $(P'; n, 5n^2\varphi)$ gives the required result. □

(6.4.8) Lemma. *The strong separation problem for* rec(P), *where* $(P; n, \varphi)$ *is a well-described polyhedron given by a strong separation oracle, can be solved in oracle-polynomial time.*

Proof. By Theorem (6.4.1) we can solve the strong nonemptiness problem for P in oracle-polynomial time. If P is empty, then $\text{rec}(P) = \{0\}$ and the strong separation problem for $\text{rec}(P)$ is trivial.

Suppose that we find a point $x_0 \in P$. Then let $y \in \mathbb{Q}^n$ be a given point. Define

$$N := 2^{+2\varphi+\langle y\rangle+\langle x_0\rangle}$$

and call the strong separation oracle for P with input $x_0 + Ny$.

First, if the separation oracle for P gives a vector $c \in \mathbb{Q}^n$ such that $c^T x < c^T(x_0 + Ny)$ for all $x \in P$, then for all $z \in \text{rec}(P)$, $c^T(x_0 + Nz) < c^T(x_0 + Ny)$ and so $c^T z < c^T y$. Thus c is a valid strong separation for $\text{rec}(P)$ and y.

Second, suppose that the separation oracle for P confirms that $x_0 + Ny$ is in P. We show that y is in $\text{rec}(P)$. To this end, let $a^T x \leq \alpha$ be a valid inequality for P of encoding length at most φ. Then in particular we have

$$a^T(x_0 + Ny) \leq \alpha$$

or

$$a^T y \leq \frac{1}{N}(\alpha - a^T x_0).$$

Here $a^T y$ is a rational number with denominator at most $2^{\varphi+\langle y\rangle}$ while the right hand side is at most

$$\frac{1}{N}(|\alpha| + \|a\|\,\|x_0\|) \leq \frac{1}{N}(2^{\varphi-1} + 2^{\varphi-1}2^{\langle x_0\rangle}) < 2^{-\varphi-\langle y\rangle}.$$

Hence $a^T y \leq 0$ which shows that $y \in \text{rec}(P)$. □

We are now prepared to prove one of the main theorems of this book.

(6.4.9) Theorem. *Any one of the following three problems:*

- *strong separation,*
- *strong violation,*
- *strong optimization,*

can be solved in oracle-polynomial time for any well-described polyhedron given by an oracle for any of the other two problems.

Proof. I. Suppose first that $(P; n, \varphi)$ is given by a strong separation oracle. To show that the strong optimization problem is solvable in oracle-polynomial time, let $c \in \mathbb{Q}^n$ be a vector.

Run the algorithm of Theorem (6.4.1). If it concludes that P is empty, then we are done. Suppose it finds a point $x_0 \in P$.

Next we run the strong nonemptiness algorithm of Theorem (6.4.1) for $\text{rec}(P) \cap \{x \mid c^T x = 1\}$ using the strong separation algorithm for $\text{rec}(P)$ given by Lemma (6.4.8). If it gives a vector $z \in \text{rec}(P) \cap \{x \mid c^T x = 1\}$ then the optimization algorithm asserts that $z \in \text{rec}(P)$ and $c^T z \geq 1$. Otherwise we know that $c^T x$ is bounded on P from above – in fact $c^T x \leq N$, where

$$N := 2^{\langle c \rangle + 4n^2 \varphi}.$$

Define

$$\varepsilon := \frac{1}{4N^2}.$$

For every rational γ we can decide in oracle-polynomial time whether the polyhedron

$$P_\gamma := P \cap \{x \mid \gamma \leq c^T x \leq N\}$$

is empty by Theorem (6.4.1). So by binary search we can find a rational number $\gamma \in [c^T x_0, N]$ such that P_γ is nonempty but $P_{\gamma+\varepsilon}$ is empty. By the method of continued fractions find a rational number $r = p/q$ such that

$$|\gamma - \frac{p}{q}| < \frac{1}{2Nq}, \quad 0 < q \leq 2N.$$

We claim that

$$\max\{c^T x \mid x \in P\} = r.$$

In fact, the left hand side is a rational number s/t with $0 < t < N$ and with

$$|\gamma - \frac{s}{t}| < \varepsilon \leq \frac{1}{2Nq}.$$

Hence

$$|\frac{p}{q} - \frac{s}{t}| < \frac{1}{Nq} \leq \frac{1}{qt}.$$

Thus $p/q = s/t$.

Now the strong nonemptiness algorithm applied to P_r gives a vector $y \in P$ maximizing $c^T x$ on P.

II. Obviously, a strong optimization oracle for P yields an oracle-polynomial strong violation algorithm for P.

III. Suppose we have a strong violation oracle for $(P; n, \varphi)$. We are going to design a strong separation algorithm for P. Consider (the so-called γ-**polar** of P)

$$Q := \{(z^T, \lambda)^T \in \mathbb{R}^{n+1} \mid z^T x \leq \lambda \text{ for all } x \in P\}.$$

Then $(Q; n+1, 4n^2\varphi + 3)$ is a well-described rational polyhedron. Moreover, the strong violation oracle for P yields a strong separation algorithm for Q. By parts I and II this yields a strong violation algorithm for Q which in turn yields a strong separation algorithm for P.

We leave it to the reader to check that all the above algorithms are oracle-polynomial. □

(6.4.10) Remark. (a) It is clear that if the polyhedron is not full-dimensional then a strong membership oracle is not sufficient to solve any of the three equivalent problems mentioned in Theorem (6.4.9) in oracle-polynomial time. Similarly, for unbounded polyhedra a strong validity oracle does not yield oracle-polynomial algorithms for these other problems.
(b) In the special case where P is a cone, the strong violation problem for P is obviously the same as a strong separation problem for the polar cone P^*. So a strong separation algorithm for P yields a strong separation algorithm for P^*. □

(6.4.11) Remark. In case we know in advance that the well-described polyhedron $(P; n, \varphi)$ is nonempty, the algorithm described in the proof of Theorem (6.4.1) works even if we replace the strong separation oracle for P by the following weaker separation oracle: Given $y \in \mathbb{Q}^n$, the oracle may conclude that $y \in P$, or it may provide a vector $c \in \mathbb{Q}^n$ such that $c^T y \geq c^T x$ for all $x \in P$, and such that in case $y \notin P$, $c^T y > c^T x$ holds for all $x \in P$. So the oracle may provide us with a "separating hyperplane" even if $y \in P$. This implies that for nonempty well-described polyhedra we can relax the strong separation oracle in Theorem (6.4.9) similarly. This technical variation will play an important role in Chapter 10. □

At this point we want to remark that Theorem (6.4.9) yields Khachiyan's result on the polynomial time solvability of linear programming problems. We should not go on without mentioning that this result, in fact, was the starting point of most of the research presented in this book.

(6.4.12) Theorem. *For any $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, and $c \in \mathbb{Q}^n$, the linear program*

$$\max c^T x$$
$$Ax \leq b$$

can be solved in polynomial time.

Proof. The strong separation problem for $P := \{x \mid Ax \leq b\}$ can be solved trivially in polynomial time by checking whether a given vector $y \in \mathbb{Q}^n$ satisfies $Ax \leq b$ or not. So the strong optimization for P can be solved in polynomial time by Theorem (6.4.9). □

This result was extended to convex quadratic programming by KHACHIYAN, KOZLOV and TARAZOV (1980).

*6.5 Further Problems for Polyhedra

There are many other algorithmic problems concerning polyhedra which one often encounters. In this section we show that many of these problems (finding the dimension, the affine hull, a vertex, dual solutions to linear programs etc.) can be solved in oracle-polynomial time given an oracle for one of the three equivalent basic problems discussed in Theorem (6.4.9). First we settle some of these problems for polytopes.

(6.5.1) Lemma. *There exists an oracle-polynomial time algorithm that, for any nonempty, well-described polytope $(P; n, \varphi)$ given by a strong optimization oracle, finds a vertex of P.*

Proof. Find successively

$$\begin{aligned} c_1 &:= \max\{x_1 \mid x \in P\}, \\ c_2 &:= \max\{x_2 \mid x \in P, x_1 = c_1\}, \\ &\vdots \\ c_n &:= \max\{x_n \mid x \in P, x_1 = c_1, \ldots, x_{n-1} = c_{n-1}\}. \end{aligned}$$

Then $(c_1, \ldots, c_n)^T$ is a vertex of P. □

(6.5.2) Remark. If we want to find an optimum vertex solution of the optimization problem $\max\{c^T x \mid x \in P\}$, where P is a polytope, then we can find first the optimum value γ and then find a vertex of the polytope $P \cap \{x \mid c^T x = \gamma\}$. A more direct way is to "perturb" the objective function, i. e., replace c by $\bar{c} := c + (\varepsilon, \varepsilon^2, \ldots, \varepsilon^n)^T$ where ε is small enough, such that the optimum of $\bar{c}^T x$ over $x \in P$ is attained at a single vertex of P that also maximizes $c^T x$ over P.

Another application of the perturbation idea is the following one solving a multiobjective linear programming problem. Suppose we have a strong optimization oracle for P and let $c, \bar{c} \in \mathbb{Q}^n$. Let F be the face of points of P maximizing $c^T x$ over P. Then we can maximize the objective function $\bar{c}^T x$ over F by maximizing the objective function $(c + \varepsilon\bar{c})^T x$ over P for an appropriately small $\varepsilon > 0$. □

The following lemma is due to EDMONDS, LOVÁSZ and PULLEYBLANK (1982):

(6.5.3) Lemma. *There exists an oracle-polynomial time algorithm that, for any nonempty, well-described polytope $(P; n, \varphi)$ given by a strong optimization oracle, determines the affine hull of P. Specifically, the algorithm finds affinely independent vertices $v_0, v_1, \ldots, v_k$ of P and linearly independent equations $c_1^T x = \gamma_1, \ldots, c_{n-k}^T x = \gamma_{n-k}$ valid for P.*

Note that such vertices and equations necessarily satisfy

$$\text{aff}(P) = \text{aff}(\{v_0, v_1, \ldots, v_k\}) = \{x \in \mathbb{R}^n \mid c_i^T x = \gamma_i, i = 1, \ldots, n-k\}.$$

Proof. First find any vertex v_0 of P with the algorithm of Lemma (6.5.1).

Suppose we have found affinely independent vertices $v_0, \ldots, v_i$ $(i \geq 0)$ of P and linearly independent equations $c_1^T x = \gamma_1, \ldots, c_j^T x = \gamma_j$ $(j \geq 0)$ valid for P. In the beginning we have $i = 0$, $j = 0$. If $i + j < n$ then we find a basis of the subspace orthogonal to $\text{aff}(\{v_0, \ldots, v_i\})$ and pick any element c of it linearly independent from $c_1, \ldots, c_j$. Determine a vertex v' maximizing $c^T x$ over P and a vertex v'' minimizing $c^T x$ over P. If $c^T v' = c^T v''$ then we set $c_{j+1} := c$ and $\gamma_{j+1} := c^T v'$. If $c^T v' > c^T v''$ then at least one of v' and v'' is not in $\text{aff}(\{v_0, \ldots, v_i\})$ and we can choose this vertex as v_{i+1}.

As soon as $i + j = n$ we are finished. □

(6.5.4) Remark. (a) Note that the encoding length of the equations $c_i^T x = \gamma_i$ found in (6.5.3) is bounded by a polynomial in φ and n.

(b) Lemma (6.5.3) also immediately yields the dimension of P. Moreover,

$$v := \frac{1}{k+1}(v_0 + \ldots + v_k)$$

is a point in the relative interior of P. □

It is now easy to extend the results above to the case of general polyhedra.

(6.5.5) Theorem. *There exists an oracle-polynomial time algorithm that, for any nonempty, well-described polyhedron $(P; n, \varphi)$ given by a strong optimization oracle, determines a system of linearly independent equations $c_1^T x = \gamma_1, \ldots, c_m^T x = \gamma_m$ such that*

$$\text{aff}(P) = \{x \in \mathbb{R}^n \mid c_1^T x = \gamma_1, \ldots, c_m^T x = \gamma_m\}$$

and a system of affinely independent points $v_0, v_1, \ldots, v_{n-m}$ such that

$$\text{aff}(P) = \text{aff}(\{v_0, \ldots, v_{n-m}\}).$$

Moreover, the algorithm constructs a point in the relative interior of P.

Proof. Find such a system (resp. point) for

$$P \cap \{x \mid \|x\|_\infty \leq 2^{5n^2\varphi}\}$$

using the algorithm of Lemma (6.5.3) (resp. Remark (6.5.4) (b)). □

(6.5.6) Theorem. *There exists an oracle-polynomial time algorithm that, for any nonempty well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle, finds linearly independent vectors spanning the lineality space of P.*

Proof. By Lemma (6.4.8) we can solve the strong separation problem for $\mathrm{rec}(P)$ and hence also for $\mathrm{lineal}(P) = \mathrm{rec}(P) \cap (-\mathrm{rec}(P))$ in oracle-polynomial time. The result then follows from Theorem (6.5.5). □

(6.5.7) Theorem. *There exists an oracle-polynomial time algorithm that, for any nonempty well-described polyhedron $(P; n, \varphi)$ given by a strong optimization oracle, finds a vector contained in some minimal face of P. In particular, if P is pointed the algorithm obtains a vertex.*

Proof. Without loss of generality we may assume that P is pointed; for otherwise we can replace P by $P \cap (\mathrm{lineal}(P))^{\perp}$. By Lemma (6.4.8), Remark (6.4.10) (b) and Theorem (6.5.5) we find a vector c in the interior of $(\mathrm{rec}(P))^*$. Then the face of P maximizing $c^T x$ over P is a polytope, and we can find a vertex of it by Lemma (6.5.1). □

(6.5.8) Theorem. *There exists an oracle-polynomial time algorithm that, for any given well-described polyhedron $(P; n, \varphi)$, specified by a strong separation oracle, and for any given vector $x_0 \in P$, solves the strong separation problem for the smallest face of P containing x_0.*

Proof. Let

$$Q := \{(z^T, \lambda)^T \in \mathbb{R}^{n+1} \mid z^T x \leq \lambda \text{ for all } x \in P, z^T x_0 = \lambda\}.$$

Then the strong separation problem for Q can be solved in oracle-polynomial time, since the strong violation problem for P can be solved in oracle-polynomial time by Theorem (6.4.9). Thus we can find a point $(c^T, \gamma)^T$ in the relative interior of Q. Then the minimal face containing x_0 is

$$P \cap \{x \mid c^T x = \gamma\}$$

for which the strong separation problem is trivially solvable. □

(6.5.9) Corollary. *There exists an oracle-polynomial time algorithm that, for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle, and for any $c \in \mathbb{Q}^n$, either*

(i) *finds a system $Cx = d$ such that $\{x \in \mathbb{R}^n \mid Cx = d\}$ is a minimal face of the face of optimum solutions of $\max c^T x$, $x \in P$, or*

(ii) *asserts that P is empty or that $c^T x$ is unbounded over P.* □

(6.5.10) Corollary. *There exists an oracle-polynomial time algorithm that, for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle, and for any two rational vectors $x_1, x_2 \in P$, decides whether x_1 and x_2 are vertices of P, and if this is so, whether x_1 and x_2 are adjacent on P.* □

Next we prove an algorithmic version of Carathéodory's theorem.

(6.5.11) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-described polyhedron $(P; n, \varphi)$ given by a strong optimization oracle and for any rational vector $y_0 \in P$, finds affinely independent vertices $x_0, \ldots, x_k$ of P and positive rational numbers $\lambda_0, \ldots, \lambda_k$ with $\lambda_0 + \lambda_1 + \ldots + \lambda_k = 1$ such that $y_0 = \lambda_0 x_0 + \ldots + \lambda_k x_k$.*

Proof. By the algorithms described in Theorems (6.5.8) and (6.5.7) we first find a vertex x_0 of the smallest face F_0 of P containing y. If $y_0 = x_0$ we are done.

If $y_0 \neq x_0$ then draw a semiline starting at x_0 through y_0 and let y_1 be the last point on this semiline contained in P. Similarly as above we find a vertex x_1 of the smallest face F_1 of P containing y_1, and so on. If y_i, F_i, x_i are found and $y_i \neq x_i$, we draw the semiline starting in x_i through y_i and determine the last point, say y_{i+1} of this semiline contained in P. Let x_{i+1} be any vertex of the smallest face F_{i+1} of P containing y_{i+1}. The procedure stops if $x_i = y_i$.

Since for all $i \geq 0$, $y_i \notin F_{i+1}$ and so $x_i \notin F_{i+1}$ but $x_{i+1}, x_{i+2}, \ldots \in F_{i+1}$, the vectors $x_0, x_1, \ldots$ are affinely independent. Hence the procedure stops in $k \leq n$ steps. Clearly, $y_0 \in \operatorname{conv}\{x_0, x_1, \ldots, x_k\}$. Since $x_0, \ldots, x_k$ are affinely independent, the λ_i can be found by elementary linear algebra.

Since each vector y_i is the unique point of intersection of affine subspaces spanned by y_0 and vertices of P the encoding length of y_i can be bounded by a polynomial of n and φ. □

(6.5.12) Corollary. *There exists an oracle-polynomial time algorithm that, for any well-described, pointed, convex polyhedral cone $(P; n, \varphi)$ given by a strong optimization oracle and for any rational vector $y_0 \in P$, finds linearly independent vectors $x_1, \ldots, x_k$ contained in extremal rays of P such that $y_0 = x_1 + \ldots + x_k$.*

Proof. We may assume that $y_0 \neq 0$. By Remark (6.4.10) and Theorem (6.5.5) we can find in oracle-polynomial time a vector c in the interior of the polar cone P^*. Then

$$P' := P \cap \{x \mid c^T x = c^T y_0\}$$

is a polytope containing y_0. By Theorem (6.5.11) we can find affinely independent vertices $v_1, \ldots, v_k$ and positive rationals $\lambda_1, \ldots, \lambda_k$ such that $\lambda_1 + \ldots + \lambda_k = 1$ and $y_0 = \lambda_1 v_1 + \ldots + \lambda_k v_k$. Take $x_i := \lambda_i v_i$, $i = 1, \ldots, k$. □

(6.5.13) Corollary. *There exists an oracle-polynomial time algorithm that, for any well-described, pointed polyhedron $(P; n, \varphi)$ given by a strong optimization oracle and for any rational vector $y_0 \in P$ finds the following:*

(i) *vertices $x_0, \ldots, x_t$, $t \geq 0$, of P,*
(ii) *positive rationals $\lambda_0, \lambda_1, \ldots, \lambda_t$ with $\lambda_0 + \ldots + \lambda_t = 1$,*
(iii) *vectors $y_1, \ldots, y_s$, $s \geq 0$, in extremal rays of $\operatorname{rec}(P)$ such that $x_1 - x_0, \ldots, x_t - x_0, y_1, \ldots, y_s$ are linearly independent and*

$$y_0 = \lambda_0 x_0 + \ldots + \lambda_t x_t + y_1 + \ldots + y_s.$$

Proof. Consider the set

$$C := \{(\lambda x^T, \lambda)^T \in \mathbb{R}^{n+1} \mid x \in P\} \cup \{(x^T, 0)^T \in \mathbb{R}^{n+1} \mid x \in \text{rec}(P)\}.$$

Then it is well-known that C is a pointed polyhedral cone (sometimes called the **1-homogenization** of P) whose extremal rays are generated by the vectors $(v^T, 1)^T$ where v is a vertex of P and by $(e^T, 0)^T$ where e is a nonzero vector in an extremal ray of rec(P). The optimization problem for C is easily reduced to the optimization problem for P.

Thus we can apply Corollary (6.5.12) to $(y_0^T, 1)^T \in C$, which gives the required decomposition. □

The case of nonpointed polyhedra P can be reduced to the pointed case by intersecting P with an affine subspace that contains y_0 and is orthogonal to the lineality space of P. The results are somewhat tedious to formulate and therefore omitted.

Now we turn to the problem of finding "dual" solutions to the "linear program" $\max\{c^T x \mid x \in P\}$, where P is a nonempty polyhedron given by a strong separation oracle and where the maximum exists. Since no system defining P is given explicitly, the set of dual variables is not well-defined. To begin with, let us consider every inequality valid for P as a constraint and define accordingly an **optimum dual solution** as an assignment of positive scalars $\lambda_1, \ldots, \lambda_k$ to a finite number of valid inequalities $a_1^T x \leq \alpha_1, \ldots, a_k^T \leq \alpha_k$ such that $\lambda_1 a_1 + \ldots + \lambda_k a_k = c$ and $\lambda_1 \alpha_1 + \ldots + \lambda_k \alpha_k = \max\{c^T x \mid x \in P\}$. If $a_1, \ldots, a_k$ are linearly independent we call the optimum dual solution **basic**. An optimum dual solution in this general sense can be found trivially by setting $\lambda_1 := 1$, $a_1 := c$, and $\alpha_1 := \max\{c^T x \mid x \in P\}$. To get nontrivial results we shall restrict the family of valid inequalities to be considered.

Two ways of doing this seem meaningful for our purposes: We restrict the valid inequalities $a_i^T x \leq \alpha_i$ to such inequalities where a_i is an output of the strong separation oracle, or to inequalities in a standard representation of P – see Section 0.1 – where each equation is considered as a pair of inequalities. In the first case we shall say that the dual solution is an **optimum dual solution with oracle inequalities**, in the second that it is an **optimum standard dual solution.**

(6.5.14) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle and for any $c \in \mathbb{Q}^n$, either*

(i) *finds a basic optimum standard dual solution, or*
(ii) *asserts that the dual problem is unbounded or has no solution.*

Proof. First we run the algorithm of Corollary (6.5.9) with input P and c. If it asserts that P is empty or $c^T x$ is unbounded over P we can assert that the dual problem is unbounded or has no solution. So we may assume that $\max\{c^T x \mid x \in P\}$ exists. We set $\gamma := \max\{c^T x \mid x \in P\}$ and assume, to begin with, that P is full-dimensional. Consider the polyhedral cone

$$Q := \{(z^T, \lambda)^T \in \mathbb{R}^{n+1} \mid z^T x \leq \lambda \text{ for all } x \in P\}.$$

Since P is full-dimensional, Q is pointed. Moreover, $(c^T, \gamma)^T \in Q$. Thus by Corollary (6.5.12) we can find linearly independent vectors $(a_1^T, \alpha_1)^T, \ldots, (a_k^T, \alpha_k)^T$ contained in extremal rays of Q such that $a_1 + \ldots + a_k = c$ and $\alpha_1 + \ldots + \alpha_k = \gamma$. It is easy to see that every extremal ray of Q either determines a facet of P or is generated by $(0, 0, \ldots, 0, 1)^T$. Because of the definition of γ this latter ray does not occur.

Assume now that P is not full-dimensional. Choose a vector $p \in P$. We first determine the affine hull of P via Theorem (6.5.5). Then we can find c_0 and a_0 such that $c = c_0 + a_0$, c_0 is in the linear space $L = \text{aff}(P) - \{p\}$, and a_0 is orthogonal to L. $P - \{p\}$ is a full-dimensional polyhedron in L. Using the method described above we find a basic optimum standard dual solution for $\max\{c_0^T x \mid x \in P - \{p\}\}$. This basic optimum dual solution together with the equations from (6.5.5) yield the desired dual solution. □

Now we want to find dual solutions with inequalities that are outputs of the separation oracle. However, such a dual solution does not always exist, not even in the full-dimensional case. For instance, if $c^T x \leq \gamma$ defines a facet of P, then the existence of a dual solution as above would imply that a positive multiple of c would occur as an output of the separation oracle. This cannot be expected in general, as for any input $y \notin P$ to the separation oracle there is enough play in it to give as output a separating hyperplane not parallel to any facet. But if we know in advance an upper bound φ on the encoding length of the vectors returned by the separation oracle (which is the usual case in our combinatorial applications) we can find a dual solution in the above sense, even if P is not full-dimensional.

(6.5.15) Lemma. *There exists an oracle-polynomial time algorithm that, for any vector $c \in \mathbb{Q}^n$ and for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle where every output has encoding length at most φ, either*

(i) *finds a basic optimum dual solution with oracle inequalities, or*
(ii) *asserts that the dual problem is unbounded or has no solution.*

Proof. We first check whether P is empty or $c^T x$ is unbounded over P with the algorithm of Corollary (6.5.9). If one of these cases occurs we assert (ii).

Otherwise, let $\varphi' := 6n^2\varphi$. We apply the ellipsoid method of Theorem (6.4.9) to $(P; n, \varphi')$ with the given separation oracle to find $\gamma := \max\{c^T x \mid x \in P\}$. Let $b_1, \ldots, b_N$ be the outputs of the separation oracle that occured during the execution of the ellipsoid method. Now we can find, again with the ellipsoid method, rational numbers $\beta_1, \ldots, \beta_N$ such that $\beta_i = \max\{b_i^T x \mid x \in P\}$. Let $P' := \{x \in \mathbb{R}^n \mid b_1^T x \leq \beta_1, \ldots, b_N^T x \leq \beta_N\}$. Then P' has facet-complexity at most φ'. Let $\gamma' = \max\{c^T x \mid x \in P'\}$. We claim that $\gamma = \gamma'$.

Since $P \subseteq P'$ we know that $\gamma \leq \gamma'$. On the other hand, if the ellipsoid method concludes that $\gamma = \max\{c^T x \mid x \in P\}$ only using the inequalities $b_1^T x \leq \beta_1, \ldots, b_N^T x \leq \beta_N$ and the fact that P has facet-complexity at most φ', it necessarily has to conclude that $\gamma = \max\{c^T x \mid x \in P'\}$ when applied to $(P'; n, \varphi')$. So $\gamma = \gamma'$.

Now

$$\begin{aligned} &\max c^T x \\ &b_1^T x \leq \beta_1 \\ &\vdots \\ &b_N^T x \leq \beta_N \end{aligned}$$

is an explicitly given linear program for which we can find a basic optimum solution of the (explicitly given) dual program

$$\min\{\pi_1\beta_1 + \ldots + \pi_N\beta_N \mid \pi_1, \ldots, \pi_N \geq 0, \pi_1 b_1 + \ldots + \pi_N b_N = c\}$$

with the ellipsoid method. This is a basic optimum dual solution with oracle inequalities for the original problem as required. □

Now we show that any strong separation oracle for $(P; n, \varphi)$ can be modified to a strong separation oracle that always gives facet defining inequalities or implicit equalities as output. More precisely, we can derive a strong facet-separation oracle in the following way.

(6.5.16) Theorem. *The following problem can be solved in oracle-polynomial time for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle:*

Given $y \in \mathbb{Q}^n$, decide whether y is in P. If y is not in P, find a valid inequality $a^T x \leq \alpha$ such that $a^T y > \alpha$ and $P \cap \{x \mid a^T x = \alpha\}$ is a facet of P or is equal to P itself.

Moreover, if the outputs of the strong separation oracle have encoding length at most φ, we can require that the output vector a is an output of the oracle itself.

Proof. We may assume that P is nonempty. Call the separation oracle with the given $y \in \mathbb{Q}^n$. If it concludes that y is in P we are done.

Suppose it returns a vector $c \in \mathbb{Q}^n$ such that $c^T x < c^T y$ for all $x \in P$. Determine $\gamma := \max\{c^T x \mid x \in P\}$. Then $c^T x \leq \gamma$ is a valid inequality and so by Theorem (6.5.14) we can find valid inequalities $a_1^T x \leq \alpha_1, \ldots, a_k^T x \leq \alpha_k$ such that $P \cap \{x \mid a_i^T x = \alpha_i\}$ is either P itself or a facet of P and such that $a_1 + \ldots + a_k = c$ and $\alpha_1 + \ldots + \alpha_k \leq \gamma < c^T y$. Hence for at least one $i \in \{1, \ldots, k\}$ we have $\alpha_i < a_i^T y$ and hence for every $x \in P$, $a_i^T x \leq \alpha_i < a_i^T y$.

Suppose now that the outputs of the separation oracle have encoding length at most φ. Then by the above we can find a valid inequality $a^T x \leq \alpha$ violated by y such that $P \cap \{x \mid a^T x = \alpha\}$ is equal to P or is a facet of P. Then with Lemma (6.5.15) we can find valid inequalities $b_1^T x \leq \beta_1, \ldots, b_k^T x \leq \beta_k$ and positive rationals $\pi_1, \ldots, \pi_k$ such that $\pi_1 b_1 + \ldots + \pi_k b_k = a$ and $\pi_1\beta_1 + \ldots + \pi_k\beta_k = \alpha$ and such that each b_i occured as output of the separation oracle. Clearly, as $a^T y > \alpha$, there exists an $i \in \{1, \ldots, k\}$ with $b_i^T y > \beta_i$. Now again, $P \cap \{x \mid b_i^T x = \beta_i\}$ is equal to P or is a facet of P, since $P \cap \{x \mid a^T x = \alpha\} \subseteq P \cap \{x \mid b_i^T x = \beta_i\}$. □

Using Theorem (6.5.16) we can strengthen Lemma (6.5.15) so that the positive dual variables correspond to facet-defining inequalities and implicit equalities only.

(6.5.17) Theorem. *There exists an oracle-polynomial time algorithm that, for any vector $c \in \mathbb{Q}^n$ and for any well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle where every output has encoding length at most φ, either*

(i) *finds a basic optimum dual solution with oracle inequalities such that each inequality defines either a facet or an implicit equation of P, or*

(ii) *asserts that the dual problem is unbounded or has no solution.*

Proof. Combine Lemma (6.5.15) and Theorem (6.5.16). □

Analogously to the results in Section 4.7, various operations on polyhedra preserve the existence of polynomial time algorithms:

(6.5.18) Exercise. *Let $(P; n, \varphi)$ and $(Q; n, \psi)$ be well-described polyhedra given by strong separation oracles. Prove that the strong separation (optimization, violation) problem is solvable in oracle-polynomial time for each of the following well-described polyhedra:*

(a) $(P \cap Q; n, \max\{\varphi, \psi\})$,
(b) $(P + Q; n, 24n^4(\varphi + \psi))$,
(c) $(\mathrm{conv}(P \cup Q); n, 12n^4 \max\{\varphi, \psi\})$,
(d) $(P^*; n, 4n^2\varphi)$,
(e) $(\mathrm{bl}(P); n, 4n^2\varphi)$,
(f) $(\mathrm{abl}(P); n, 4n^2\varphi)$. □

Finally let us turn to a special nonlinear optimization problem. The problem of minimizing a convex function can also be solved in a strong version for rational polyhedra. A **polyhedral** (or **piecewise linear**) **function** is a function which is the maximum of a finite number of linear functions. We say that the encoding length of a polyhedral function is at most Ψ if each of these linear functions has encoding length at most Ψ.

(6.5.19) Theorem. *The following problem can be solved in oracle-polynomial time:*

Input: *A well-described polyhedron $(P; n, \varphi)$ given by a strong separation oracle and a polyhedral function $f : \mathbb{Q}^n \to \mathbb{Q}$ of encoding length at most Ψ given by an oracle that for any $x \in \mathbb{Q}^n$ returns $f(x)$.*

Output: *A vector $x^* \in P$ minimizing f over P.*

Proof. Exercise. □

*6.6 Strongly Polynomial Algorithms

The ellipsoid method solves linear programs in polynomial time. This means that it solves any given LP-problem

$$\max\{c^T x \mid Ax \leq b\}$$

in at most $p(\langle A, b, c\rangle)$ binary operations, for some polynomial p. Although such an algorithm is by definition "efficient", it can have some drawbacks. In the

ellipsoid method the encoding lengths of the input numbers influence the number of arithmetic steps to be performed. This does not conflict with the definition of polynomiality, but it would be nicer, if the encoding lengths of the input numbers only influenced the numbers to which these arithmetic operations are applied and if the number of arithmetic operations could be bounded polynomially by the problem dimensions (i. e., by the number of rationals in the input). An algorithm with these properties is called strongly polynomial (see Section 1.3). More precisely, an algorithm is **strongly polynomial** if:

(a) it consists of the (elementary) arithmetic operations: addition, subtraction, comparison, multiplication and division;
(b) the number of times such operations are performed is polynomially bounded in the dimension of the input;
(c) the encoding lengths of the numbers occurring during the algorithm are polynomially bounded in the encoding length of the input.

MEGIDDO (1983) showed that any linear program with n variables and m constraints, such that each of the constraints, and the objective function, has at most two nonzero coefficients, can be solved in $O(mn^3 \log n)$ arithmetic operations. Another interesting result was obtained by TARDOS (1985, 1986), who showed that any linear program $\max\{c^T x \mid Ax \leq b\}$ can be solved in at most $p(\langle A \rangle)$ arithmetic operations on numbers of size polynomially bounded by $\langle A, b, c \rangle$, where p is a polynomial. Thus the encoding length of b and c do not contribute to the number of arithmetic steps. In particular, there is a strongly polynomial algorithm for the class of LP-problems where A is a $\{0, \pm 1\}$-matrix. This class includes network flow problems – see Section 7.1 – and also multicommodity flow problems – see Section 8.6. Therefore, Tardos' result answers a long-standing open question asked by EDMONDS and KARP (1972).

FRANK and TARDOS (1987) extended this result to "nonexplicit" linear programs, and showed thereby the strongly polynomial time solvability of most polynomially solvable combinatorial optimization problems.

First we will prove Tardos' results, using simultaneous diophantine approximation, in particular the Frank-Tardos rounding procedures (6.2.13). We remark that Tardos' algorithm avoids the use of diophantine approximation – of course at the expense of more subtle considerations. The heart of the procedure is the following lemma.

(6.6.1) Lemma. *Let $A \in \mathbb{Q}^{m \times n}$, $b, \tilde{b} \in \mathbb{Q}^m$ and $c, \tilde{c} \in \mathbb{Q}^n$ be given. Assume that for each linear inequality $p^T y \leq \pi$ ($p \in \mathbb{Q}^m$, $\pi \in \mathbb{Q}$) with encoding length at most $12m^2(n+m)^2(\langle A \rangle + m + 2)$ we have $p^T b \leq \pi$ if and only if $p^T \tilde{b} \leq \pi$. Similarly assume that for each linear inequality $r^T x \leq \rho$ ($r \in \mathbb{Q}^n$, $\rho \in \mathbb{Q}$) with encoding length at most $12n^2(n+m)^2(\langle A \rangle + n + 2)$ we have $r^T c \leq \rho$ if and only if $r^T \tilde{c} \leq \rho$.*

(a) *For each optimum solution $\tilde{x}$ of the program $\max\{\tilde{c}^T x \mid Ax \leq \tilde{b}\}$ there exists an optimum solution $\bar{x}$ of the program $\max\{c^T x \mid Ax \leq b\}$ such that for all $1 \leq i \leq m$ $A_i.\tilde{x} = \tilde{b}_i$ iff $A_i.\bar{x} = b_i$.*
(b) *For each optimum solution $\tilde{y}$ of the dual program $\min\{\tilde{b}^T y \mid A^T y = \tilde{c}, y \geq 0\}$ there exists an optimum solution $\bar{y}$ of $\min\{b^T y \mid A^T y = c, y \geq 0\}$ such that $\tilde{y}$ and $\bar{y}$ have the same support.*

Proof. We prove (a) and (b) simultaneously. We may assume that the matrix A has the form $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ and that $\tilde{b} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}$ such that $A_1\tilde{x} = \tilde{b}_1$, $A_2\tilde{x} < \tilde{b}_2$. Consider the polyhedron

$$P := \{(x^T, z_1^T, z_2^T)^T \in \mathbb{R}^{n+m} \mid A_1x - z_1 = 0,\ A_2x - z_2 \leq 0\}.$$

Then $(\tilde{x}^T, \tilde{b}^T)^T \in P$, and moreover, since $(\tilde{x}^T, \tilde{b}^T)^T$ satisfies each inequality in the definition of P with strict inequality, $(\tilde{x}^T, \tilde{b}^T)^T$ is in the relative interior of P. It also follows that the definition of P contains no implicit equations. Let

$$P' := \{z \in \mathbb{R}^m \mid \exists\, x \in \mathbb{R}^n \text{ such that } (x^T, z^T)^T \in P\}.$$

It follows from the definition that $\tilde{b}$ is in the relative interior of P'. Since the vertex-complexity of P' is not larger than the vertex-complexity of P we obtain from Lemma (6.2.4) that the facet-complexity of P' is at most $12m^2(n+m)^2(\langle A\rangle + m + 2)$. By the hypothesis of the lemma, b is also in the relative interior of P'. Hence there is a vector $\bar{x} \in \mathbb{R}^n$ such that $(\bar{x}^T, b^T)^T$ is in the relative interior of P, i. e., $A_1\bar{x} = b_1$ and $A_2\bar{x} < b_2$ where $b = (b_1^T, b_2^T)^T$ corresponds to the partition of $\tilde{b}$. By the same projection argument we can find a vector $\bar{y} \in \mathbb{R}^m$ such that $A^T\bar{y} = c$, $\bar{y} \geq 0$, and $\bar{y}$ has the same support as $\tilde{y}$. It follows from the Complementary Slackness Theorem (0.1.50) that $\bar{x}$ and $\bar{y}$ form a pair of optimum primal and dual solutions for $\max\{c^Tx \mid Ax \leq b\}$. □

(6.6.2) Remark. Note that Lemma (6.6.1) implies that if one of the two linear programs $\max\{c^Tx \mid Ax \leq b\}$ and $\max\{\tilde{c}^Tx \mid Ax \leq \tilde{b}\}$ has an optimum solution so does the other. By considering the objective function $c = \tilde{c} = 0$, we see that if one of these linear programs is infeasible so is the other. Hence if one of them is unbounded so is the other. □

Now we can prove a result due to TARDOS (1986).

(6.6.3) Theorem. *There exists a polynomial time algorithm that, for any given $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, solves the linear program $\max\{c^Tx \mid Ax \leq b\}$. The number of elementary arithmetic operations used by the algorithm is bounded by a polynomial in $\langle A\rangle$. The algorithm finds an optimum vertex solution if one exists.*

Proof. Using the algorithm of Frank and Tardos – see Theorem (6.2.13) – for Problem (6.2.11), we find vectors $\tilde{c}$ and $\tilde{b}$ with the properties required in Lemma (6.6.1) such that

$$\langle \tilde{b}\rangle \leq 528m^3(m+1)^2(n+m)^2(\langle A\rangle + m + 2),$$
$$\langle \tilde{c}\rangle \leq 528n^3(n+1)^2(n+m)^2(\langle A\rangle + n + 2).$$

So the encoding lengths of $\tilde{b}$ and $\tilde{c}$ are bounded by a polynomial in $\langle A \rangle$. Thus we can solve the linear program

$$\max\{\tilde{c}^T x \mid Ax \leq \tilde{b}\}$$

in time polynomial in $\langle A \rangle$ by the ellipsoid method.

If this linear program is infeasible or unbounded, we are done by Remark (6.6.2). Otherwise we use the algorithm of Theorem (6.5.7) to find a vector $\tilde{x}$ that is on a minimal face of the optimum face. This can be done in time polynomial in $\langle A, \tilde{b}, \tilde{c} \rangle$ and hence in $\langle A \rangle$. We partition A and $\tilde{b}$ such that $A_1\tilde{x} = \tilde{b}_1$ and $A_2\tilde{x} < \tilde{b}_2$. Let $b = (b_1^T, b_2^T)^T$ be the corresponding partition of b.

Claim. Every solution of $A_1x = b_1$ is an optimum solution of $\max\{c^T x \mid Ax \leq b\}$.

Proof. By Lemma (6.6.1) (a) there is an optimum solution $\bar{x}$ of this LP such that $A_1\bar{x} = b_1$ and $A_2\bar{x} < b_2$. By complementary slackness, it suffices to show that every solution of $A_1x = b_1$ satisfies $Ax \leq b$. Suppose not. Then there is a vector y satisfying $Ay \leq b$, $A_1y = b_1$ and at least one of the inequalities $A_2y \leq b_2$ with equation. Using Lemma (6.6.1) (a) again, there is an optimum solution x^* of $\max\{\tilde{c}^T x \mid Ax \leq \tilde{b}\}$ satisfying $A_1x^* = \tilde{b}_1$ and satisfying at least one of the inequalities $A_2x^* \leq \tilde{b}_2$ with equation. So x^* is contained in a smaller face than $\tilde{x}$, which contradicts the choice of $\tilde{x}$. This proves the claim.

Using Gaussian elimination, we find a solution of the equation system $A_1x = b_1$. This is an optimum solution of $\max\{c^T x \mid Ax \leq b\}$ which lies on the minimal face. In particular, if the set of feasible solutions is pointed, it is a vertex solution. □

(6.6.4) Corollary. *There exists a strongly polynomial time algorithm for rational LP-problems with $\{0, \pm 1\}$-constraint matrix.* □

In this section we have so far considered explicitly given linear programs only. We shall now give a general result about optimization problems for polyhedra and their dual problems in "strongly polynomial" time. More exactly, we show that the number of elementary arithmetic operations to solve an optimization problem over a well-described polyhedron and to solve its dual problem does not depend on the encoding length of the objective function.

(6.6.5) Theorem. *There exist algorithms that, for any well-described polyhedron $(P; n, \varphi)$ specified by a strong separation oracle, and for any given vector $c \in \mathbb{Q}^n$,*

(a) *solve the strong optimization problem* $\max\{c^T x \mid x \in P\}$, *and*
(b) *find an optimum vertex solution of* $\max\{c^T x \mid c \in P\}$ *if one exists, and*
(c) *find a basic optimum standard dual solution if one exists.*

The number of calls on the separation oracle, and the number of elementary arithmetic operations executed by the algorithms are bounded by a polynomial in φ. All arithmetic operations are performed on numbers whose encoding length is bounded by a polynomial in $\varphi + \langle c \rangle$.

Proof. Using the algorithm of Lemma (6.2.19) we can find a vector $\tilde{c} \in \mathbb{Q}^n$ such that

$$\langle \tilde{c} \rangle \leq 88(n+1)^3(4n^2\varphi + 1) < 400\varphi^6$$

and the objective functions $c^T x$ and $\tilde{c}^T x$ are maximized by the same vectors in P. Now we solve $\max\{\tilde{c}^T x \mid x \in P\}$ using the algorithm of Theorem (6.4.9). The size of the input to this algorithm is bounded by a polynomial in φ, and hence so is its running time. The output of this algorithm is also a valid output for $\max\{c^T x \mid x \in P\}$. This solves (a). By Remark (6.5.2) we can find a vertex solution of $\max\{\tilde{c}^T x \mid x \in P\}$, if any. This is also a vertex solution of $\max\{c^T x \mid x \in P\}$.

To get an optimum dual solution (if there is any), we have to round c a little more carefully. Using the algorithm of Theorem (6.2.13) for Problem (6.2.12), we find a vector $\tilde{c} \in \mathbb{Q}^n$ such that

(i) for each $y \in \mathbb{Q}^n$ with encoding length at most $105n^4\varphi$, we have $c^T y \leq 0$ if and only if $\tilde{c}^T y \leq 0$,
(ii) $\langle \tilde{c} \rangle \leq 4620n^5(n+1)^2\varphi$.

Now we find a basic optimum standard dual solution for $\max\{\tilde{c}^T x \mid x \in P\}$, say $\lambda_1, \ldots, \lambda_k > 0$ and $a_1^T x \leq \alpha_1, \ldots, a_k^T x \leq \alpha_k$. So $\tilde{c} = \lambda_1 a_1 + \ldots + \lambda_k a_k \in \text{cone}\{a_1, \ldots, a_k\}$. Since by Lemma (6.2.3) $\text{cone}\{a_1, \ldots, a_k\}$ has vertex-complexity at most $35n^2\varphi$, by Lemma (6.2.4) it has facet-complexity at most $105n^4\varphi$. Hence by (i), the vector c is in this cone. Since the vectors a_i are linearly independent, we can use Gaussian elimination to determine scalars $\mu_1, \ldots, \mu_k \geq 0$ such that $c = \mu_1 a_1 + \ldots + \mu_k a_k$. By complementary slackness, $\mu_1, \ldots, \mu_k$ and the inequalities $a_1^T x \leq \alpha_1, \ldots, a_k^T x \leq \alpha_k$ form a basic optimum standard dual solution. It is obvious that the running times are as claimed. □

An important application of this theorem will be that one can turn many polynomial time combinatorial optimization algorithms into strongly polynomial algorithms. For, consider a class of polyhedra defined, e. g., by graphs. If we can solve the optimization problem $\max\{c^T x \mid x \in P\}$ for each polyhedron P in this class in time polynomial in $\langle c \rangle$ and in the encoding length of the associated graph, then we can also solve the optimization problem in strongly polynomial time, i. e., with an algorithm in which the number of arithmetic operations does not depend on $\langle c \rangle$.

*6.7 Integer Programming in Bounded Dimension

Recall from Section 0.1 that the linear diophantine inequalities problem is as follows:

(6.7.1) Linear Diophantine Inequalities Problem. *Given a system of linear inequalities*

$$a_i^T x \leq b_i \quad (i = 1, \ldots, m), \tag{6.7.2}$$

where $a_i \in \mathbb{Q}^n$, $b_i \in \mathbb{Q}$, either find an integral solution or conclude that no integral solution exists.

In this section we show that this problem is solvable in polynomial time for every fixed n, a celebrated result due to LENSTRA (1983). The approach given here is described in GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1984a).

First we solve the following problem.

(6.7.3) *Given a system of linear inequalities* (6.7.2), *either*

(i) *find an integral solution x of* (6.7.2), *or*

(ii) *find a vector $c \in \mathbb{Z}^n \setminus \{0\}$ such that $c^T(x - x') < n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}$ for any two (not necessarily integral) solutions x, x' of* (6.7.2).

Roughly speaking, by solving this auxiliary problem (6.7.3) we either find a solution of the integer programming problem or find a direction in which the solution set is "flat". The important thing about (6.7.3) is that it can be solved in polynomial time even for varying n.

(6.7.4) Theorem. *Problem* (6.7.3) *can be solved in polynomial time.*

Proof. I. First we show that the problem can be reduced to the case when the solution set P of (6.7.2) is bounded. Let φ be the maximum encoding length for the inequalities in (6.7.2). Then the vertex-complexity of P is at most $\nu = 4n^2\varphi$.

Consider the inequality system

$$a_i^T x \leq b_i \quad (i = 1, \ldots, m), \tag{6.7.5}$$
$$\|x\|_\infty \leq 2^{2\nu+1} n(n+1)2^{\frac{1}{2}\binom{n}{2}}.$$

Suppose we can solve problem (6.7.3) for this system of linear inequalities. If the output is an integral solution, then we are done. Suppose that the output is an integral vector $c \neq 0$ such that $\gamma \leq c^T x \leq \gamma + n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}$ for some $\gamma \in \mathbb{Q}$ and for all solutions x of (6.7.5). Then we claim the same is true for all solutions of (6.7.2). In fact, every solution of (6.7.2) can be written in the form

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{b} \mu_j u_j,$$

where $\lambda_i, \mu_j \geq 0$, $\sum \lambda_i = 1$, and $v_i \in P$, $u_j \in \operatorname{rec}(P)$ and $\langle u_j \rangle, \langle v_i \rangle \leq \nu$. Then $\|v_i\| \leq 2^\nu$ and hence

$$x' := \sum_{i=1}^{k} \lambda_i v_i$$

is a solution of (6.7.5). So

$$\gamma \leq c^T x' \leq \gamma + n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}.$$

Furthermore, we claim that $c^T u_i = 0$. Suppose not; then $|c^T u_i| > 2^{-v}$, say $c^T u_i > 2^{-v}$. Consider the vector

$$x'' := x' + 2^v n(n+1)2^{\frac{1}{2}\binom{n}{2}+1} u_i.$$

Then

$$c^T x'' > \gamma + n(n+1)2^{\frac{1}{2}\binom{n}{2}+1},$$

which is a contradiction since x'' is a solution of (6.7.5). This contradiction proves that $c^T u_i = 0$ for all i and hence

$$\gamma \leq c^T x \leq \gamma + n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}$$

as claimed.

II. So we may assume that $P \subseteq S(0, 2^v)$, where $v := 4n^2\varphi$. Let us define a shallow separation oracle for P as follows. Given an ellipsoid $E(A, a)$, check whether P contains the concentrical ellipsoid $E((n+1)^{-2}A, a)$. This can be done by checking, for $i = 1, \ldots, m$, whether $(n+1)^{-2} a_i^T A a_i \leq (b_i - a_i^T a)^2$ holds. If so, declare $E(A, a)$ "tough". If not, then for any index i violating it, the inequality $a_i^T x \leq b_i$ is a shallow cut.

Apply the shallow-cut ellipsoid method with this shallow separation oracle, with $R = 2^v$ and $\varepsilon = 1$.

Case 1. The output is an ellipsoid $E(A, a)$ containing P, of volume at most 1. Then consider the norm $N(x) := \sqrt{x^T A x}$. By Theorems (5.3.16) and (5.3.13), we find a vector $c \in \mathbb{Z}^n$ such that $N(c) \leq 2^{(n-1)/4} \cdot (\det A)^{1/(2n)} \leq 2^{(n-1)/4} V_n^{-1/n} \leq n \cdot 2^{(n-1)/4}$.

Hence for any vector $x \in P$, we have $c^T(x-a) \leq (c^T A c)((x-a)^T A^{-1}(x-a)) \leq n^2 2^{(n-1)/2} < n(n+1) \cdot 2^{\binom{n}{2}/2}$. So c is a vector satisfying (b) of (6.7.3).

Case 2. Suppose the shallow-cut ellipsoid method ends up with a tough ellipsoid $E(A, a)$. Consider the norm $N(x) := \sqrt{x^T A^{-1} x}$. By Theorems (5.3.16) and (5.3.13), we find a basis $\{b_1, \ldots, b_n\}$ of $\mathbb{Z}^n$ such that

$$N(b_1) \cdot \ldots \cdot N(b_n) \leq 2^{\frac{1}{2}\binom{n}{2}} (\det A^{-1})^{\frac{1}{2}}.$$

Write

$$a = \sum_{i=1}^{n} \lambda_i b_i$$

and consider the vector

$$a' := \sum_{i=1}^{n} \lfloor \lambda_i \rfloor b_i \in \mathbb{Z}^n.$$

If $a' \in P$ we are done, so suppose that $a' \notin P$. Then a' is not contained in the concentrical ellipsoid $E((n+1)^{-2}A, a)$, and hence $N(a'-a) > (n+1)^{-1}$. But

$$N(a'-a) = N\left(\sum_{i=1}^{n} (\lambda_i - \lfloor \lambda_i \rfloor) b_i\right) \leq \sum_{i=1}^{n} N(b_i).$$

Hence there is an index i, $1 \leq i \leq n$, so that $N(b_i) > (n(n+1))^{-1}$. Without loss of generality, assume that $i = n$.

Define an integral vector c by

$$c^T x = \det(b_1, \ldots, b_{n-1}, x).$$

Then for any $x \in P$, we can estimate $c^T(x-a)$ as follows. Write $A^{-1} = B^T B$. Then

$$\begin{aligned} |c^T(x-a)| &= |\det(Bb_1, \ldots, Bb_{n-1}, B(x-a))|(\det A)^{\frac{1}{2}} \\ &\leq N(b_1)\ldots N(b_{n-1})N(x-a)(\det A)^{\frac{1}{2}} \\ &\leq 2^{\frac{1}{2}\binom{n}{2}}(\det A^{-1})^{\frac{1}{2}} \frac{1}{N(b_n)}(\det A)^{\frac{1}{2}} \\ &\leq n(n+1)2^{\frac{1}{2}\binom{n}{2}}. \end{aligned}$$

So c is a vector satisfying (b) of (6.7.3). □

Let us mention that, with the methods of BABAI (1986) and LOVÁSZ (1986) the factor $2^{\frac{1}{2}\binom{n}{2}}$ could be improved to c^n, where c is some positive contstant – cf. also Theorem (5.3.26).

We are now able to derive the main result of this section, the theorem of LENSTRA (1983).

(6.7.6) Theorem. *For each fixed integer $n \geq 1$, the linear diophantine inequalities problem* (6.7.1) *for n variables can be solved in polynomial time.*

Proof. By induction on n. Let us run the algorithm of Theorem (6.7.4) first, with the given system of inequalities as input. If its output is an integral solution, we are done. Suppose that its output is an integer vector $c \neq 0$ such that $c^T(x_1 - x_2) \leq n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}$ for any two solutions x_1 and x_2 of (6.7.2). Let

$$\gamma_1 = \min\{c^T x \mid x \text{ satisfies (6.7.2)}\}$$

and

$$\gamma_2 = \max\{c^T x \mid x \text{ satisfies (6.7.2)}\}.$$

For every integer k with $\gamma_1 \leq k \leq \gamma_2$, consider the diophantine inequalities problem for the system of linear inequalities (6.7.2) joined with

$$c^T x = k. \tag{6.7.7}$$

We can transform the system (6.7.2), (6.7.7) by eliminating one of the variables to a form

$$A_k x \leq b_k \tag{6.7.8}$$

where $A_k \in \mathbb{Q}^{(n-1)\times m}$ and $b_k \in \mathbb{Q}^m$. Now it is clear that (6.7.2) has an integral solution if and only if (6.7.8) has an integral solution for at least one k, $\gamma_1 \leq$

$k \leq \gamma_2$. Thus to solve (6.7.2) it suffices to solve at most $\lfloor \gamma_2 - \gamma_1 \rfloor + 1 \leq 2n(n+1)2^{\frac{1}{2}\binom{n}{2}+1} + 1$ diophantine inequalities problems with $n-1$ variables. By the induction hypothesis, these can be solved in polynomial time. Since n (and thus $2n(n+1)2^{\frac{1}{2}\binom{n}{2}+1}$) is a constant, it follows that (6.7.2) can be solved in integers in polynomial time.

We have to point out that we omitted an important detail: one must show that the encoding length for (6.7.8) is bounded by a polynomial of the encoding length of (6.7.2), uniformly for all k with $\gamma_1 \leq k \leq \gamma_2$. This is, however, easily checked. □

By the same method one can prove the following theorems.

(6.7.9) Theorem. *There exists an oracle-polynomial time algorithm that, for any well-bounded convex body $(K; n, R, r)$ given by a weak separation oracle, either*

(i) *finds an integral point $x \in K$, or*

(ii) *finds a nonzero integral vector $c \in \mathbb{Z}^n$ such that $|c^T x - c^T x'| \leq n^3 2^{\frac{1}{2}\binom{n}{2}+1}$ for any two vectors $x, x' \in K$.*

□

(6.7.10) Theorem. *For any fixed integer $n \geq 1$, there exists an oracle-polynomial time algorithm that, for any well-bounded convex body $(K; n, R, r)$ given by a weak separation oracle, and for any rational number $\varepsilon > 0$, either finds an integral point in $S(K, \varepsilon)$, or concludes that K contains no integral point.* □

Finally, Lenstra's theorem (6.7.6) immediately yields – via binary search – the following important corollary for integer programming.

(6.7.11) Corollary. *For any fixed integer $n \geq 1$, the integer linear programming problem $\max\{c^T x \mid Ax \leq b, x \text{ integral}\}$ in n variables can be solved in polynomial time.* □

A similar corollary for the integer linear programming problem over well-described polyhedra $(P; n, \varphi)$ can be derived from Theorem (6.7.10).

One may wonder whether the polynomial time algorithms for the integer programming problems discussed above may be modified to run in strongly polynomial time. However, in our model of strongly polynomial computation even such a simple problem as "Given an integer a, find an integral solution of $2x = a$" cannot be solved in strongly polynomial time, i. e., the parity of an integer a cannot be decided in a fixed number of elementary arithmetic operations.

Chapter 7

Combinatorial Optimization: Some Basic Examples

In the remaining part of this book we apply the methods developed in the first part to combinatorial optimization. In this chapter we give some illuminating examples to explain the basic techniques of deriving polynomial time algorithms for combinatorial optimization problems. These techniques are based on combining the ellipsoid method and basis reduction with results from the field called "polyhedral combinatorics", where combinatorial optimization problems are formulated as linear programs. Chapter 8 contains a comprehensive survey of combinatorial problems to which these methods apply. Finally, in the last two chapters we discuss some more advanced examples in greater detail.

7.1 Flows and Cuts

A well-known problem, with many real-world applications, is the **maximum flow problem**. An instance of this is given by a directed graph $D = (V, A)$, a "source" $r \in V$, a "sink" $s \in V \setminus \{r\}$, and a "capacity function" $c : A \to \mathbb{Q}_+$. We want to send through this "capacitated network" a maximum amount of "flow" from r to s such that, for any arc, the flow going through it does not exceed its capacity. Here a flow should satisfy the conservation law, i.e., in any node $v \neq r, s$ the total flow entering v is equal to the total flow leaving v. The value (or amount) of flow means the "net flow" leaving the source r, that is the total flow leaving r minus the total flow entering r. Because of the conservation law this is equal to the net flow entering the sink s.

Recall that, for a set W of nodes, we denote the set of arcs of D entering W by $\delta^-(W)$ and leaving W by $\delta^+(W)$, while we write $\delta^-(v)$ and $\delta^+(v)$ instead of $\delta^-(\{v\})$ and $\delta^+(\{v\})$. Moreover, if $x \in \mathbb{R}^A$ and $A' \subseteq A$ is a set of arcs, then $x(A') := \sum_{a \in A'} x_a$. Using this notation we can formulate the maximum flow problem as a linear programming problem as follows.

$$\begin{aligned} (7.1.1) \qquad & \max\ x(\delta^+(r)) - x(\delta^-(r)) \\ & x(\delta^-(v)) - x(\delta^+(v)) = 0 \quad \text{for all } v \in V \setminus \{r, s\}, \\ & 0 \leq x_a \leq c_a \quad \text{for all } a \in A. \end{aligned}$$

Every vector $x \in \mathbb{R}^A$ satisfying the constraints of (7.1.1) is called an (r, s)-**flow subject to** c or just a **flow** (from r to s), and its **value** is $x(\delta^+(r)) - x(\delta^-(r))$. FORD and FULKERSON (1957) devised an algorithm to solve this max-flow problem,

which DINITS (1970) showed to be implementable as a strongly polynomial time algorithm. The polynomial time solvability also follows directly from Khachiyan's version of the ellipsoid method for linear programming, since the linear program (7.1.1) can be set up in time polynomial in the encoding length of the input data V, A, c, r, s. (Khachiyan's method does not directly give strong polynomiality, but indirectly it follows with Tardos' method – see Corollary (6.6.4).)

If the capacities c_a are integral, it turns out that Ford and Fulkerson's algorithm gives an integer valued optimum flow x. So the Ford and Fulkerson algorithm solves at the same time the integer linear programming problem obtained from (7.1.1) by adding the condition

$$x_a \in \mathbb{Z} \text{ for all } a \in A. \tag{7.1.2}$$

It also follows that, if the capacities are integers, adding (7.1.2) to (7.1.1) does not make the maximum value smaller. This is a prime example of an integer linear program in which the integrality conditions turn out to be superfluous. In fact, Ford and Fulkerson showed that the feasible region of (7.1.1) is a polytope with integral vertices if the capacities are integral. Since we can find a vertex solution with the ellipsoid method we can therefore also find an integer solution to (7.1.1) with the ellipsoid method.

Let us now consider the dual program of (7.1.1). By introducing variables y_a for all $a \in A$ and z_v for all $v \in V$ we can write the dual as follows:

$$\begin{aligned} &\min \sum_{a \in A} c_a y_a \\ &z_w - z_v + y_a \geq 0 \quad \text{for all } a = (v, w) \in A, \\ &z_r = 1, \\ &z_s = 0, \\ &y_a \geq 0 \quad \text{for all } a \in A. \end{aligned} \tag{7.1.3}$$

We want to show that this dual program has a combinatorial interpretation which yields one of the central min-max relations in combinatorial optimization, the max-flow min-cut theorem.

If W is a node set with $r \in W$ and $s \notin W$, then it is common to call the arc set $\delta^+(W)$ $(= \delta^-(V \setminus W))$ a **cut separating r from s**, or just an **(r,s)-cut**. The name derives from the fact that if the arc set $\delta^+(W)$ is removed from D, then all the links from r to s are cut and there is no way to send flow of positive value from r to s. Since every (r,s)-flow x satisfies the capacity constraints, we have $x(\delta^+(W)) \leq c(\delta^+(W))$ for all cuts $\delta^+(W)$ separating r from s. For all (r,s)-cuts we have $x(\delta^+(W)) - x(\delta^-(W)) = x(\delta^+(r)) - x(\delta^-(r))$ from the conservation law, thus the maximum flow value through the capacitated network is not larger than the minimum capacity of a cut separating r from s. To see that these two quantities are equal, we construct an (r,s)-cut whose capacity is equal to the maximum value of a flow as follows.

Let y_a, $a \in A$, and z_v, $v \in V$, be an optimum solution (possibly fractional) to (7.1.3) and let x be an optimum solution to (7.1.1), i. e., an optimum flow. Next let $V' := \{v \in V \mid z_v > 0\}$. If $a = (v,w) \in \delta^+(V')$ then $y_a \geq z_v - z_w > 0$, and

hence by the Complementary Slackness Theorem (0.1.50) we have $x_a = c_a$. If $a = (v, w) \in \delta^-(V')$, then $z_w - z_v + y_a \geq z_w - z_v > 0$, and so again by complementary slackness $x_a = 0$. Therefore

$$c(\delta^+(V')) = x(\delta^+(V')) - x(\delta^-(V')) = x(\delta^+(r)) - x(\delta^-(r)),$$

which is the net amount of flow leaving r. So $\delta^+(V')$ is a cut separating r from s with capacity equal to the maximum value of a flow from r to s. Thus we have the following famous result due to FORD and FULKERSON (1956) and ELIAS, FEINSTEIN and SHANNON (1956).

(7.1.4) Max-Flow Min-Cut Theorem. *The maximum value of a flow from r to s subject to the capacities is equal to the minimum capacity of a cut separating r from s.* □

The (constructive) argument above shows that if we have an optimum solution to (7.1.3) we can find a minimum capacity cut separating r from s in polynomial time. As we can find an optimum solution to (7.1.3) with the ellipsoid method, it follows that we can find a minimum capacity cut in polynomial time. Actually, also the maximum flow algorithm of Ford and Fulkerson yields, as a by-product, a minimum cut.

Note that every cut separating r from s defines a feasible $\{0,1\}$-solution of (7.1.3), and a minimum cut defines an optimum solution. Thus (7.1.3) always has an integral optimum solution. So, similarly to (7.1.1), adding integrality conditions to (7.1.3) does not change the optimum value.

The example of flows and cuts discussed above is an instance of a more general phenomenon: total unimodularity of matrices. A matrix M is called **totally unimodular** if every square submatrix of it has determinant 0, +1, or −1. In particular, each entry of M is 0, +1, or −1. There are quite a number of useful characterizations of totally unimodular matrices – see SCHRIJVER (1986) for a survey. For our purposes, the following observation – due to HOFFMAN and KRUSKAL (1956) – is of special interest.

(7.1.5) Theorem. *If M is a totally unimodular matrix and b is an integral vector, then for each objective function c the linear programming problem*

$$\max\{c^T x \mid Mx \leq b\}$$

has an integral optimum solution (provided the maximum is finite).

Proof. If the maximum is finite, M has a submatrix M' with full row rank so that each solution of $M'x = b'$ is an optimum solution of the linear program (here b' denotes the subvector of b corresponding to M'). Without loss of generality $M' = (M_1, M_2)$ with M_1 nonsingular. Then

$$x^* := \begin{pmatrix} M_1^{-1} b' \\ 0 \end{pmatrix}$$

is an optimum solution, which is integral, since M_1^{-1} is an integer matrix by the total unimodularity of M. □

If M is totally unimodular then the matrices M^T, $-M$, $(M,-M)$, (I,M), $(I,-I,M,-M)$, and M^{-1} (if M is nonsingular) and all submatrices of M are also totally unimodular. Theorem (7.1.5) thus yields that each of the following linear programs has integral optimum solutions (for integral vectors l, u, b, b', and c)

$$\begin{aligned}(7.1.6)\qquad &\max\{c^T x \mid Mx \leq b, x \geq 0\} = \min\{y^T b \mid y^T M \geq c^T, y \geq 0\},\\ &\max\{c^T x \mid Mx = b, x \geq 0\} = \min\{y^T b \mid y^T M \geq c^T\},\\ &\max\{c^T x \mid b' \leq Mx \leq b,\ l \leq x \leq u\}.\end{aligned}$$

This implies that the constraint systems of all linear programs in (7.1.6) are totally dual integral and thus totally primal integral – see Section 0.1. Note also that Theorem (7.1.5) implies that each face of $P := \{x \mid Mx \leq b\}$ contains integral vectors. (Polyhedra with this property are called **integral**.) So if P has vertices, all vertices are integral. Therefore, if the linear program stated in Theorem (7.1.5) has a finite optimum solution and the polyhedron P has vertices (this is equivalent to: M has full column rank) then we can find an integral optimum solution with the ellipsoid method. (If P has no vertices we can also find an optimum solution by adding appropriate upper and lower bounds on the variables. More generally, for any integral polyhedron P we can proceed in polynomial time, as follows. We find a system of equations $Cx = d$ determining a minimal face of the optimum face with the algorithm of Corollary (6.5.9). Then we can find an integral vector in this minimal face with the algorithm of (1.4.21).)

Totally unimodular matrices come up in some combinatorial problems. A prime example is the incidence matrix M of a digraph $D = (V, A)$. M is a matrix whose rows are indexed by the nodes and whose columns are indexed by the arcs of D. An entry M_{va} of M is equal to 1 if v is the head of a, is equal to -1, if v is the tail of a, and is 0 otherwise. Note that the matrix of the equation system in (7.1.1) is obtained from the incidence matrix of D by deleting the rows corresponding to the source r and the sink s. Then Theorem (7.1.5) gives that, if the capacities are integral, the optimum flow can be chosen to be integral.

For an undirected graph $G = (V, E)$, the incidence matrix M has $|E|$ columns and $|V|$ rows, and an entry M_{ve} is equal to 1 if node v is an endnode of edge e, and is 0 otherwise. If G contains an odd cycle, then the incidence matrix of this cycle is a nonsingular square submatrix of the incidence matrix of G with determinant $+2$ or -2. In fact, it is easy to see by induction that the incidence matrix M of a graph G is totally unimodular if and only if G contains no odd cycle, i. e., if and only if G is bipartite. This observation goes back to EGERVÁRY (1931).

Another interesting class of totally unimodular matrices consists of those matrices that have the consecutive-ones-property. These are matrices A in which every row $A_{i.}$ looks like $A_{i.} = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$, i. e., there are at most three strings of equal numbers, the first string is a string of zeros which is followed by a string of ones which is followed by a string of zeros. (Some of the strings can be empty.)

SEYMOUR (1980a) gave a characterization of totally unimodular matrices showing that all totally unimodular matrices arise by applying certain operations

to incidence matrices of digraphs and to two special 5×5-matrices. This characterization can be used to determine in polynomial time whether a matrix is totally unimodular or not. Thus, totally unimodular matrices are algorithmically "well under control". The fastest method for testing total unimodularity known at present is the algorithm of TRUEMPER (1987).

7.2 Arborescences

The example of maximum flows and minimum cuts (or more generally, of totally unimodular matrices) is not typical for the combinatorial applications of the ellipsoid method. These problems can be formulated as explicit linear programs and therefore their polynomial time solvability follows already from the polynomial time solvability of explicitly given linear programs. So in these cases the power of the ellipsoid method as described in the foregoing chapters is not fully exploited. In this and the next sections we shall see some more illustrative examples.

Let $D = (V, A)$ be a digraph, let $r \in V$ be a fixed node, called the **root**. A set A' of arcs is called an **arborescence rooted at** r or an r-**arborescence** if the arcs in A' form a spanning tree of D and every node different from r is the head of exactly one arc in A'. So for every node $s \in V$, A' contains a unique directed path from r to s. A set $C \subseteq A$ is called a **cut rooted at** r, or an r-**cut**, if $C = \delta^-(V')$ for some nonempty subset V' of $V \setminus \{r\}$. Note that an r-cut may contain another one as a subset, and sometimes we will consider (inclusionwise) minimal r-cuts.

The arborescences and minimal cuts rooted at r are related by the following so-called **blocking relation**. The r-arborescences are exactly the (inclusionwise) minimal sets intersecting all r-cuts, and conversely: the minimal r-cuts are exactly the minimal sets intersecting all r-arborescences. We shall elaborate on this relation in a general setting in Section 8.1.

Let a digraph $D = (V, A)$ with root $r \in V$ and a "length" $c_a > 0$, for every arc a, be given. The **shortest** r-**arborescence problem** is to find an arborescence A' rooted at r such that $c(A') := \sum_{a \in A'} c_a$ is as small as possible. Polynomial time algorithms for this problem have been designed by CHU and LIU (1965), EDMONDS (1967a) and others. To apply the ellipsoid method to this problem, we have to find a linear programming formulation of it.

To this end, we associate with every arc $a \in A$ a variable x_a. Let $\mathbb{R}^A$ be the vector space of mappings from A into $\mathbb{R}$. As usual, we view the elements of $\mathbb{R}^A$ as vectors whose components are indexed by the elements of A. For each subset $B \subseteq A$, we define its **incidence vector** $\chi^B = (\chi^B_a)_{a \in A} \in \mathbb{R}^A$ by

$$\chi^B_a = \begin{cases} 1 & \text{if } a \in B \\ 0 & \text{if } a \notin B. \end{cases}$$

The length function $c : A \to \mathbb{Q}_+$ is also an element of $\mathbb{R}^A$, and for every r-arborescence B, $c(B) = c^T \chi^B$.

First we formulate our problem as an integer linear program. So we are looking for a set of inequalities whose integral solutions are exactly the incidence

vectors of r-arborescences. The observation that each r-arborescence intersects each r-cut suggests the following integer linear programming formulation:

$$\begin{aligned} (7.2.1) \qquad \min\ & c^T x & & \\ x(\delta^-(W)) &\geq 1 & \text{for all} \quad & \emptyset \neq W \subseteq V \setminus \{r\}, \\ 0 \leq x_a &\leq 1 & \text{for all} \quad & a \in A, \\ x_a \text{ integer} & & \text{for all} \quad & a \in A. \end{aligned}$$

Since every r-arborescence intersects every r-cut, the incidence vector of any r-arborescence is a feasible solution for (7.2.1). Conversely, if x^* is an optimum solution for (7.2.1) then the set $\{a \in A \mid x_a^* = 1\}$ contains an r-arborescence, since it is a set intersecting all r-cuts. Hence $x^* \geq \chi^B$ for some r-arborescence B. In fact, x^* must be equal to χ^B since otherwise $c^T x^* > c^T \chi^B$ (as c is positive) contradicting the fact that x^* attains the minimum of (7.2.1). Therefore, the minimum of (7.2.1) is achieved by the incidence vector of an r-arborescence, and thus, (7.2.1) is equivalent to the shortest r-arborescence problem.

EDMONDS (1967a) showed that one may skip the integrality condition in (7.2.1) without changing the minimum. That is, (7.2.1) has the same optimum value as the following linear programming problem

$$\begin{aligned} (7.2.2) \qquad \min\ & c^T x & & \\ x(\delta^-(W)) &\geq 1 & \text{for all} \quad & \emptyset \neq W \subseteq V \setminus \{r\}, \\ 0 \leq x_a &\leq 1 & \text{for all} \quad & a \in A. \end{aligned}$$

In other words, Edmonds showed that the feasible region of (7.2.2) is exactly the polytope

$$\text{ARB}(D) := \text{conv}\{\chi^B \in \mathbb{R}^A \mid B \text{ contains an } r\text{-arborescence}\}.$$

So we could try to solve (7.2.2) (and hence (7.2.1)) with LP-techniques. However, just to write down the linear program (7.2.2) takes exponential time and space. Yet the ellipsoid method applies, since we can solve the strong separation problem for the polytopes ARB(D) in polynomial time.

To see this, let y be a vector in $\mathbb{Q}^A$. We first test whether $0 \leq y_a \leq 1$ for all $a \in A$. This clearly can be done in polynomial time, and if one of these inequalities is violated we have a hyperplane separating y from ARB(D). If y satisfies $0 \leq y_a \leq 1$ then we continue as follows. Consider y as a capacity function on the arcs of D. For every node $s \neq r$ we determine an (r,s)-cut C_s of minimum capacity. In Section 7.1 we saw that this can be done in polynomial time, e. g., with the Ford-Fulkerson algorithm or the ellipsoid method. Now it is easy to see that

$$\min\{y(C_s) \mid s \in V \setminus \{r\}\} = \min\{y(\delta^-(W)) \mid \emptyset \neq W \subseteq V \setminus \{r\}\}.$$

So by calculating $|V| - 1$ minimum capacity (r,s)-cuts, we can find a minimum capacity r-cut, say $\delta^-(W^*)$. If $y(\delta^-(W^*)) \geq 1$, all r-cut constraints of (7.2.2) are satisfied, and hence we conclude that y belongs to ARB(D). Otherwise, the

inequality $x(\delta^-(W^*)) \geq 1$ is violated by y and a separating hyperplane is found. As the vertices of ARB(D) are $\{0,1\}$-vectors, by Theorem (6.4.9) we know that the class of problems (7.2.2) can be solved in polynomial time, and since we can find an optimum vertex solution of (7.2.2) by Lemma (6.5.1) we thus can find the incidence vector of an optimum r-arborescence in polynomial time.

The above shows that from the polynomial time solvability of the minimum capacity r-cut problem we can derive the polynomial time solvability of the minimum length r-arborescence problem, using the ellipsoid method and polyhedral combinatorics. In Chapter 8 we shall see that this derivation can also be made the other way around: From the polynomial time solvability of the minimum length r-arborescence problem the polynomial time solvability of the minimum capacity r-cut problem follows. This is essentially due to the equivalence of the strong optimization problem and the strong separation problem for polyhedra – see Chapter 6. It is a special case of a more general duality relation between certain combinatorial optimization problems where the polynomial solvability of one class of combinatorial optimization problems follows from that of the class of dual problems, and conversely. This duality phenomenon is often based on the theory of blocking and antiblocking polyhedra – see Section 8.1 and Chapter 9. Other examples are the duality of the shortest (r,s)-path problem and the minimum capacity (r,s)-cut problem, of the shortest directed cut covering problem and the minimum capacitated directed cut problem, and of the weighted matching problem and the minimum capacitated odd cut problem. The last pair of problems is considered in the following section.

7.3 Matching

In the foregoing section we demonstrated one major approach to handle a combinatorial optimization problem. The problem was formulated as an integer linear program in which the integrality stipulations turned out superfluous. However, most combinatorial optimization problems have a "natural" integer linear programming formulation where skipping the integrality conditions does change the optimum value. This can always be repaired by adding some additional inequalities, called "cutting planes". But in many cases these extra constraints are difficult to describe explicitly. What is even worse for our purposes, it is usually hard to check whether a given point satisfies them. This prohibits the use of the ellipsoid method for deriving the polynomial time solvability of many combinatorial optimization problem. (We will make some further comments on this matter in Section 7.7.) In this section, however, we shall give an example where this approach does work.

Let $G = (V,E)$ be an undirected graph with an even number of nodes. An edge set $M \subseteq E$ is called a **perfect matching** if every node of G is contained in exactly one edge in M. The problem whether a given graph contains a perfect matching is a classical problem in graph theory. For the bipartite case it has been solved by FROBENIUS (1917).

(7.3.1) Frobenius' Marriage Theorem. *A bipartite graph G with bipartition A, B of the node set V has a perfect matching if and only if $|A| = |B|$ and each $X \subseteq A$ has at least $|X|$ neighbours in B.* □

It is not difficult to derive this result from the Max-Flow Min-Cut Theorem (7.1.4). The general case was characterized by TUTTE (1947).

(7.3.2) Tutte's Perfect Matching Theorem. *A graph G has a perfect matching if and only if for each $X \subseteq V$ the graph $G - X$ has at most $|X|$ components with an odd number of nodes.* □

In combinatorial optimization, besides the existence of perfect matchings one is interested in finding an optimum perfect matching.

Given, in addition to G, a weight function $c : E \to \mathbb{Q}$, the **minimum weighted perfect matching problem** is to find a perfect matching M in G of minimum weight $c(M)$. EDMONDS (1965b) designed a polynomial time algorithm for this problem. We show here that the polynomial time solvability also follows from the ellipsoid method.

The perfect matching problem is easily seen to be equivalent to the integer linear programming problem

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \\ x(\delta(v)) = 1 \quad & \text{for all} \quad v \in V, \\ x_e \geq 0 \quad & \text{for all} \quad e \in E, \\ x_e \in \mathbb{Z} \quad & \text{for all} \quad e \in E. \end{aligned} \tag{7.3.3}$$

Here $\delta(v)$ denotes the set of edges incident to v.

Note that the matrix of the equation system of (7.3.3) is the incidence matrix of G. We have mentioned in Section 7.1 that this matrix is totally unimodular if and only if G is bipartite, thus:

(7.3.4) *If G is bipartite, then we can drop the integrality conditions from* (7.3.3).

In general, however, we cannot simply drop the integrality constraints without changing the optimum value. E. g., if G is the graph shown in Figure 7.1 with weights as indicated, then (7.3.3) has optimum value 4 while its "LP-relaxation" (i. e., the program (7.3.3) without integrality stipulations) has optimum value 3.

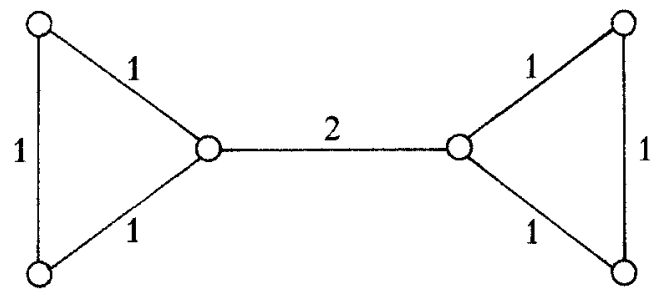


Figure 7.1

To apply LP-techniques to the solution of the perfect matching problem, we have to study the so-called **perfect matching polytope** PM(G) of G, which is the convex hull of the incidence vectors of perfect matchings in G, i. e.,

$$\mathrm{PM}(G) := \mathrm{conv}\{\chi^M \in \mathbb{R}^E \mid M \text{ perfect matching in } G\}.$$

So PM(G) is equal to the convex hull of the feasible solutions of (7.3.3). It is clear that $\min\{c^T x \mid x \in \mathrm{PM}(G)\}$ is equal to the minimum weight of a perfect matching in G. To write (7.3.3) as a linear program, we need to describe PM(G) by a system of linear inequalities. This was done by EDMONDS (1965b):

(7.3.5) Edmonds' Perfect Matching Polytope Theorem. *The perfect matching polytope* PM(G) *of a graph* $G = (V, E)$ *is determined by the following inequalities and equations:*

$$\begin{array}{lrll} \text{(i)} & x(\delta(v)) = 1 & \text{for all} & v \in V, \\ \text{(ii)} & x(\delta(W)) \geq 1 & \text{for all} & W \subseteq V \text{ with } |W| \text{ odd}, \\ \text{(iii)} & x_e \geq 0 & \text{for all} & e \in E. \end{array}$$

(The sets $\delta(W) = \{ij \in E \mid i \in W, j \in V \setminus W\}$ are called cuts, and the sets $\delta(W)$ with $|W|$ odd are called **odd cuts**; therefore, the inequalities (ii) are called **odd cut constraints**.)

Proof. Let P be the polytope defined by the constraints (i), (ii), and (iii). It is clear that the incidence vectors of the perfect matchings of G satisfy (i), (ii), and (iii), hence $\mathrm{PM}(G) \subseteq P$.

To show $P \subseteq \mathrm{PM}(G)$, suppose to the contrary that $G = (V, E)$ is a graph such that $P \not\subseteq \mathrm{PM}(G)$, and moreover, that $|V| + |E|$ is as small as possible. Then P must have a vertex x, say, that is not in PM(G).

Our assumption implies that $0 < x_e < 1$ for all $e \in E$, since otherwise we could delete from G the edge e, and, if $x_e = 1$, also the two endnodes of e and obtain a smaller counterexample. Moreover, G has no isolated nodes, as they could be removed; also G has no node of degree 1, since then the edge e containing this node would have $x_e = 1$ by (i). Finally, not all nodes of G have degree 2 as this would imply that G is the disjoint union of cycles for which the theorem obviously holds. It follows that $|E| > |V|$.

As x is a vertex, there are $|E|$ linearly independent constraints among (i), (ii) which are satisfied by x with equality. Hence there is a subset $W \subseteq V$ of odd size such that $x(\delta(W)) = 1$, $|W| \geq 3$, and $|V \setminus W| \geq 3$. Now we contract $V \setminus W$ to a single new node, say u, to obtain $G' = (W \cup \{u\}, E')$. Define $x' \in \mathbb{Q}^{E'}$ by $x'_e := x_e$ for all $e \in E(W)$, and $x'_{wu} := \sum_{v \in V \setminus W, wv \in E} x_{wv}$ for all $w \in W$. (We call x' the projection of x on G'.) It is easy to see that x' satisfies (i), (ii), and (iii) with respect to G'. As G is a smallest counterexample, we know that $x' \in \mathrm{PM}(G')$ and thus x' is a convex combination of incidence vectors of perfect matchings M' in G', say $x' = \sum_{M'} \lambda_{M'} \chi^{M'}$.

Similarly, by contracting W to a single new node, say t, we obtain a graph $G'' = ((V \setminus W) \cup \{t\}, E'')$ and a vector $x'' \in \mathbb{Q}^{E''}$, which by the same reasoning is

contained in PM(G''). Let $x'' = \sum_{M''} \mu_{M''} \chi^{M''}$, where M'' ranges over all perfect matchings in G''.

Now for each perfect matching M in G with $|M \cap \delta(W)| = 1$, let M' and M'' denote the associated perfect matchings in G' and G'' obtained by contraction. Then

$$x = \sum_{e \in \delta(W)} \sum_{\substack{M \text{ perfect matching} \\ \text{with } M \cap \delta(W) = \{e\}}} \frac{\lambda_{M'} \mu_{M''}}{x_e} \chi^M.$$

We leave it to the reader to show that this equality holds and that it represents x as a convex combination of incidence vectors of perfect matchings in G. This contradicts our assumption that x is not in PM(G). □

This theorem can be used to obtain a good characterization of those graphs having a perfect matching. Namely, a graph G has no perfect matching if and only if its perfect matching polyhedron PM(G) is empty. By the Farkas lemma (0.1.41) this is equivalent to the existence of real numbers ("multipliers") λ_v ($v \in V$) and $\mu_W \geq 0$ ($W \subseteq V$ and $|W|$ odd) such that for each edge $e \in E$

$$\sum_{\substack{v \in V \\ e \in \delta(v)}} \lambda_v + \sum_{\substack{W \subseteq V \\ e \in \delta(W),\, |W| \text{odd}}} \mu_W \leq 0, \tag{7.3.6}$$

but

$$\sum_{v \in V} \lambda_v + \sum_{W \subseteq V, |W| \text{ odd}} \mu_W > 0. \tag{7.3.7}$$

How does this condition relate to Tutte's, given in (7.3.2)? If Tutte's condition is violated then such multipliers are easily obtained as follows. Suppose there exists a **Tutte-set**, i. e., a set $X \subseteq V$ such that $G - X$ has odd components $W_1, \ldots, W_t$ with $t > |X|$. Then we set

$$\begin{aligned} \lambda_v &:= -1 \quad \text{if} \quad v \in X, \\ \mu_{W_i} &:= \ \ 1 \quad \text{for} \quad i = 1, \ldots, t, \end{aligned}$$

and set all other multipliers equal to zero. Then the conditions (7.3.6) and (7.3.7) are satisfied. Conversely, from any solution of (7.3.6) and (7.3.7) we can obtain a Tutte-set. However, the argument is quite complicated and we omit it.

Edmonds' perfect matching polyhedron theorem (7.3.5) immediately implies that the minimum perfect matching problem (and thus also the integer linear program (7.3.3)) is equivalent to the linear programming problem

$$\min c^T x \tag{7.3.8}$$

$$\begin{aligned} &\text{(i)} && x(\delta(v)) = 1 && \text{for all} && v \in V, \\ &\text{(ii)} && x(\delta(W)) \geq 1 && \text{for all} && W \subseteq V,\ |W| \text{ odd}, \\ &\text{(iii)} && x_e \geq 0 && \text{for all} && e \in E. \end{aligned}$$

So the odd cut constraints give us the cutting planes to be added to (7.3.3) to transform (7.3.3) to a linear program. As in the previous section, we cannot view this as an explicit linear program since there are exponentially many constraints. But again the ellipsoid method can be applied if we have a polynomial time algorithm to test the constraints. Such an algorithm indeed exists.

For a given point $y \in \mathbb{Q}^E$, we can check (i) and (iii) in polynomial time by substitution. So we may assume that y satisfies (i) and (iii). Then the problem to check (ii) can be formulated as follows. Consider the values y_e, $e \in E$, as capacities on the edges of G. The task is to decide whether there is an odd cut $\delta(W)$ of capacity $y(\delta(W))$ less than 1. This is clearly equivalent to finding a minimum capacity odd cut. A polynomial time algorithm to solve this problem was first designed by PADBERG and RAO (1982). It is convenient to describe an algorithm which solves a more general problem.

Let a graph $G = (V, E)$ be given and a set $\emptyset \neq T \subseteq V$ with $|T|$ even. A **T-cut** is a set of edges of the form $\delta(W)$ with $|T \cap W|$ odd. Thus odd cuts are just V-cuts (if $|V|$ is even, what we have assumed for the perfect matching problem). Now the more general problem, called **minimum weighted T-cut problem**, is: "Given a capacity function $y : E \to \mathbb{Q}_+$ on the edges of G, find a minimum capacity T-cut".

First we remark that one can find in polynomial time a minimum capacity cut $\delta(U)$ with $U \cap T \neq \emptyset$ and $T \setminus U \neq \emptyset$, i. e., a cut **separating** T. To see this, we transform G into a digraph D replacing each edge $ij \in E$ with capacity y_{ij} by two arcs (i,j) and (j,i) both with capacity y_{ij}. Then we run the algorithm to find a minimum (r,s)-cut in D for all pairs $r, s \in T$, $r \neq s$, described in Section 7.1. This way we get a minimum capacity cut in D separating T, which by construction yields a minimum capacity cut in G separating T, say $\delta(U)$.

If $|T \cap U|$ is odd, we are done. So let us suppose that $|T \cap U|$ is even.

(7.3.9) Lemma. *There exists a minimum capacity T-cut $\delta(W)$ in G with $U \subseteq W$ or $W \subseteq U$.*

Proof. Let $\delta(W')$ determine any minimum capacity T-cut. As $|W' \cap T| = |W' \cap T \cap U| + |W' \cap T \cap (V \setminus U)|$ is odd, we may assume that $|W' \cap T \cap U|$ is even (otherwise replace U by its complement and note that the statement of the lemma is invariant under taking the complement of U). Furthermore, since $|U \cap T| = |U \cap T \cap W'| + |U \cap T \cap (V \setminus W')|$ is nonzero we may assume that $|U \cap T \cap W'|$ is nonzero (otherwise replace W' by its complement). Note that

$$|(U \cup W') \cap T| = |U \cap T| + |W' \cap T| - |U \cap W' \cap T|$$

is odd. The following inequality is easy to verify:

$$y(\delta(U \cup W')) + y(\delta(U \cap W')) \leq y(\delta(U)) + y(\delta(W')). \tag{7.3.10}$$

But

$$y(\delta(U \cup W')) \geq y(\delta(W')),$$

since $\delta(W')$ is a minimum capacity T-cut, and

$$y(\delta(U \cap W')) \geq y(\delta(U)),$$

since $\delta(U)$ is a minimum capacity cut separating T. Hence we must have equality in (7.3.10) and the two subsequent inequalities. In particular, this shows that $W := U \cup W'$ determines a minimum capacity T-cut. □

This lemma shows that, if $|T \cap U|$ is even, then the problem of finding a minimum capacity T-cut in G can be reduced to two smaller problems. Namely, let G' and G'' be the graphs obtained from G by contracting U and $V \setminus U$, respectively, let $T' := T \setminus U$, $T'' := T \cap U$, and let y' and y'' be the corresponding projections of y on G' and G''. Find a minimum capacity T'-cut in G' and a minimum capacity T''-cut in G''. By Lemma (7.3.9) the smaller of these two cuts will correspond to a minimum capacity T-cut in G.

This defines a polynomial time algorithm inductively. In fact, let $p(\langle y \rangle, |V|)$ be a polynomial upper bound on the time needed to find a minimum capacity cut $\delta(U)$ separating T together with the time needed to carry out the contractions. We showed above that such a polynomial exists. Then it follows by induction on $|T|$ that the whole algorithm needs at most $(|T| - 1)p(\langle y \rangle, |V|)$ time.

This completes the description of the algorithm for finding a minimum capacity T-cut. It shows in particular that we can find a minimum capacity odd cut in polynomial time. Therefore, we can solve the separation problem for the perfect matching polyhedron in polynomial time. And hence, by the ellipsoid method, the weighted perfect matching problem can be solved in polynomial time.

Moreover, we can also decide in polynomial time whether a given graph has a perfect matching. If the answer is yes, we can actually find a perfect matching, i. e., a vertex of PM(G). If the answer is no, then the methods of Chapter 6 yield a solution of (7.3.6), (7.3.7). (One can use this solution to obtain a Tutte-set in polynomial time, but this procedure is rather involved.)

7.4 Edge Coloring

There are some combinatorial optimization problems that do not have a natural formulation as an optimization problem over nice polytopes. One class of examples consists of coloring problems. As an illustration, we consider the edge coloring problem for graphs.

Let $G = (V, E)$ be a graph. An **edge coloring** of G is a coloring of the edges of G so that no two adjacent edges have the same color. Equivalently, an edge coloring is a partition of the edge set into matchings (since clearly all edges with the same color form a matching). The **edge coloring number** (or **chromatic index**) $\gamma(G)$ is the minimum number of colors in any edge coloring of G.

It is again a classical problem of graph theory to determine $\gamma(G)$. HOLYER (1981) proved that this problem is $\mathcal{NP}$-complete. Clearly, the following inequality holds between the maximum degree $\Delta(G)$ and the chromatic index $\gamma(G)$ of any graph:

$$\Delta(G) \leq \gamma(G), \tag{7.4.1}$$

and the triangle K_3 shows that strict inequality may occur. On the other hand VIZING (1964) proved that for every simple graph G:

$$\gamma(G) \leq \Delta(G) + 1. \tag{7.4.2}$$

So, for a simple graph G, $\gamma(G)$ is either equal to $\Delta(G)$ or equal to $\Delta(G)+1$ (this makes Holyer's result surprising). KÖNIG (1916) showed that for bipartite graphs equality holds in (7.4.1).

(7.4.3) König's Edge Coloring Theorem. *If G is a bipartite graph then the maximum degree $\Delta(G)$ is equal to the edge coloring number $\gamma(G)$.* □

Since the maximum degree $\Delta(G)$ of a graph can be computed trivially, the edge coloring number $\gamma(G)$ of a bipartite graph G can be determined in polynomial time. König's original proof of (7.4.3) (using alternating path techniques) also provides a polynomial time algorithm to find an explicit optimum edge coloring.

To formulate the edge coloring problem for bipartite graphs as a linear program over a polyhedron is not immediate. In order to explain the difficulty let us first formulate the **maximum degree problem** as a linear program (even though this problem is trivial). The problem is to find a maximum cardinality star in a graph $G = (V, E)$. (A **star** is an edge set in which all edges have a common endnode.) It is easy to see that for a bipartite graph the incidence vectors of stars are just the integral solutions of the following system of linear inequalities:

$$\begin{aligned} \sum_{e \in M} x_e &\leq 1 \quad \text{for all matchings } M \subseteq E. \\ x_e &\geq 0 \quad \text{for all } e \in E, \end{aligned}$$

So the maximum degree in a bipartite graph G is the optimum value of the integral solutions of the following linear program

$$\begin{aligned} &\max \mathbb{1}^T x \\ &\sum_{e \in M} x_e \leq 1 \quad \text{for all matchings } M \subseteq E, \\ &x_e \geq 0 \quad \text{for all } e \in E. \end{aligned} \tag{7.4.4}$$

The dual of (7.4.4) is

$$\begin{aligned} &\min y^T \mathbb{1} \\ &\sum_{M \ni e} y_M \geq 1 \quad \text{for all } e \in E, \\ &y_M \geq 0 \quad \text{for all matchings } M \subseteq E. \end{aligned} \tag{7.4.5}$$

Note that every edge coloring of G yields an integral solution of (7.4.5). So if we denote the common optimum value of (7.4.4) and (7.4.5) by μ, then we have

$$\Delta(G) \leq \mu \leq \gamma(G).$$

König's edge coloring theorem (7.4.3) implies that for bipartite graphs we have equality above. Equivalently, for bipartite graphs, (7.4.4) and (7.4.5) both have optimum solutions which are integral. By a similar argument we can show that for bipartite graphs, the feasible region of (7.4.4) has only integral vertices. This is not true for the feasible region of (7.4.5). Consider the bipartite graph G of Figure 7.2. The four matchings $\{a, c, e\}$, $\{a, d, f\}$, $\{b, c, f\}$, $\{b, d, e\}$ of G taken with coefficient $\frac{1}{2}$ and all other matchings of G with coefficient 0 form a vertex of the feasible region of (7.4.5) for G.

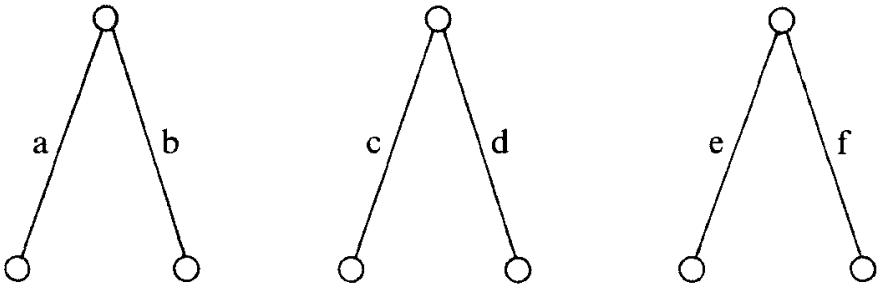


Figure 7.2

Let us see what the general results obtained by the ellipsoid method give us in this situation.

Suppose G is a bipartite graph. First of all, we can maximize any linear objective function $c^T x$ over the feasible region of (7.4.4) by inspecting the sets $\delta(v)$, $v \in V(G)$. Hence, by the general methods developed in Chapter 6 we can find a basic optimum solution of the dual problem (7.4.5). Unfortunately, this basic optimum solution need not be integral. Even though we do know that there is a basic integer optimum solution for (7.4.5), we do not know a general method to find it.

But in this special case (and some other cases that we will describe later) we can find additional properties of the linear programs which enable us to construct an optimum integral solution from an optimum solution. Here is one such trick:

Let y be any optimum solution of (7.4.5), and let M be a matching of G with $y_M > 0$. Then by complementary slackness – see (0.1.50) – M intersects each maximum size star, as each such star gives an optimum solution for the primal problem (7.4.4). So $\Delta(G - M) = \Delta(G) - 1$. Deleting M and repeating this procedure we obtain a collection of $\Delta(G)$ matchings partitioning the edges of G, i. e., we obtain in polynomial time a minimum edge coloring of the bipartite graph G.

7.5 Matroids

Matroids are combinatorial abstractions of linear dependence in vector spaces and of certain aspects of graphs. They appear to become more and more central objects in combinatorics, especially in combinatorial optimization and combinatorial geometry. For an introduction to the subject we recommend

WELSH (1976), and for applications, see BIXBY (1982), IRI (1983), and RECSKI (1988). Here we discuss only some polyhedral and optimization aspects of matroids.

A **matroid** is a pair $(E, \mathcal{I})$ where E is a finite set and $\mathcal{I}$ is a set of subsets of E satisfying

(7.5.1) $\emptyset \in \mathcal{I}$,

(7.5.2) *If* $T \in \mathcal{I}$ *and* $S \subseteq T$ *then* $S \in \mathcal{I}$,

(7.5.3) *If* $S, T \in \mathcal{I}$ *and* $|T| > |S|$ *then there exists an element* $t \in T \setminus S$ *such that* $S \cup \{t\} \in \mathcal{I}$.

The elements of $\mathcal{I}$ are called the **independent sets** of the matroid. Three standard examples of matroids are the following.

(7.5.4) Linear Matroids. E is a finite set of vectors in a linear space and $\mathcal{I}$ is the collection of linearly independent subsets of E. In this case axiom (7.5.3) is equivalent to the Steinitz exchange property. □

(7.5.5) Graphic Matroids. E is the set of edges of an undirected graph G and $\mathcal{I}$ is the collection of all forests in G. □

(7.5.6) Transversal Matroids. Let $G = (V, E)$ be a bipartite graph with bipartition $\{U, W\}$. Let $\mathcal{I}$ be the collection of subsets of U covered by some matching in G. Then $(U, \mathcal{I})$ is a matroid.

A simple special case is obtained when all degrees in U are 1. Then the components of the bipartite graph are stars which induce a partition $\{U_1, \ldots, U_k\}$ of U, and $U' \subseteq U$ is independent if and only if $|U' \cap U_i| \leq 1$ for $i = 1, \ldots, k$. These matroids are called **partition matroids**. □

If $(E, \mathcal{I})$ is a matroid and $F \subseteq E$, then all maximal independent subsets of F have the same cardinality, by (7.5.3). These maximal independent subsets of F are called the **bases** of F, and their cardinality is called the **rank** of F, denoted by $r(F)$. The rank function has the following properties.

(7.5.7) $r(\emptyset) = 0$,

(7.5.8) $S \subseteq T$ *implies* $r(S) \leq r(T)$,

(7.5.9) $r(S) \leq |S|$ *for all* $S \subseteq E$,

(7.5.10) $r(S \cup T) + r(S \cap T) \leq r(S) + r(T)$ *for all* $S, T \subseteq E$.

Note that $S \in \mathcal{I}$ if and only if $r(S) = |S|$, so a matroid is uniquely determined by its rank function. Property (7.5.8) is called **monotonicity**, property (7.5.9) **subcardinality**, and property (7.5.10) **submodularity**. One can show that every integer valued function $r : 2^E \to \mathbb{Z}$ that satisfies (7.5.7) – (7.5.10) is the rank function of a (unique) matroid. (An example of a submodular function that

does not satisfy (7.5.8) and (7.5.9) is the function $f : 2^V \to \mathbb{R}$ that associates with every node set $W \subseteq V$ of a graph $G = (V, E)$ with edge capacities c_e, $e \in E$, the capacity $f(W) := c(\delta(W))$ of its cut.) Generalizing the well-known techniques of deleting and contracting edges of a graph we define **deletion** and **contraction** of an element $e \in E$ in a matroid $(E, \mathcal{I})$ as follows. Deletion of e results in the matroid $(E \setminus \{e\}, \mathcal{I} \cap 2^{E \setminus \{e\}})$. Contraction of e results in the matroid $(E \setminus \{e\}, \{S \setminus \{e\} \mid e \in S \in \mathcal{I}\})$ if $\{e\} \in \mathcal{I}$, and in the matroid $(E \setminus \{e\}, \mathcal{I})$ if $\{e\} \notin \mathcal{I}$. Deletion leaves the rank function on subsets of $E \setminus \{e\}$ unchanged, while contraction of e yields the following rank function:

$$r'(F) := r(F \cup \{e\}) - r(\{e\}), \quad F \subseteq E \setminus \{e\}.$$

These operations are very helpful in inductive proofs.

Matroids are important for combinatorial optimization, since a maximum weighted independent set can be found by a very elementary algorithm, called the greedy algorithm. Let $(E, \mathcal{I})$ be a matroid and $c : E \to \mathbb{Q}_+$ be a weight function. The problem is to find an independent set of maximum weight. The following method works.

(7.5.11) Greedy Algorithm.

(i) *Choose an element $e_1 \in E$ such that $\{e_1\} \in \mathcal{I}$ and $c(e_1)$ is maximal.*
(ii) *Assuming that $e_1, \ldots, e_i$ are chosen, choose e_{i+1} in $E \setminus \{e_1, \ldots, e_i\}$ such that $\{e_1, \ldots, e_{i+1}\} \in \mathcal{I}$ and $c(e_{i+1})$ is maximal.*
(iii) *Repeat (ii) until no such e_{i+1} exists.*

□

(7.5.12) Theorem. *The greedy algorithm ends up with a set $\{e_1, \ldots, e_r\} \in \mathcal{I}$ of maximum weight.*

Proof. By axiom (7.5.3) it is clear that the greedy algorithm produces a basis of E. So $r = r(E)$. Suppose that the greedy algorithm does not work, i. e., that there is a set $\{f_1, \ldots, f_t\} \in \mathcal{I}$ with larger weight than the greedy solution. As $r = r(E)$ we have $t \leq r$. Without loss of generality let $c(f_1) \geq \ldots \geq c(f_t)$. As $c(\{f_1, \ldots, f_t\}) > c(\{e_1, \ldots, e_r\})$ there is an index j, $1 \leq j \leq t$ with $c(f_j) > c(e_j)$. By axiom (7.5.3) applied to $\{f_1, \ldots, f_j\}$ and $\{e_1, \ldots, e_{j-1}\}$ there is an index i with $1 \leq i \leq j$ such that $f_i \notin \{e_1, \ldots, e_{j-1}\}$ and $\{e_1, \ldots, e_{j-1}, f_i\} \in \mathcal{I}$. But $c(f_i) \geq c(f_j) > c(e_j)$, which contradicts the choice of e_j. □

It can be shown that matroids are exactly those structures $(E, \mathcal{I})$ satisfying (7.5.1) and (7.5.2) for which the greedy algorithm gives a maximum independent set in $\mathcal{I}$ for every weight function $c : E \to \mathbb{Q}_+$.

To discuss the running time of the greedy algorithm we should first understand how a matroid is given. Clearly, if it is given by listing all its independent sets, it is trivial to find an optimum independent set by scanning through this list. This trivial algorithm is polynomial (even linear) in the encoding length. In our examples (7.5.4), (7.5.5), (7.5.6), however, the matroids are given in a

much more compact form, viz. by a matrix, a graph, and a bipartite graph, respectively. Then enumerating all independent sets would of course require running time exponential in the encoding length of this compact input. The greedy algorithm, however, works in polynomial time, given that any set can be tested for independence in polynomial time.

This is indeed the case in our examples. For linear matroids (7.5.4) we need to check whether a set of vectors is linearly independent, and this can be done by Gaussian elimination. For graphic matroids (7.5.5) we can decide by depth-first search whether a set contains no cycle (i. e., is a forest), and for transversal matroids (7.5.6) we can test whether a subset of U is covered by some matching with the help of the maximum flow algorithm of Section 7.1 or the matching algorithm of Section 7.3.

One might think that to describe a matroid by listing all its independent sets is an uneconomical way to do. But in general one cannot do much better. It is known that there are at least $2^{2^{n/2}}$ matroids on n elements, and so for any encoding there would be n-element matroids with code length at least $2^{n/2}$.

So, to get nontrivial algorithmic results about matroids we define the encoding length of a matroid as the number of the elements of E. In the running time of a matroid algorithm we count an independence testing as one step. That is, we assume that the matroid is given by some oracle that can be asked if a given set is independent or not. In this model the greedy algorithm is oracle-polynomial.

As in the previous sections we can formulate the matroid optimization problem in terms of polyhedra. For any matroid $M = (E, \mathcal{I})$ let $\mathrm{IND}(M)$ be the convex hull of the incidence vectors of independent sets of M, i. e.,

$$\mathrm{IND}(M) := \mathrm{conv}\{\chi^I \in \mathbb{R}^E \mid I \in \mathcal{I}\}.$$

$\mathrm{IND}(M)$ is called the **matroid polytope** of M. Then the matroid optimization problem is equivalent to finding $\max\{c^T x \mid x \in \mathrm{IND}(M)\}$. Of course, we know how to solve this (by the greedy algorithm), but yet it will be interesting to formulate this problem as a linear programming problem. The following inequalities, for vectors $x \in \mathbb{R}^E$, are trivially satisfied by the incidence vectors of the independent sets of M:

$$x_e \geq 0 \quad \text{for all} \quad e \in E, \tag{7.5.13}$$

$$x(F) \leq r(F) \quad \text{for all} \quad F \subseteq E. \tag{7.5.14}$$

Edmonds (1970, 1971) proved that these inequalities suffice:

(7.5.15) Theorem. *For every matroid $M = (E, \mathcal{I})$, $\mathrm{IND}(M)$ is the set of all vectors $x \in \mathbb{R}^E$ satisfying* (7.5.13) *and* (7.5.14).

Proof. Without loss of generality we may assume that each singleton $\{e\}$ is independent in M. Then $\mathrm{IND}(M)$ contains all unit vectors and the zero vector – hence $\mathrm{IND}(M)$ has full dimension $|E|$.

Let $a^T x \leq b$ be an inequality defining a facet of $\mathrm{IND}(M)$. We want to show that $a^T x \leq b$ is a positive multiple of one of the inqualities (7.5.13) or (7.5.14).

Suppose first that $a_e < 0$ for some $e \in E$. If there is a set $I \in \mathcal{I}$ with $e \in I$ and $a^T\chi^I = b$, then $J := I \setminus \{e\} \in \mathcal{I}$ and $a^T\chi^J > b$ which contradicts the validity of $a^Tx \leq b$. Hence $x_e = 0$ holds for all points of the facet $\mathrm{IND}(M) \cap \{x \in \mathbb{R}^E \mid a^Tx = b\}$. Thus, this is the same facet as $\mathrm{IND}(M) \cap \{x \mid x_e = 0\}$, which implies that $a^Tx \leq b$ is a positive multiple of $-x_e \leq 0$.

Suppose now that $a_e \geq 0$ for all $e \in E$, and set $F := \{e \in E \mid a_e > 0\}$. We claim that $a^Tx \leq b$ is a positive multiple of $x(F) \leq r(F)$. To prove this we show that if $I \in \mathcal{I}$ and $a^T\chi^I = b$ then $\chi^I(F) = r(F)$, i. e., $I \cap F$ is a basis of F. For, suppose there is an element $f \in F \setminus I$ such that $I' := (I \cap F) \cup \{f\} \in \mathcal{I}$. Then $a^T\chi^{I'} = a^T\chi^I + a_f > a^T\chi^I = b$, contradicting that $a^Tx \leq b$ is a valid inequality for $\mathrm{IND}(M)$. □

Now that we have a description of $\mathrm{IND}(M)$ by linear inequalities we could try to replace the greedy algorithm by the ellipsoid method. But there is no obvious way known to check whether a vector $y \in \mathbb{Q}^E$ satisfies the inequalities (7.5.13) and (7.5.14).

We can, however, use the ellipsoid method the other way around: since we can solve the strong optimization problem for $\mathrm{IND}(M)$ in polynomial time by the greedy algorithm, we can also solve the strong separation problem for $\mathrm{IND}(M)$ in polynomial time – see Theorem (6.4.9). That is, we can test in polynomial time whether a given nonnegative vector $y \in \mathbb{Q}^E$ satisfies $y(F) \leq r(F)$ for all $F \subseteq E$. This algorithm for testing membership in $\mathrm{IND}(M)$, resulting from the ellipsoid method, is far from being practical – it shows just the polynomial solvability of this problem. CUNNINGHAM (1984) designed a direct combinatorial algorithm for the separation problem for matroid polytopes based on augmenting path techniques.

The above can be extended considerably by considering *intersections* of matroids. Many combinatorial structures can be viewed as the common independent sets (or common bases) of two matroids. For instance, the set of r-arborescences in a digraph $D = (V, A)$ is the set of common bases of the graphic matroid of the underlying undirected graph of $(V, A \setminus \delta^-(r))$ and the partition matroid coming from the partition $\{\delta^-(v) \mid v \in V \setminus \{r\}\}$ of $A \setminus \delta^-(r)$. Furthermore, matchings in a bipartite graph are the common independent sets of two partition matroids.

If $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ are two matroids on the same ground set E, their **intersection** is the pair $M_1 \cap M_2 := (E, \mathcal{I}_1 \cap \mathcal{I}_2)$. In general this intersection is not a matroid; nevertheless, as was detected by Edmonds, two nice properties are maintained: the polynomial time solvability of the optimization problem over $\mathcal{I}_1 \cap \mathcal{I}_2$ and the linear description of the associated polytope. Let us consider the polytope associated with the intersection of two matroids, defined as follows:

$$\mathrm{IND}(M_1 \cap M_2) := \mathrm{conv}\{\chi^I \mid I \in \mathcal{I}_1 \cap \mathcal{I}_2\}.$$

EDMONDS (1970) proved the following theorem.

(7.5.16) Theorem. *Let M_1, M_2 be two matroids on the same ground set E. Then*

$$\mathrm{IND}(M_1 \cap M_2) = \mathrm{IND}(M_1) \cap \mathrm{IND}(M_2),$$

i. e., $\mathrm{IND}(M_1 \cap M_2)$ *is determined by the inequality system*

$$\begin{array}{lll} \text{(i)} & x_e \geq 0 & \text{for all } e \in E, \\ \text{(ii)} & x(F) \leq \min\{r_1(F), r_2(F)\} & \text{for all } F \subseteq E, \end{array}$$

where r_i *is the rank function of* M_i, $i = 1, 2$.

Proof. Clearly $\mathrm{IND}(M_1 \cap M_2) \subseteq \mathrm{IND}(M_1) \cap \mathrm{IND}(M_2)$. To show the reverse inclusion, we use induction on $|E|$. For $E = \emptyset$, the theorem is trivial. So let $E \neq \emptyset$ and let y be a vertex of the polytope $\mathrm{IND}(M_1) \cap \mathrm{IND}(M_2)$. We prove that y is integral, which clearly implies that $y \in \mathrm{IND}(M_1 \cap M_2)$.

We first show that y has at least one integral component. We may assume $y > 0$. Let $\mathcal{T}_i := \{S \subseteq E \mid y(S) = r_i(S)\}$, $i = 1, 2$. Then note that $\mathcal{T}_1$ and $\mathcal{T}_2$ are closed under union and intersection, since, if $S, T \in \mathcal{T}_i$, then by (7.5.10):

$$\begin{aligned} r_i(S) + r_i(T) &\geq r_i(S \cap T) + r_i(S \cup T) \geq y(S \cap T) + y(S \cup T) \\ &= y(S) + y(T) = r_i(S) + r_i(T) \end{aligned}$$

and so we must have equality throughout. We define two equivalence relations $\sim_i$ $(i = 1, 2)$ on E as follows:

$e \sim_i f$ if either $e = f$, or there is no subset $S \subseteq E$ with $S \in \mathcal{T}_i$ and $|S \cap \{e, f\}| = 1$.

Let $A_1, \ldots, A_k$ be the equivalence classes of $\sim_1$ and let $B_1, \ldots, B_l$ be the equivalence classes of $\sim_2$. If $S \in \mathcal{T}_1$, then S is the union of classes A_i. Similarly, if $S \in \mathcal{T}_2$, then S is the union of classes B_j.

Since y is a vertex of the polytope defined by the inequalities (i) and (ii) having no 0-component, it is the unique solution of equations of the form $x(F) = \min\{r_1(F), r_2(F)\}$. By our definitions, y is therefore the unique solution of the equations

$$\begin{aligned} x(S) &= r_1(S), \quad S \in \mathcal{T}_1, \\ x(S) &= r_2(S), \quad S \in \mathcal{T}_2. \end{aligned}$$

It follows that the normal vectors of these equations, i. e., the incidence vectors χ^S, $S \in \mathcal{T}_1 \cup \mathcal{T}_2$, span $\mathbb{R}^E$; and since χ^S, $S \in \mathcal{T}_1 \cup \mathcal{T}_2$, is the sum of vectors χ^{A_j} or χ^{B_j} by the remark above, the vectors $\{\chi^{A_i} \mid i = 1, \ldots, k\} \cup \{\chi^{B_j} \mid j = 1, \ldots, l\}$ also span $\mathbb{R}^E$. This implies that $k + l \geq |E|$, and in fact, strict inequality must hold as $\chi^{A_1} + \ldots + \chi^{A_k} = \chi^{B_1} + \ldots + \chi^{B_l}$.

Without loss of generality we may assume that $k > \frac{1}{2}|E|$. Then one of the classes A_i must be a singleton, say $A_1 = \{f\}$. Let

$$\begin{aligned} U &:= \cup\{S \in \mathcal{T}_1 \mid f \notin S\} \quad \text{and} \\ V &:= \cap\{S \in \mathcal{T}_1 \mid f \in S\}. \end{aligned}$$

Since $\mathcal{T}_1$ is closed under union and intersection, U and V are also in $\mathcal{T}_1$. Moreover, $V \setminus U = \{f\}$. Namely, $e \in V \setminus U$ implies that for every set $S \in \mathcal{T}_1$, $e \in S$ if and only if $f \in S$, and this means, $e \sim_1 f$. But since $A_1 = \{f\}$ there is no

no element $e \neq f$ with $e \sim_1 f$. Therefore, we get $U \cup \{f\} = U \cup V$, and since U and V are in $\mathcal{T}_1$, $U \cup \{f\} \in \mathcal{T}_1$. Hence

$$y_f = y(U \cup V) - y(U) = r_1(U \cup V) - r_1(U);$$

and thus y_f is an integer. Therefore, y has at least one integral component, say y_f, which clearly is either 0 or 1.

If $y_f = 0$ then delete f from M_1 and M_2 to obtain M_1' and M_2'. The vector y' obtained from y by projecting y onto $\mathbb{R}^{E \setminus \{f\}}$ is a vertex of $\mathrm{IND}(M_1') \cap \mathrm{IND}(M_2')$. So, by the induction hypothesis, y' is integral. Thus y is also integral.

If $y_f = 1$ we conclude similarly by contracting f in the matroids M_1 and M_2. □

From this theorem one can derive the following important formula.

(7.5.17) Corollary. *Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two matroids, with rank functions r_1 and r_2, respectively. Then*

$$\max\{|I| \mid I \in \mathcal{I}_1 \cap \mathcal{I}_2\} = \min\{r_1(S) + r_2(E \setminus S) \mid S \subseteq E\}.$$

□

Polynomial time algorithms to find a maximum weight common independent set of two matroids were given by EDMONDS (1979), FRANK (1981a), and LAWLER (1975). These algorithms are much more involved than the greedy algorithm. Here we show that polynomial solvability of this problem also follows from the ellipsoid method and the greedy algorithm.

In fact, we mentioned before that the strong separation problem for matroid polytopes is solvable in polynomial time using the greedy algorithm and the ellipsoid method. Therefore, we can solve the strong separation problem for the intersection of two matroid polytopes. So by the ellipsoid method again we can solve the optimization problem for the intersection of two matroid polytopes in polynomial time – see Exercise (6.5.18). As we have shown in Theorem (7.5.16), the convex hull of the incidence vectors of the common independent sets of two matroids is equal to the intersection of two matroid polytopes, we can find a maximum weight common independent set of two matroids in polynomial time (as we can find an optimum vertex of $\mathrm{IND}(M_1 \cap M_2)$).

By the same procedure, we can optimize over the intersection of three or more matroid polytopes. Alas, the vertices of such an intersection are generally not integral, i. e., there is no extension of Theorem (7.5.16) to three or more matroids. Actually, the problem of finding a maximum cardinality common independent set of three matroids (even of three matroids as simple as partition matroids) is $\mathcal{NP}$-complete. One such example is the asymmetric traveling salesman problem which can be represented as the intersection of two partition matroids and a graphic matroid.

We now give some examples of matroid polyhedra. First, consider the graphic matroid (7.5.5) of a graph $G = (V, E)$. By Theorem (7.5.15), the convex hull of the incidence vectors of all forests of G, called the **forest polytope of** G,

is the polytope defined by the nonnegativity constraints (7.5.13) and the rank inequalities (7.5.14). In the graphic matroid on E, the rank of a set $F \subseteq E$ is the maximum number of edges of a forest contained in F, i. e.,

$$r(F) = |V| - \text{ number of components of } (V,F).$$

We show that the system

$$\begin{aligned} x_e &\geq 0 && \text{for all} \quad e \in E, \\ x((E(W)) &\leq |W| - 1 && \text{for all} \quad \emptyset \neq W \subseteq V \end{aligned} \tag{7.5.18}$$

is a complete linear description of the forest polytope. First of all, each of the inequalities (7.5.18) is implied by the system given in Theorem (7.5.15), as $r(E(W)) \leq |W| - 1$. Conversely, given a rank inequality $x(F) \leq r(F)$, let $(V_1,F_1), \ldots, (V_k,F_k)$ be the components of (V,F). Then

$$\begin{aligned} x(F) &= x(F_1) + \ldots + x(F_k) \leq x(E(V_1)) + \ldots + x(E(V_k)) \\ &\leq (|V_1| - 1) + \ldots + (|V_k| - 1) = |V| - k = r(F) \end{aligned}$$

holds for each vector x satisfying (7.5.18). So $x(F) \leq r(F)$ is implied by (7.5.18). Since we can optimize over (7.5.18) in polynomial time with the greedy algorithm we can solve the strong separation problem for (7.5.18) in polynomial time. (A combinatorial algorithm for this problem based on flow techniques has been designed by PADBERG and WOLSEY (1984).)

Second, let $D = (V,A)$ be a digraph and, for every node $v \in V$, let a nonnegative integer b_v be given. Call a set $B \subseteq A$ independent if $|B \cap \delta^-(v)| \leq b_v$ for all $v \in V$. These independent sets form a partition matroid on A – cf. (7.5.6). It is easy to see that for a complete description of the convex hull of all incidence vectors of this partition matroid on A, the following inequalities suffice

$$\begin{aligned} 0 \leq x_a &\leq 1 && \text{for all} \quad a \in A, \\ x(\delta^-(v)) &\leq b_v && \text{for all} \quad v \in V. \end{aligned} \tag{7.5.19}$$

So the separation problem for the polytope associated with the partition matroid defined above and a given vector y can be solved by substituting y into the inequalities (7.5.19).

We shall now intersect these two matroid polyhedra. Let $D = (V,A)$ be a digraph and let $r \in V$ be a vertex, called the root. We may assume that r is not entered by any arc. Let $b_v = 1$ for all $v \in V$. Consider the forest matroid on A (induced by the underlying graph of D) and the partition matroid defined by the vector $b \in \mathbb{R}^V$. Theorem (7.5.16) then implies that the convex hull of the incidence vectors of the sets independent in both matroids is given by

$$\begin{aligned} &\text{(i)} && x_a \geq 0 && \text{for all} \quad a \in A, \\ &\text{(ii)} && x(\delta^-(v)) \leq 1 && \text{for all} \quad v \in V \setminus \{r\}, \\ &\text{(iii)} && x(A(W)) \leq |W| - 1 && \text{for all} \quad \emptyset \neq W \subseteq V. \end{aligned} \tag{7.5.20}$$

The vertices of this polytope are the incidence vectors of all branchings in D. To obtain the convex hull of the r-arborescences we may simply set the constraints (ii) of (7.5.20) to equality; namely, by this we obtain a face of the polytope defined by (7.5.20) whose vertices are integral (since the polytope is integral) and which are incidence vectors of branchings in which every node except r is entered by exactly one arc. These arc sets are the r-arborescences in D. Now we observe that the inequalities $x(A(W)) \leq |W| - 1$ with $r \in W$ are superfluous, namely in this case $x(A(W)) \leq \sum_{v \in W, v \neq r} x(\delta^-(v)) = |W| - 1$. Moreover, if $\emptyset \neq W \subseteq V \setminus \{r\}$ then we can transform inequality (iii) as follows:

$$x(A(W)) \leq |W| - 1 \iff x(A(W)) - \sum_{v \in W} x(\delta^-(v)) \leq -1 \iff x(\delta^-(W)) \geq 1.$$

Therefore, we can conclude that the convex hull of the incidence vectors of all r-arborescences in D is given by

$$\begin{aligned} x_a &\geq 0 \quad \text{for all} \quad a \in A, \\ x(\delta^-(v)) &= 1 \quad \text{for all} \quad v \in V \setminus \{r\}, \\ x(\delta^-(W)) &\geq 1 \quad \text{for all} \quad \emptyset \neq W \subseteq V \setminus \{r\}. \end{aligned} \tag{7.5.21}$$

Again, we can optimize over this polytope in polynomial time, and we can solve the strong separation problem for (7.5.21) in polynomial time.

We leave it to the reader to explore the relation of the polytope (7.5.21) to the polytope (7.2.2), in particular to derive from the integrality of (7.5.21) the integrality of (7.2.2).

7.6 Subset Sums

Most of the polynomial time combinatorial optimization algorithms we have treated in this chapter were obtained using just the ellipsoid method. In Chapter 6, the main role of basis reduction was in handling non-full-dimensional polyhedra. Typically, the polyhedra occurring in combinatorial optimization are, however, either full-dimensional or can easily be transformed into such polyhedra. In fact, in GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981) a number of combinatorial applications were derived using rounding procedures not more sophisticated than continued fractions. (To obtain strongly polynomial algorithms, however, the use of basis reduction seems much more essential.) In this section we illustrate that basis reduction can sometimes be used directly in designing polynomial time algorithms in combinatorial optimization. This application is due to LAGARIAS and ODLYZKO (1983).

Let us start with a story. Assume that a small company sends out bills for amounts $a_1, a_2, \ldots, a_n$, respectively. After a month, they get a report from their bank that a total amount of b has been transferred to their account. Since this is less than the total outstanding amount, they want to write letters to those

costumers who have not paid. But the bank does not tell *which* of the amounts a_i have arrived, and the company tries to figure this out by trying to find a subset of the numbers a_i that adds up to b.

Of course, there may be several solutions (the most trivial way in that this can happen is when two of the a_i's are equal), and then finding this subset does not definitely indicate whom the company should urge. But if the numbers a_i are large compared with n and sufficiently random then – as we shall see – it is unlikely that two subsets of them would sum up to exactly the same amount.

Mathematically, the problem is the following:

(7.6.1) Subset Sum Problem. *Given n positive integers $a_1, \ldots, a_n$, and a positive integer b, find a subset $\{i_1, \ldots, i_k\}$ of $\{1, \ldots, n\}$ such that*

$$a_{i_1} + \ldots + a_{i_k} = b.$$

Another way to put this is that we are looking for a nontrivial $\{0,1\}$-solution to the following equation:

$$a_1 x_1 + \ldots + a_n x_n - by = 0. \tag{7.6.2}$$

Problem (7.6.1) is known to be $\mathcal{NP}$-complete – see GAREY and JOHNSON (1979). This seems to be of little help to the company. But the company can try to make use of some special features of the problem. First of all, we know:

(7.6.3) *Equation* (7.6.2) *has a nonzero $\{0,1\}$-solution.*

To locate the customers who have paid it is essential to be sure that (7.6.2) has a unique nontrivial $\{0,1\}$-solution. In fact, if the a_i are large and random the subsequent Lemma (7.6.5) shows that the company may even expect the following stronger condition to hold:

(7.6.4) *Equation* (7.6.2) *has a unique nonzero $\{0,1\}$-solution, and every nonzero integral solution of* (7.6.2) *that is not a multiple of this unique solution has at least one component whose absolute value is at least 2^n.*

(7.6.5) Lemma. *Let $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ be nonempty and let N be a positive integer. Suppose that the integers a_i $(i = 1, \ldots, n)$ are chosen independently and uniformly from $\{1, \ldots, N\}$ and let $b := a_{i_1} + \ldots + a_{i_k}$. Then* (7.6.4) *is satisfied with probability larger than $1 - 2^{(n+2)n}/N$.*

Proof. By the choice of b, (7.6.2) has a nonzero $\{0,1\}$-solution $(x_1, x_2, \ldots, x_n, y)^T$, say. Suppose (7.6.2) has a solution violating (7.6.4), i. e., there is a nonzero integral vector $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{y})^T$ satisfying (7.6.2) and $|\tilde{x}_i| < 2^n$ $(i = 1, \ldots, n)$, $|\tilde{y}| < 2^n$ that is not a multiple of $(x_1, \ldots, x_n, y)^T$. Then

$$\bar{x}_i := \begin{cases} \tilde{x}_i - \tilde{y} & \text{if } i \in \{i_1, \ldots, i_k\}, \\ \tilde{x}_i & \text{otherwise} \end{cases}$$

gives a nonzero vector $(\bar{x}_1, \ldots, \bar{x}_n)^T$ satisfying

$$a_1\bar{x}_1 + \ldots + a_n\bar{x}_n = 0 \tag{7.6.6}$$

and

$$-2^{n+1} < \bar{x}_i < 2^{n+1}. \tag{7.6.7}$$

Consider now any fixed nonzero integral vector $\bar{x} := (\bar{x}_1, \ldots, \bar{x}_n)^T$. Since the a_i are chosen uniformly and independently from $\{1, \ldots, N\}$, trivially the probability that the a_i satisfy (7.6.6) is not larger than $1/N$. Since the number of integral vectors satisfying (7.6.7) is smaller than $2^{(n+2)n}$, it follows that the probability that (7.6.6) has a solution satisfying (7.6.7) is smaller than $2^{(n+2)n}/N$. This implies the statement of the lemma. □

The significance of assumption (7.6.4) is not only that it is satisfied with high probability (if N is large enough) but also that, if it is satisfied, the subset sum problem can be solved in polynomial time, using basis reduction:

(7.6.8) Lemma. *There exists a polynomial time algorithm that, for given positive integers $a_1, \ldots, a_n$, and b, either*

(i) *supplies a nonzero integral solution of* (7.6.2) *where all components have absolute value less than 2^n, or*

(ii) *proves that the subset sum problem has no solution.*

In particular, if (7.6.4) *is satisfied the algorithm solves the subset sum problem.*

Proof. For $n = 1$ the problem is trivial. So assume $n \geq 2$. First, consider the integral vectors $(x_1, \ldots, x_n, y)^T$ that satisfy equation (7.6.2). These form a lattice L. Using the polynomial time algorithm of Corollary (5.4.10) we can construct a basis of L. Next we run the algorithm of Theorem (5.3.16) to find a reduced basis $(b_1, \ldots, b_n, b_{n+1})$ of the lattice L.

We consider 2 cases.

Case 1: $\|b_1\|_\infty < 2^n$. So we can conclude with (i).

Case 2: $\|b_1\|_\infty \geq 2^n$. By Theorem (5.3.13) (b), $\|b_1\| \leq 2^{n/2} \min\{\|b\| \mid b \in L, b \neq 0\}$. Hence $\min\{\|b\| \mid b \in L, b \neq 0\} \geq 2^{n/2} > \sqrt{n+1}$. This shows that L contains no nonzero $\{0,1\}$-vector, and thus we can conclude with (ii).

In particular, if (7.6.4) is satisfied, the vector b_1 yields a solution for the subset sum problem. □

Combining these two lemmas, we immediately have the following theorem due to LAGARIAS and ODLYZKO (1983).

(7.6.9) Theorem. *There exists a polynomial time algorithm that finds a solution of the subset sum problem* (7.6.1) *with probability at least $1-2^{(n+2)n}/N$, if the given integers $a_1, \ldots, a_n$ are chosen independently and uniformly from $\{1, \ldots, N\}$ and if the given integer b is chosen such that the subset sum problem has a solution.* □

The condition (7.6.4), which guarantees the success of the algorithm of Lemma (7.6.8), seems, even in the case of the introductory example, rather strong and somewhat artificial. But exactly this condition is satisfied if one tries to break so-called knapsack-based cryptosystems with high redundancy rate. The methods sketched in this section imply that most such cryptosystems are insecure in some sense.

For instance, the basis reduction method was employed by ADLEMAN (1983) to break the single iteration Graham-Shamir scheme, by ADLEMAN (1983) and LAGARIAS (1984) to attack doubly iterated Merkle-Hellman schemes, by LAGARIAS and ODLYZKO (1983) to cryptoanalyse Merkle-Hellman und Graham-Shamir schemes with low information rate and by BRICKELL (1985) to show the insecurity of the iterated Merkle-Hellman, iterated Graham-Shamir schemes and Shamir's ultimate knapsack scheme.

7.7 Concluding Remarks

In this chapter we have discussed several problems that illustrate some of the standard techniques that are used to apply the ellipsoid method and diophantine approximation to combinatorial optimization problems. The framework of this approach can be described as follows. Most combinatorial optimization problems can be transformed into the following form:

Given a finite set S of vectors in $\mathbb{R}^n$ and a linear objective function c, find a vector $x \in S$ maximizing $c^T x$ over S.

There is of course a trivial finite algorithm to solve this problem: Scan all elements of S. To do better we have to assume that S is structured, i. e., that S is encoded in much less space than the encoding length of S. For instance, S may be the set of incidence vectors of r-arborescences, r-cuts, perfect matchings, odd cuts etc. in a digraph or graph; so in these cases S is given by specifying this digraph or graph (plus an expression (name) of bounded length like "perfect matchings in", "odd cuts in" ...).

Our starting point for investigating these problems from a polyhedral point of view and for applying the ellipsoid method is the observation that

$$\max\{c^T x \mid x \in S\} = \max\{c^T x \mid x \in \text{conv}(S)\}.$$

The set $\text{conv}(S)$ is a polytope, and hence it can be described by a finite system of linear inequalities

$$a_i^T x \leq b_i,\ i = 1, \ldots, m. \tag{7.7.1}$$

So our combinatorial optimization problem is equivalent to the linear program

$$\begin{aligned} &\max\ c^T x \\ &a_i^T x \leq b_i,\ i = 1, \ldots, m. \end{aligned} \tag{7.7.2}$$

It may happen in some very fortunate cases that such inequalities can be easily found and listed as in the case of perfect matchings of bipartite graphs – cf. (7.3.4).

In other cases, m is exponentially large but the system still has a nice characterization in the sense that there is a polynomial time algorithm to test whether a given point satisfies all inequalities, and if not, to find a violated inequality. Then the ellipsoid method can be applied to solve the combinatorial optimization problem in polynomial time. In this approach the combinatorial difficulties are concentrated on finding the set of inequalities describing conv(S). Historically, such systems of inequalities were frequently derived as by-products of polynomial time algorithms. As we saw before, often there are fairly short direct proofs of such polyhedral characterizations. The ellipsoid method shows that the implication can be turned around and that a polyhedral characterization can directly give the polynomial solvability of a problem.

Sometimes, dual solutions of (7.7.2) are of combinatorial interest, e. g., they may give a Tutte-set as in Section 7.3. We have also seen an example where certain integral dual solutions are of primary interest, namely the edge colorings of bipartite graphs in Section 7.4. If the primal problem can be solved in polynomial time, the general methods of Chapter 6 enable us to find basic optimum dual solutions in polynomial time. However, these basic dual solutions may not be integral, even if integral optimal dual solutions exist. So they may not correspond to the desired combinatorial structures (like Tutte-sets or edge colorings). No general method is known to obtain integral optimum dual solutions, but in many cases special properties of the problem may be used to construct such solutions from any optimal dual solution.

In most cases the original linear program (7.7.2) is also beyond control in various respects:

- *There is (usually) no (nontrivial) characterization of a linear system known describing* conv(S).
- *The problem of deciding whether an inequality belongs to such a system is often at least as hard as the original optimization problem.*
- *Sometimes, even if such a system of inequalities is known, deciding whether a point satisfies all these inequalities is not the least bit easier than the original problem.*

But even in such cases the methodology described before might be quite helpful – even for practical purposes, a matter we were not concerned too much about yet. The idea behind this approach is the following.

Quite frequently it is possible to determine some large classes of inequalities that define facets of conv(S) and for which one can check in polynomial time whether a given point satisfies them or not. Thus, using the ellipsoid method, one can optimize over these inequalities in polynomial time – theoretically.

It is known that the ellipsoid method is not the most efficient algorithm in practice. So one simply replaces the ellipsoid method by the simplex method (or whatever linear programming algorithm one prefers) and proceeds in the same manner by iteratively adding cutting planes. That is, one chooses some classes of valid inequalities for conv(S), all facet defining if possible, for which the separation problem is solvable in polynomial time. If no polynomial time

separation algorithms exist (or if they are too time consuming), well-designed and fast separation heuristics may also do. Then one chooses an initial linear programming relaxation, finds an optimum vertex solution, and checks whether the optimum solution is in S. If not, one searches the chosen classes for inequalities violated by the present optimum, adds them to the linear program and continues until a point is found that optimizes the given objective function over the inequalities considered. If the point is in S the problem is solved. If it is not in S, usually a very good bound on the optimum value of the original problem is obtained.

There are a number of hard combinatorial optimization problems where the known classes of facet defining inequalities are rather good approximations of conv(S), so that in practical computations optimum solutions over these inequalities often turn out to be in S. Even if the solutions obtained this way are not in conv(S) they may be very helpful in a branch-and-bound framework – see GRÖTSCHEL (1982), PADBERG and GRÖTSCHEL (1985). These methods are not polynomial, but they work reasonably well in practice.

Inspired by this practical experience one may even go back to the starting point and use these ideas to solve problems known to be in $\mathcal{P}$ with nonpolynomial cutting plane techniques. This has for instance been done in GRÖTSCHEL and HOLLAND (1985) for the perfect matching problem. The simplex method is used to solve the linear program (7.3.3) (without integrality conditions). If the solution is integral, one terminates. Otherwise separation heuristics and, if these fail, the separation algorithm of PADBERG and RAO (1982) for this problem – cf. Section 7.3 – is used to find violated odd cut constraints. These are added to the present LP, and the simplex method is called again, etc. This method is guaranteed to terminate in finite (but not in polynomial) time. Practical computational experience has shown that for large problems, say at least 300 nodes, this cutting plane method is compatible with the best known combinatorial algorithms. Whether for other combinatorial optimization problems such cutting plane algorithms are also competitive with the best combinatorial methods remains to be explored.

The significance of the ellipsoid method in combinatorics is not in suggesting alternative techniques to efficient algorithms. Rather, one can *prove the existence* of a polynomial time algorithm for many problems by the techniques described above quite easily. This points out those problems for which the search for practically efficient (combinatorial and not ellipsoidal) polynomial time methods should start. We shall give a number of examples in subsequent chapters where the only polynomial time algorithms known to date use the ellipsoid method (minimization of submodular functions, finding maximum stable sets and minimum colorings of perfect graphs). Here a field of research is opened, namely, finding "really good" combinatorial algorithms for these problems.

One way to view the techniques described in this chapter is that the ellipsoid method reduces one combinatorial problem to another one. Examples, already mentioned, are the reduction of the minimum r-arborescence problem to the minimum r-cut problem, the reduction of the minimum perfect matching problem to the minimum odd cut problem, etc. The ellipsoid method can again be applied to reduce the minimum r-cut problem to the minimum r-arborescence problem, and the minimum odd cut problem to the minimum perfect matching problem,

etc. It is thus a matter of taste for which of these problems one wants to design a polynomial time algorithm. The ellipsoid method automatically yields a polynomial time algorithm for the other.

To go further, one might think of using the ellipsoid method as a unifying procedure that – supplemented with trivial transformations – reduces any polynomially solvable combinatorial optimization problem to a "basic problem" for which a very simple polynomial time algorithm is known. As we shall see in Chapter 10, one such "almost basic problem" is the polymatroid optimization problem, which can be easily solved with the greedy algorithm. We shall show that quite a large number of combinatorial optimization problems can be reduced to this problem using the ellipsoid method and simple transformations.

*Chapter 8

Combinatorial Optimization: A Tour d'Horizon

In Chapter 7 we have introduced several basic combinatorial optimization problems, and we have shown in detail how the ellipsoid method and basis reduction together with polyhedral information about these problems can be used to design polynomial time algorithms. In this chapter we give an overview about combinatorial optimization problems that are solvable in polynomial time. We also survey important theorems that provide polyhedral descriptions of the polytopes associated with these problems. We indicate how these results can be employed to derive polynomial time algorithms based on the ellipsoid method and basis reduction. The results of this chapter are presented in a condensed form, to cover as much material as possible.

For most of the problems considered, the strongly polynomial time solvability of the problems follows directly from Theorem (6.6.5). This holds for all "primal" problems in particular. Among the "dual" problems we treat, there are a few exceptions. Sometimes the standard procedure does not work but there are ad hoc methods with which strong polynomiality can be derived. For instance, the algorithm given for the arborescence packing problem (8.4.11) is not strongly polynomial, but using a method of MADER (1983) it can be made strongly polynomial. In a few cases our polynomial time procedures do not translate into strongly polynomial ones, e. g., the algorithm given for the halfintegral multicommodity cut packing problem (8.6.4) for certain types of graphs does not seem to be modifyable into a strongly polynomial one. If we allow the operation of rounding a number to the nearest integer among the elementary arithmetic operations, all these algorithms can, however, be made strongly polynomial.

*8.1 Blocking Hypergraphs and Polyhedra

Many combinatorial optimization problems can be nicely treated in the following framework, due to FULKERSON (1968, 1971) and LEHMAN (1965).

A **hypergraph** H is a finite set of finite sets. The elements of $\cup H$ are called the **nodes** of H, and the elements of H are called the **edges** of H. A **clutter** is a hypergraph in which no edge contains another.

The **blocker** $\mathrm{BL}(H)$ of a hypergraph H is the set of all inclusion-minimal sets intersecting all edges of H. So $\mathrm{BL}(H)$ is a clutter. For example, given a strongly connected digraph $D = (V, A)$ and $r \in V$, the set H of all r-cuts of D forms a hypergraph with node set $A \setminus \delta^-(r)$; its blocker $\mathrm{BL}(H)$ is the set of all r-arborescences of D – see Section 7.2.

The following result is elementary. If H is a clutter, then

$$\text{BL}(\text{BL}(H)) = H. \tag{8.1.1}$$

We shall be interested in finding an edge with minimum weight in a hypergraph with nonnegative weights on its nodes. To formulate this problem polyhedrally, we consider the **dominant of the incidence vectors of the edges of** H, denoted by $\text{dmt}(H)$, i. e.:

$$\text{dmt}(H) = \text{conv}\{\chi^E \in \mathbb{R}^{\cup H} \mid E \in H\} + \mathbb{R}_+^{\cup H}. \tag{8.1.2}$$

We can use the blocker $\text{BL}(H)$ of H to obtain valid inequalities for the dominant of H:

$$\begin{aligned} &\text{(a)} \quad x_v \geq 0 \quad \text{for all} \quad v \in \cup H, \\ &\text{(b)} \quad x(F) \geq 1 \quad \text{for all} \quad F \in \text{BL}(H), \end{aligned} \tag{8.1.3}$$

where $x(F) := \sum_{e \in F} x_e$. A hypergraph H has the **max-flow min-cut property** (or **$\mathbb{Q}_+$-max-flow min-cut property**, or **$\mathbb{Q}_+$-MFMC property**) if (8.1.3) is sufficient to describe $\text{dmt}(H)$. Clearly this is equivalent to saying that the polyhedron described by (8.1.3) has integral vertices only, i. e., the system (8.1.3) is TPI. To digest the name, the reader may work out that the hypergraph of (r, s)-cuts in a digraph has the max-flow min-cut property. This definition is due to LEHMAN (1965). The significance of it is that H has the max-flow min-cut property if and only if for every weight function $w : \cup H \to \mathbb{Z}_+$,

$$\begin{aligned} \min\{w(E) \mid E \in H\} = \max\{ &\sum_{F \in \text{BL}(H)} \lambda_F \mid \lambda_F \geq 0 \text{ for all } F \in \text{BL}(H), \\ &\sum_{F \ni v} \lambda_F \leq w_v \text{ for all } v \in \cup H\}. \end{aligned} \tag{8.1.4}$$

This is an immediate consequence of the duality theorem of linear programming. Lehman proved the following interesting fact.

(8.1.5) Theorem. *A hypergraph has the max-flow min-cut property if and only if its blocker has.*

Proof. Recall the definition of the blocker $\text{bl}(S)$ of a set $S \subseteq \mathbb{R}_+^n$ given in Section 0.1, and observe that the max-flow min-cut property is equivalent to the equation

$$\text{dmt}(H) = \text{bl}(\text{dmt}(\text{BL}(H))). \tag{8.1.6}$$

Using (8.1.1) and applying the operator bl to (8.1.6), we get

$$\begin{aligned} \text{bl}(\text{dmt}(\text{BL}(\text{BL}(H)))) = \text{bl}(\text{dmt}(H)) &= \text{bl}(\text{bl}(\text{dmt}(\text{BL}(H)))) \\ &= \text{dmt}(\text{BL}(H)). \end{aligned}$$

□

It will be interesting to see in the examples that follow how this simple result implies the equivalence of rather different min-max results.

The right hand side of equation (8.1.4) is of additional combinatorial interest if there is an integral optimal system of the λ's. In this case, the maximum can be interpreted as a maximum packing of edges of the blocker. If this happens for each $w : UH \to \mathbb{Z}_+$, the hypergraph $BL(H)$ is said to have the **$\mathbb{Z}_+$-max-flow min-cut property ($\mathbb{Z}_+$-MFMC property)**. In other words, $BL(H)$ has the $\mathbb{Z}_+$-MFMC property if the system (8.1.3) is TDI. Lehman's theorem (8.1.5), however, does not extend to the $\mathbb{Z}_+$-max-flow min-cut property – consider, e. g., the hypergraph of triangles in K_4.

Even if it does not have the max-flow min-cut property, the blocker of a hypergraph H provides an integer programming formulation of the problem of finding a minimum weight member of a hypergraph, namely, for every weight function $w : UH \to \mathbb{Q}_+$,

$$\begin{aligned}(8.1.7)\quad \min\{w(E) \mid E \in H\} = \min\{w^T x \mid x(F) &\geq 1 \text{ for all } F \in BL(H),\\ x_e &\geq 0 \text{ for all } e \in UH, x \text{ integral}\}.\end{aligned}$$

Then we may consider the LP-relaxation, its dual, and the associated integer linear program, which are related as follows:

$$\begin{aligned}(8.1.8)\quad &\min\{w(E) \mid E \in H\} =\\ &= \min\{w^T x \mid x(F) \geq 1 \text{ for all } F \in BL(H),\ x \geq 0,\ x \text{ integral}\}\\ &\geq \min\{w^T x \mid x(F) \geq 1 \text{ for all } F \in BL(H),\ x \geq 0\}\\ &= \max\{\textstyle\sum_{F\in BL(H)} \lambda_F \mid \sum_{F\in BL(H)} \lambda_F \chi^F \leq w,\ \lambda_F \geq 0 \text{ for all } F \in BL(H)\}\\ &\geq \max\{\textstyle\sum_{F\in BL(H)} \lambda_F \mid \sum_{F\in BL(H)} \lambda_F \chi^F \leq w,\ \lambda_F \geq 0 \text{ for all } F \in BL(H),\\ &\qquad \lambda_F \text{ integral for all } F \in BL(H)\}.\end{aligned}$$

Similarly for the blocker $BL(H)$ of H we have:

$$\begin{aligned}(8.1.9)\quad &\min\{w(F) \mid F \in BL(H)\} =\\ &= \min\{w^T x \mid x(E) \geq 1 \text{ for all } E \in H,\ x \geq 0,\ x \text{ integral}\}\\ &\geq \min\{w^T x \mid x(E) \geq 1 \text{ for all } E \in H,\ x \geq 0\}\end{aligned}$$

$$= \max\{\textstyle\sum_{E\in H} \lambda_E \mid \sum_{E\in H} \lambda_E \chi^E \leq w,$$
$$\lambda_E \geq 0 \text{ for all } E \in H\}$$

$$\geq \max\{\textstyle\sum_{E\in H} \lambda_E \mid \sum_{E\in H} \lambda_E \chi^E \leq w,$$
$$\lambda_E \geq 0 \text{ for all } E \in H,$$
$$\lambda_E \text{ integral for all } E \in H\}.$$

The first and last programs in (8.1.8) and (8.1.9) have combinatorial contents. The first lines express the problem of finding a minimum weight edge of a clutter or, equivalently, finding a minimum weight node cover for its blocker. The last lines are problems of packing edges of a clutter. The two middle lines may be viewed as fractional versions of these problems.

We know by Lehman's theorem that, if the first inequality in (8.1.8) holds with equality for all $w \geq 0$, then so does the first inequality in (8.1.9), and vice versa. We also know from Theorem (0.1.53) that, if the second inequality in (8.1.8) holds with equality for all $w \geq 0$ then so does the first inequality in (8.1.8). So in this case, BL(H) has the $\mathbb{Z}_+$-MFMC property. Similar results hold with H and BL(H) interchanged.

The results described above can be exploited algorithmically in the following way. Whenever H has the max-flow min-cut property, equality holds in the first inequality of (8.1.8), and thus, we have a linear programming formulation of the problem of finding a minimum weight edge of a hypergraph. By the results of Chapter 6, this linear program can be solved in polynomial time if and only if the separation problem for the inequality system (8.1.3) can be solved in polynomial time. Given $y \in \mathbb{R}^{\cup H}$, the nonnegativity constraints here are easy to check. The separation problem for the blocking constraints (b) of (8.1.3) is equivalent to finding an edge $F^* \in \text{BL}(H)$ of minimum weight $y(F^*)$. If $y(F^*) < 1$, then $x(F^*) \geq 1$ is a violated inequality, otherwise y satisfies (8.1.3). Summing up these considerations and using Theorems (6.4.9) and (6.5.14) we obtain the following result.

(8.1.10) Theorem. *There exist oracle-polynomial time algorithms that, for any hypergraph H with the $\mathbb{Q}_+$-MFMC property given by its underlying set $V = \bigcup H$ and by an oracle that, for every weight function $c : V \to \mathbb{Q}_+$, returns a minimum weight edge of H, solve the following problems:*

(a) *given any weight function $w : V \to \mathbb{Q}_+$, find a minimum weight edge of* BL(H)*;*

(b) *given any weight function $w : V \to \mathbb{Q}_+$, find edges $E_1, \ldots, E_t$ of H and positive rationals $\lambda_1, \ldots, \lambda_t$ such that $\lambda_1\chi^{E_1} + \ldots + \lambda_t\chi^{E_t} \leq w$ and such that $\lambda_1 + \ldots + \lambda_t$ is as large as possible;*

(c) *the problem in* (b) *with H replaced by* BL(H). □

If a hypergraph H has the $\mathbb{Z}_+$-max-flow min-cut property then (8.1.4) provides a linear programming formulation of the problem of finding a maximum w-packing of the edges of H. However, though the existence of an integral optimum solution of this LP is guaranteed, the ellipsoid method may fail to find

it, and we do not know any general procedure to calculate an integral optimum solution from a fractional one that works for all hypergraphs with the $\mathbb{Z}_+$-max-flow min-cut property. Nevertheless, in all cases we know of, special techniques like "uncrossing" (to be described later in this chapter – see, e. g., (8.4.5)) yield such integral optimum solutions.

As described above for hypergraphs with the max-flow min-cut property, there is an oracle-polynomial time algorithm to find a minimum weight edge of a hypergraph H if and only if there is such an algorithm for the same problem for the blocker of H. For all hypergraphs with the max-flow min-cut property we know, the minimum weight edge problem is in fact solvable in oracle-polynomial time. An outstanding open problem is the following: Is there an oracle-polynomial time algorithm that, for any hypergraph H (given by an oracle telling whether a given set belongs to H or not) with the max-flow min-cut property and for any $w : UH \to \mathbb{Q}_+$, finds a minimum weight edge?

In Section 0.1 we have not only introduced blocking polyhedra but also antiblocking polyhedra. Analogously, one can define the antiblocker of a hypergraph. This notion gives rise, at least partially, to similar results as above, but is best treated in the framework of perfect graphs, and is therefore postponed to Section 9.2.

*8.2 Problems on Bipartite Graphs

In this section we discuss a number of optimization problems defined on bipartite graphs that will turn out to be basic to many other combinatorial optimization problems. They are basic in the sense that a wide range of combinatorial optimization problems and almost all combinatorial min-max relations contain a bipartite graph problem as a – generally nontrivial – special case. For instance, the matching problem for bipartite graphs can be considered as a special case of the max-flow problem, of the matching problem for general graphs, of the stable set problem for perfect graphs, and of the matroid intersection problem.

First let us recall some definitions and notation. Let $G = (V, E)$ be a graph. A **matching** is a collection of pairwise disjoint edges. The **matching number** of G, denoted by $\nu(G)$, is the maximum size of a matching. A **stable set** (**clique**, respectively) is a subset S of the node set V such that each two nodes in S are nonadjacent (adjacent, respectively). The **stability number** resp. the **clique number** is the maximum cardinality of a stable set resp. clique in G, and is denoted by $\alpha(G)$ resp. $\omega(G)$. A **node covering** (**edge covering**, respectively) is a subset $W \subseteq V$ ($F \subseteq E$, respectively) such that every edge contains at least one node in W (every node is contained in at least one edge in F, respectively). The minimum cardinality of a node covering resp. edge covering is called the **node covering number** resp. the **edge covering number**, and is denoted by $\tau(G)$ resp. $\rho(G)$. A partition of V into stable sets (cliques, respectively) is called a **coloring** (**clique covering**, respectively) of G. The **coloring number** resp. the **clique covering number** is the smallest number of stable sets in a coloring resp. cliques in a clique covering of G, and is denoted by $\chi(G)$ resp. $\bar{\chi}(G)$. An **edge coloring** is a partition of E

into matchings. The **edge coloring number** is the smallest number of matchings in an edge coloring of G, and is denoted by $\gamma(G)$. The **maximum degree** in G is denoted by $\Delta(G)$, and the **minimum degree** by $\delta(G)$.

We will see that for a bipartite graph G, various relations exist between the $\nu(G)$, $\tau(G)$, $\alpha(G)$, $\omega(G)$, $\rho(G)$, $\chi(G)$, $\bar{\chi}(G)$, $\gamma(G)$, $\Delta(G)$, $\delta(G)$, and that each of these numbers can be computed in polynomial time (the last two trivially, of course).

First of all, for each graph $G = (V, E)$ without isolated nodes, one has the following obvious relations:

$$\begin{aligned} \nu(G) &\leq \tau(G), \\ \alpha(G) &\leq \rho(G). \end{aligned} \tag{8.2.1}$$

In addition, we have the following identities – see GALLAI (1959):

$$\begin{aligned} \alpha(G) + \tau(G) &= |V|, \\ \nu(G) + \rho(G) &= |V|, \end{aligned} \tag{8.2.2}$$

where the last equation only holds if G has no isolated nodes.

The triangle K_3 shows that strict inequalities can occur in (8.2.1). By Gallai's identities (8.2.2), equality in one of the inequalities (8.2.1) implies equality in the other (provided G has no isolated nodes).

König was one of the first to study optimization problems for graphs. The following theorem was presented in KÖNIG (1931), but finds its roots in earlier papers by FROBENIUS (1912, 1917) – see Theorem (7.3.1) – and KÖNIG (1915, 1916) (for an account of the history of these and related results, and for proofs see LOVÁSZ and PLUMMER (1986)).

(8.2.3) König's Matching Theorem. *If G is a bipartite graph, then the matching number $\nu(G)$ is equal to the node covering number $\tau(G)$.* □

This theorem combined with Gallai's identities gives another result of KÖNIG (1933).

(8.2.4) König's Edge Covering Theorem. *If G is a bipartite graph without isolated nodes, then the edge covering number $\rho(G)$ is equal to the stability number $\alpha(G)$.* □

These two theorems of König are among the first in a row of useful min-max relations. König's matching theorem (8.2.3) gives a good characterization of both the matching problem and the node covering problem for bipartite graphs. That is, these problems belong to $\mathcal{NP} \cap$ co-$\mathcal{NP}$. A similar conclusion follows from König's edge covering theorem (8.2.4). This is typical for most of the min-max relations to be dealt with in this chapter: they all yield good characterizations for the corresponding optimization problems.

In fact, each of the optimization problems corresponding to the König's theorems is in $\mathcal{P}$. Polynomial algorithms to find explicitly a matching of maximum

size and a node covering of minimum size in a bipartite graph have been designed by FORD and FULKERSON (1957) and HALL (1956) (running time $O(|V|^3)$), and by HOPCROFT and KARP (1973) ($O(|V|^{5/2})$). In fact, the running time of the Hopcroft-Karp algorithm is $O(|E||V|^{1/2})$, as was shown by GALIL (1983). Using the fact that also Gallai's identities are polynomially constructive, one easily transforms these algorithms to solving the stable set and edge covering problems for bipartite graphs.

Since cliques in bipartite graphs have size one or two, König's edge covering theorem, equivalently states that, for bipartite graphs G, the stability number $\alpha(G)$ is equal to the clique covering number $\bar{\chi}(G)$. Similarly, the clique covering problem for bipartite graphs is polynomially solvable.

These methods and results can be extended to the weighted case. If $G = (V, E)$ is a graph and $w : E \to \mathbb{Q}$ is a weight function on the edges then we denote the maximum weight $w(M) := \sum_{e \in M} w_e$ of a matching M in G by $\nu_w(G)$. If $w : V \to \mathbb{Q}$ is a weight function on the nodes, then we denote the maximum weight $w(S) := \sum_{v \in S} w_v$ of a stable set S in G by $\alpha_w(G)$.

In general, if w is some weight function and $\varphi(G)$ is some graph parameter (like $\tau(G)$, $\omega(G)$ etc. as defined above) we denote the weighted version of this parameter by $\varphi_w(G)$ (like $\tau_w(G)$, $\omega_w(G)$ etc.), or – if the indices become too clumsy – we write $\varphi(G, w)$ etc..

König's theorems have the following weighted versions.

(8.2.5) Theorem. *For any bipartite graph $G = (V, E)$ and any weight function $w : E \to \mathbb{Z}_+$ on the edges, the maximum weight $\nu_w(G)$ of a matching is equal to the minimum value of $\sum_{v \in V} y_v$, where, for each node $v \in V$, y_v is a nonnegative integer such that $y_u + y_v \geq w_{uv}$ for each edge $uv \in E$.* □

(8.2.6) Theorem. *For any bipartite graph $G = (V, E)$ and any weight function $w : V \to \mathbb{Z}_+$ on the nodes, the minimum weight $\tau_w(G)$ of a node covering is equal to the maximum value of $\sum_{e \in E} x_e$, where, for each edge $e \in E$, x_e is a nonnegative integer such that $x(\delta(v)) \leq w_v$ for each node $v \in V$.* □

(8.2.7) Theorem. *For any bipartite graph $G = (V, E)$ without isolated nodes and any weight function $w : E \to \mathbb{Z}_+$ on the edges, the minimum weight $\rho_w(G)$ of an edge covering is equal to the maximum value of $\sum_{v \in V} y_v$, where, for each node $v \in V$, y_v is a nonnegative integer such that $y_u + y_v \leq w_{uv}$ for each edge $uv \in E$.* □

(8.2.8) Theorem. *For any bipartite graph $G = (V, E)$ without isolated nodes and any weight function $w : V \to \mathbb{Z}_+$ on the nodes, the maximum weight $\alpha_w(G)$ of a stable set is equal to the minimum value of $\sum_{e \in E} x_e$, where, for each edge $e \in E$, x_e is a nonnegative integer such that $x(\delta(v)) \geq w_v$ for each node $v \in V$.* □

These results contain König's theorems by taking all weights equal to one. Result (8.2.5) was first stated by EGERVÁRY (1931). The assertions easily follow

from the total unimodularity of the node-edge incidence matrix M of the bipartite graph G – see Section 7.1. Indeed, the four theorems above correspond to the following four linear programming duality equations:

$$
\begin{aligned}
(8.2.9)\qquad & \max\{w^T x \mid x \geq 0, Mx \leq \mathbb{1}\} = \min\{\mathbb{1}^T y \mid y \geq 0, y^T M \geq w^T\},\\
& \min\{y^T w \mid y \geq 0, y^T M \geq \mathbb{1}^T\} = \max\{\mathbb{1}^T x \mid x \geq 0, Mx \leq w\},\\
& \min\{w^T x \mid x \geq 0, Mx \geq \mathbb{1}\} = \max\{\mathbb{1}^T y \mid y \geq 0, y^T M \leq w^T\},\\
& \max\{y^T w \mid y \geq 0, y^T M \leq \mathbb{1}^T\} = \min\{\mathbb{1}^T x \mid x \geq 0, Mx \geq w\},
\end{aligned}
$$

where in the last two cases the underlying bipartite graph G is assumed to have no isolated nodes.

The total unimodularity of M yields that each of these linear programming problems has integer optimum solutions, which implies the results (8.2.5) up to (8.2.8). (Alternatively, results (8.2.6) and (8.2.8) can be derived directly from König's theorems (8.2.3) and (8.2.4) by splitting each node v into w_v mutually nonadjacent new nodes.)

So each of the numbers $\alpha_w(G)$, $\tau_w(G)$, $\nu_w(G)$, $\rho_w(G)$ can be defined as the optimum value of a linear program whose constraints can be trivially constructed from the given bipartite graph. Hence it follows that they can be computed in polynomial time using the ellipsoid method. Note that this is a direct application of Khachiyan's original version of the ellipsoid method. By finding vertex solutions we can also determine maximum stable sets, minimum node coverings, maximum matchings, and minimum edge coverings in polynomial time.

Result (8.2.8) implies that $\alpha_w(G) = \bar{\chi}_w(G)$ for all bipartite graphs. It is trivial to see that also the (complementary) relation $\omega_w(G) = \chi_w(G)$ holds for bipartite graphs (EGERVÁRY (1931)).

There are many closely related optimization problems which are important in practical applications and which can be handled along similar lines, e. g.:

(8.2.10) Optimal Assignment Problem. *Given an integral $n \times n$-matrix (w_{ij}), find a permutation σ of $\{1, \ldots, n\}$ such that $\sum_{i=1}^{n} w_{i,\sigma(i)}$ is as large as possible.*

(8.2.11) The (Hitchcock-Koopmans) Transportation Problem. *Given an integral $m \times n$-cost matrix (c_{ij}), an integral "supply" vector $b = (b_1, \ldots, b_m)^T$, and an integral "demand" vector $d = (d_1, \ldots, d_n)^T$, find nonnegative integers x_{ij} $(i = 1, \ldots, m; j = 1, \ldots, n)$, such that $\sum_{j=1}^{n} x_{ij} \leq b_i$ $(i = 1, \ldots, m)$ and $\sum_{i=1}^{m} x_{ij} \geq d_j$ $(j = 1, \ldots, n)$, and such that $\sum_{i,j} c_{ij} x_{ij}$ is as small as possible.*

(8.2.12) The Capacitated (Hitchcock-Koopmans) Transportation Problem. *This is problem* (8.2.11) *where in addition integral upper bounds are given for the x_{ij}.*

Again, good characterizations may be derived from the total unimodularity of the corresponding matrices. A common generalization of many of these problems is the following.

(8.2.13) *Given a bipartite graph $G = (V, E)$ and functions $b_1, b_2 : V \to \mathbb{Z}$, $c_1, c_2, w : E \to \mathbb{Z}$, find a vector $x \in \mathbb{Z}^E$ satisfying $c_1(e) \leq x(e) \leq c_2(e)$ for each*

edge e and $b_1(v) \leq x(\delta(v)) \leq b_2(v)$ for each node v, such that $\sum_{e \in E} w(e)x(e)$ is as small as possible.

By the total unimodularity of the corresponding matrix, the optimum value of problem (8.2.13) is equal to the maximum value of

$$(8.2.14) \qquad \sum_{v \in V} (y_1(v) \cdot b_1(v) - y_2(v) \cdot b_2(v)) + \sum_{e \in E} (z_1(e) \cdot c_1(e) - z_2(e) \cdot c_2(e)),$$

where y_1, y_2 are nonnegative integral vectors in $\mathbb{R}^V$ and z_1, z_2 are nonnegative integral vectors in $\mathbb{R}^E$ such that $z_1(e) - z_2(e) + \sum_{v \in e}(y_1(v) - y_2(v))$ is equal to $w(e)$, for each edge $e \in E$.

Therefore, problem (8.2.13) and its dual include the problems given by (8.2.3) up to (8.2.12). Note that in (8.2.13) we may have negative values, and that problem (8.2.13) can be reduced easily to the case where $c_1 = 0$ and $b_1 \geq 0$.

As a special case of the previous min-max result we obtain:

(8.2.15) Supply-Demand Theorem. *Let $G = (V, E)$ be a bipartite graph with color classes S and T. Let $b : T \to \mathbb{R}_+$ be a "supply function", and $d : S \to \mathbb{R}_+$, a "demand function". Then this demand is "satisfiable from the supply", i. e., there exists a function $y : E \to \mathbb{R}_+$ such that*

$$\sum_{e \ni s} y(e) = d(s) \quad \text{for all } s \in S,$$

$$\sum_{e \ni t} y(e) \leq b(t) \quad \text{for all } t \in T,$$

if and only if

$$d(U) \leq b(\Gamma(U)) \quad \text{for all } U \subseteq S.$$

□

Here $\Gamma(U)$ denotes the set of nodes in T adjacent to at least one node in U.

KUHN (1955, 1956) developed a polynomial time algorithm to solve the assignment problem. This algorithm, called the "Hungarian method", at the same time also solves the dual of the assignment problem and has a running time of order $|V|^3$ (see also MUNKRES (1957) and BALINSKI and GOMORY (1964)). With some adaptations, this method also solves the problems (8.2.6) up to (8.2.12). In fact, also the general problem (8.2.13) can be solved in polynomial time, viz. by the minimum cost flow algorithm – cf. Section 8.3. In the same way as described above these problems can be solved with the ellipsoid method in polynomial time.

*8.3 Flows, Paths, Chains, and Cuts

We next discuss a first type of generalization of the problems on bipartite graphs dealt with in Section 8.2, namely problems on flows, dipaths and cuts in directed

graphs, and on chains and antichains in partially ordered sets. The problems and results treated below are in terms of *directed* graphs. The corresponding problems and methods for *undirected* graphs mostly follow straightforwardly by replacing undirected edges uv by two (directed) arcs (u,v) and (v,u).

One of the most important classes of combinatorial optimization problems is formed by the so-called **network flow problems**. The first breakthroughs in the theory and the applications of combinatorial optimization were obtained in this field, and they initiated various areas of research like "polyhedral combinatorics" and "good algorithms". The classical monograph on flow theory is FORD and FULKERSON (1962).

We start with some easy problems and results.

Let $D = (V, A)$ be a digraph. A **potential** is a function $\phi : V \to \mathbb{Q}$. Consider the following problem.

(8.3.1) Maximum Potential Problem. *Given a digraph $D = (V, A)$, two different nodes $r, s \in V$, and a length function $l : A \to \mathbb{Q}_+$, find a potential ϕ such that $\phi(v) - \phi(u) \leq l(a)$ for each arc $a = (u, v) \in A$ and such that the potential difference $\phi(s) - \phi(r)$ is as large as possible.*

It is trivial to show that the optimum value of this problem is equal to the optimum value of the following problem.

(8.3.2) Shortest Dipath Problem. *Given a digraph $D = (V, A)$, two different nodes $r, s \in V$, and a length function $l : A \to \mathbb{Q}_+$, find an (r,s)-dipath in D of minimum length.*

The following (easy) theorem will turn out to be analogous to the Max-Flow Min-Cut Theorem (7.1.4).

(8.3.3) Max-Potential Min-Work Theorem. *For any digraph $D = (V, A)$, any two different nodes $r, s \in V$, and any length function $l : A \to \mathbb{Q}_+$, the maximum potential difference $\phi(s) - \phi(r)$ subject to $\phi(v) - \phi(u) \leq l(a)$ for each arc $a = (u, v) \in A$ is equal to the minimum length of an (r,s)-dipath in D. Moreover, if all lengths are integral, there exists an integral optimum potential.*

Proof. To see that both optimum values are equal, first observe that the optimum value of (8.3.1) cannot be more than the optimum value of (8.3.2), and second that defining $\phi(v) :=$ the minimum length of an (r,v)-dipath, for v in V, yields a potential for which equality holds. □

These results can also be derived from the total unimodularity of the corresponding matrices.

If ϕ is an integer potential on $D = (V, A)$ with $\phi(s) - \phi(r) = t$, then there exist (r,s)-cuts $C_1, \ldots, C_t$ such that for each arc $a = (u, v)$ the number of those i for which arc a belongs to C_i is at most $\phi(v) - \phi(u)$. This then implies the following combinatorial version of the max-potential min-work theorem, which seems to have been first observed by FULKERSON (1968).

(8.3.4) Theorem. *For any digraph $D = (V, A)$ and any two different nodes $r, s \in V$, the maximum number of pairwise disjoint (r,s)-cuts is equal to the minimum number of arcs in an (r,s)-dipath.* □

Both problems (8.3.1) and (8.3.2) can be solved in polynomial time by simple algorithms, e. g., the ones by DIJKSTRA (1959), PRIM (1957), MOORE (1959), FLOYD (1962), WARSHALL (1962).

The maximum potential problem (8.3.1) is an explicit linear program and can hence be solved in polynomial time by Khachiyan's method as well. The shortest (r,s)-dipath problem (8.3.2) is the dual of this linear program, namely (8.3.2) can be written as:

$$\begin{aligned} (8.3.5) \qquad \min \sum_{a \in A} l_a x_a - x(\delta^+(v)) + x(\delta^-(v)) &= \begin{cases} 0 & \text{if } r \neq v \neq s, \\ -1 & \text{if } v = r \\ 1 & \text{if } v = s, \end{cases} \\ x_a &\geq 0 \quad \text{for all } a \in A. \end{aligned}$$

The incidence vector of any (r,s)-dipath is obviously a feasible solution of (8.3.5). Suppose we have any optimum solution x^* of (8.3.5). We can construct from x^* an integral solution as follows. By depth-first search we can easily find an (r,s)-dipath in $D = (V, A^*)$ where $A^* = \{a \in A \mid x_a^* > 0\}$. This dipath $v_0, v_1, v_2, \ldots, v_k$, say, with $v_0 = r$, $v_k = s$, is a shortest (r,s)-dipath, since for the optimal potential ϕ^* we have by complementary slackness – see (0.1.50) –

$$\phi^*(v_i) - \phi^*(v_{i-1}) = l_{v_{i-1}v_i}, \; i = 1, \ldots, k,$$

and hence by adding these equations, $\phi^*(s) - \phi^*(r) = l_{v_0v_1} + l_{v_1v_2} + \ldots + l_{v_{k-1}v_k}$. Another way to obtain a shortest (r,s)-dipath is to find an optimum vertex solution of (8.3.5). From the total unimodularity of (8.3.5) it follows that this is the incidence vector of a shortest (r,s)-dipath.

Some of the algorithms mentioned above allow negative lengths, provided that no directed cycle of negative total length occurs. If we did not pose any condition like this the problem would become $\mathcal{NP}$-complete.

When negative lengths occur we should be careful in deriving similar results for undirected graphs: replacing an undirected edge of negative length by two oppositely oriented arcs would create a directed cycle of negative length. Yet the shortest path problem for undirected graphs is polynomially solvable also if negative lengths occur, provided that no (undirected) circuit of negative total length occurs. This however is more difficult – see the remarks after (8.5.17).

Shortest dipath algorithms can be applied to solve several other combinatorial optimization problems. The problems of finding a shortest directed cycle in a digraph or a shortest circuit in a graph can be trivially reduced to the shortest dipath problem. As a less trivial example, consider the following problem:

(8.3.6) Shortest Odd Cycle Problem. *Given a digraph $D = (V, A)$ and a length function $l : A \to \mathbb{Q}_+$, find an odd directed cycle of minimum length.*

Problem (8.3.6) can be solved by at most $|V|$ applications of a shortest dipath algorithm as follows. Split each node $v \in V$ into two nodes v_1 and v_2. For each arc $(v, w) \in A$ make new arcs (v_1, w_2) and (v_2, w_1), both of length l_{vw}. Let D' be the digraph constructed this way. For each $v \in V$, find a shortest (v_1, v_2)-dipath in D'. The shortest among these dipaths will give us the shortest odd directed cycle in D.

This algorithm also gives us a method to find a shortest odd circuit in an undirected graph. The problem of finding a shortest even circuit in an undirected graph with nonnegative edge lengths is also polynomially solvable. Namely, this problem can be reduced trivially to $|E|$ shortest odd path problems. This latter problem can be reduced to a minimum weight perfect matching problem by taking two copies of the graph and joining corresponding nodes by edges of weight 0.

Because of the polynomial solvability of (8.3.6) one might expect similarly that the even version of (8.3.6) is also easy. But at present it is not known whether the problem of deciding whether a digraph contains an even directed cycle is hard or easy. In fact, THOMASSEN (1985) showed that the problem of deciding whether a digraph $D = (V, A)$ contains an even resp. odd directed cycle containing a prescribed arc $a \in A$ is $\mathcal{NP}$-complete. This also implies that it is $\mathcal{NP}$-complete to find a shortest dipath of even resp. odd length in a digraph.

As a side remark we mention that there are some open polyhedral questions with respect to odd and even cycles, circuits and paths. Since – for nonnegative weight functions – we can solve the shortest cycle and odd cycle problem in digraphs and the shortest circuit, odd circuit, even circuit, odd path, and even path problem in undirected graphs, we can minimize nonnegative weight functions over the dominant of the convex hull of the incidence vectors of cycles, odd cycles, circuits, odd circuits etc.. Hence by the ellipsoid method, for every inequality $c^T x \geq \gamma$, $c \geq 0$ valid for one of these dominants we can find inequalities $a_1^T x \geq \alpha_1, \ldots, a_k^T x \geq \alpha_k$ defining facets of the dominant such that $c = a_1 + \ldots + a_k$ and $\gamma \leq \alpha_1 + \ldots + \alpha_k$ in polynomial time. But for none of these dominants is an explicit description by a system of linear inequalities (let alone a list of facets) known.

We now relate the results stated above to flow problems. Recall the following fundamental problem of flow theory from Section 7.1.

(8.3.7) Maximum (r, s)-Flow Problem. *Given a digraph $D = (V, A)$, two different nodes $r, s \in V$ and a capacity function $c : A \to \mathbb{Q}_+$, find an (r, s)-flow subject to c of maximum value.*

There are numerous good algorithms to solve the maximum flow problem with which instances of tremendous sizes can be solved very quickly in practice. As mentioned in Section 7.1, the most prominent algorithm among these is the Ford-Fulkerson method. A polynomial implementation of this method was found by DINITS (1970). Surveys of the state-of-the-art on theoretical efficiency can be found in GALIL (1981) and PAPADIMITRIOU and STEIGLITZ (1982), and on practical efficiency in GLOVER, KLINGMAN, MOTE and WHITMAN (1979), CHEUNG (1980),

TARJAN (1986). An interesting new method was recently discovered by GOLDBERG and TARJAN (1986).

We have discussed that Ford and Fulkerson's algorithm directly also solves the following problem:

(8.3.8) Minimum (r,s)-Cut Problem. *Given a digraph $D = (V,A)$, two different nodes $r,s \in V$, and a capacity function $c : A \to \mathbb{Q}_+$, find an (r,s)-cut of minimum capacity.*

The famous max-flow min-cut theorem has already been stated in (7.1.4). There is an alternative way of formulating this theorem. Notice that if $D = (V,A)$ is a digraph, $r,s \in V$, and x is an integer (r,s)-flow of value t, then there exist (r,s)-dipaths $P_1, \ldots, P_t$ in D such that for each arc a, the number of those i for which arc a belongs to P_i is at most x_a. This implies:

(8.3.9) Max-Flow Min-Cut Theorem (second version). *For any digraph $D = (V,A)$, any two different nodes $r,s \in V$ and any capacity function $c : A \to \mathbb{Z}_+$, the minimum capacity of an (r,s)-cut is equal to the maximum number of (not necessarily distinct) (r,s)-dipaths such that each arc a is contained in at most c_a of these dipaths.* □

This version of the max-flow min-cut theorem may be less appealing than the original one, but there is an interesting point to make here. We have seen that the problem of finding a minimum (r,s)-cut is equivalent to solving the linear programming dual of the max-flow problem – cf. (7.1.1) and (7.1.3). Let us try to treat the minimum cut problem more directly. Let S be the set of all incidence vectors of (r,s)-cuts and consider the polyhedron $\text{conv}(S) + \mathbb{R}^A_+$, i. e., the dominant of $\text{conv}(S)$. The Max-Flow Min-Cut Theorem (8.3.9) implies that this polyhedron is determined by the inequalities

$$\begin{aligned} x(P) &\geq 1 \quad \text{for all } (r,s)\text{-dipaths } P \subseteq A, \\ x_a &\geq 0 \quad \text{for all } a \in A. \end{aligned} \tag{8.3.10}$$

And so, for $c \geq 0$, the minimum (r,s)-cut problem (8.3.8) is equivalent to the linear program:

$$\begin{aligned} &\min c^T x \\ &x(P) \geq 1 \quad \text{for all } (r,s)\text{-dipaths } P \subseteq A, \\ &x(a) \geq 0 \quad \text{for all } a \in A. \end{aligned} \tag{8.3.11}$$

Now the second version (8.3.9) of the max-flow min-cut theorem states that if c is nonnegative and integral, problem (8.3.11) and its LP-dual have integral optimum solutions.

These observations also show that the minimum cut problem can be reduced to the shortest (r,s)-dipath problem using the ellipsoid method in a way different from the treatment in Section 7.1. In fact, the shortest dipath problem is just

the separation problem for the polyhedron (8.3.10). So we can solve (8.3.11) in polynomial time using a shortest dipath algorithm as separation subroutine.

Moreover, we can determine an optimum dual solution of (8.3.11) which yields a maximum (r,s)-flow. Thus, also in this manner the ellipsoid method gives us a (quite inefficient) polynomial time max-flow algorithm.

The second version of the max-flow min-cut theorem (8.3.9) implies a classical theorem of MENGER (1927). Namely, if we take all capacities equal to 1 in (8.3.9) we obtain:

(8.3.12) Menger's Theorem. *For any digraph $D = (V,A)$ and any two different nodes $r,s \in V$, the maximum number of pairwise arc-disjoint (r,s)-paths is equal to the minimum cardinality of an (r,s)-cut.* □

In turn, one can derive (8.3.9) from (8.3.12) by replacing each arc a by $c(a)$ parallel arcs.

König's matching theorem (8.2.3) is a special case of Menger's theorem. To see this, let $G = (V,E)$ be a bipartite graph, say with color classes V_1 and V_2. Then orient the edges of G from V_1 to V_2, add two new nodes r and s, and add arcs (r,v) for each v in V_1, and arcs (v,s) for each v in V_2. Then Menger's theorem for this directed graph gives König's matching theorem. Also Theorem (8.2.6) can be derived in a similar way from the max-flow min-cut theorem. In fact, ORDEN (1956) and HOFFMAN (1960) showed that by a direct construction also the converse implication holds.

The pairs of problems described above (shortest path and maximum flow) are contained in one problem, the minimum cost flow problem:

(8.3.13) Minimum-Cost Flow Problem. *Given a digraph $D = (V,A)$, two different nodes $r,s \in V$ and a capacity function $d : A \to \mathbb{Q}_+$, a cost function $c : A \to \mathbb{Q}_+$, and a rational number t, find an (r,s)-flow x, subject to the capacity function d, of value t, and with minimum cost $\sum_{a \in A} c_a x_a$.*

The minimum-cost flow problem (8.3.13) can be solved with the famous "out-of-kilter" method developed independently by YAKOVLEVA (1959), MINTY (1960), and FULKERSON (1961). This algorithm works very fast in practice; for a polynomially bounded version see EDMONDS and KARP (1972). TARDOS (1985) found a strongly polynomial time algorithm for the minimum cost flow problem; another such algorithm was given by ORLIN (1987).

The minimum-cost flow problem can be conceived of as a linear programming problem, with a totally unimodular constraint matrix. This yields a min-max relation for the minimum cost (and a good characterization) and gives that, if t and d are integer, there is an integer minimum-cost flow. This also shows that (8.3.13) can be solved by the ellipsoid method in polynomial time. By taking $t = 1$ and $d = \infty$ (or very large), problem (8.3.13) is equivalent to the shortest dipath problem (8.3.2). By taking $c = 0$, the problem is equivalent to the maximum (r,s)-flow problem (for instance, by binary search over t).

Note that, e. g., also the transportation problems (8.2.11) and (8.2.12) are special cases of the minimum-cost flow problem. But, as ORDEN (1956) and

HOFFMAN (1960) observed, also the converse is true! (See FORD and FULKERSON (1962) and PAPADIMITRIOU and STEIGLITZ (1982).)

A similar analysis can be made when studying flows satisfying lower bounds instead of upper bounds. This will lead to results analogous to the above. First consider the following problem.

(8.3.14) Minimum (r,s)-Flow Problem. *Given an acyclic digraph $D = (V,A)$, two different nodes $r,s \in V$ such that each arc of D is contained in some (r,s)-dipath, and a "requirement" function $d : A \to \mathbb{Q}_+$, find an (r,s)-flow x satisfying $x \geq d$ with minimum value.*

(The restrictions on D and r and s made here and below seem necessary, as otherwise the problems may become meaningless (by unboundedness or infeasibility), and since arcs contained in directed cycles or not contained in any (r,s)-dipath may be contracted or deleted.)

It is not difficult to derive from any max-flow algorithm an algorithm for problem (8.3.14). Indeed, let x_0 be an integer (r,s)-flow such that $x_0 \geq d$ (which exists by the assumptions), and let M be a large number. Then the minimum (r,s)-flow problem is equivalent to the maximum (r,s)-flow problem for the capacity function $Mx_0 - d$. Moreover, this construction yields that the optimum value for (8.3.14) is equal to the optimum value of the following problem, where an (r,s)-**dicut** is an (r,s)-cut of the form $\delta^+(V')$, with $r \in V'$, $s \notin V'$ and $\delta^-(V') = \emptyset$.

(8.3.15) Maximum (r,s)-Dicut Problem. *Given D, r, s, d as in* (8.3.14), *find an (r,s)-dicut of maximum requirement.*

Note that the following related problem is $\mathcal{NP}$-complete, as was shown by KARP (1972).

(8.3.16) Maximum (r,s)-Cut Problem. *Given a digraph (or graph) and two nodes r and s, find an (r,s)-cut (or $[r,s]$-cut) of maximum cardinality.*

By the construction described above, the Max-Flow Min-Cut Theorem (7.1.4) gives:

(8.3.17) Min-Flow Max-Cut Theorem. *For D, r, s, d as in* (8.3.14), *the minimum value of an (r,s)-flow x with $x \geq d$ is equal to the maximum requirement of an (r,s)-dicut. Moreover, if d is integer there exists an integer flow of minimum value.* □

This result can be derived alternatively from the total unimodularity of the corresponding matrix. Since an integer (r,s)-flow in an acyclic directed graph can always be decomposed into a sum of incidence vectors of (r,s)-dipaths, the following theorem is equivalent to (8.3.17):

(8.3.18) Theorem. *For any acyclic digraph $D = (V,A)$, and any r, s in V such that each arc of D is contained in some (r,s)-dipath, the minimum number of (r,s)-dipaths needed to cover A is equal to the maximum size of an (r,s)-dicut.* □

It is not difficult to see that Theorem (8.3.18) is equivalent to a classical theorem of DILWORTH (1950). Recall that a chain (antichain) in a partially ordered set is a set of pairwise comparable (incomparable) elements.

(8.3.19) Dilworth's Theorem. *If $(X, \leq)$ is a partially ordered set, then the minimum number of chains needed to cover X is equal to the maximum size of an antichain.* □

One easily derives König's edge covering Theorem (8.2.4) and Theorem (8.2.7) as special cases of each of the last three theorems above.

Polynomial algorithms to find maximum antichains and minimum coverings by chains can be derived from any maximum flow algorithm.

Again, there are easier parallel problems to these results, in terms of potentials. Consider the following problems.

(8.3.20) Minimum Potential Problem. *Given an acyclic digraph $D = (V, A)$, $r, s \in V$, and a length function $l : A \to \mathbb{Q}_+$, find a potential $\phi : V \to \mathbb{Q}$ such that $\phi(v) - \phi(u) \geq l(a)$ for each arc $a = (u, v)$, and such that $\phi(s) - \phi(r)$ is as small as possible.*

(8.3.21) Longest Dipath Problem. *Given a directed graph $D = (V, A)$, $r, s \in V$, and a length function $l : A \to \mathbb{Q}_+$, find an (r, s)-dipath of maximum length.*

In general, this second problem is $\mathcal{NP}$-complete. (The first is just a linear program.) However, for acyclic digraphs there is an easy, straightforward way to solve the problems (8.3.20) and (8.3.21) in polynomial time. This method is the basis of widely used project scheduling methods like PERT and CPM. In fact, the optimum values of (8.3.20) and (8.3.21) for acyclic digraphs are the same:

(8.3.22) Min-Potential Max-Work Theorem. *For any acyclic digraph $D = (V, A)$, two different nodes $r, s \in V$, and any length function $l : A \to \mathbb{Q}_+$, the minimum value of $\phi(s) - \phi(r)$, where ϕ is a potential such that $\phi(v) - \phi(u) \geq l(a)$ for each arc $a = (u, v)$, is equal to the maximum length of an (r, s)-dipath. If l is integer then there is an optimum potential that is integer.* □

Interpretation: l_a is the time needed for the job represented by arc a, and $\phi(s) - \phi(r)$ is the time needed for the total project. So by (8.3.22), the minimum time to finish the whole project (min-potential) equals the maximum time spent on a series of jobs (max-work).

Again, one easily derives the following equivalent theorems.

(8.3.23) Theorem. *If $D = (V, A)$ is an acyclic digraph and r, s are different nodes of D such that each arc of D is contained in some (r, s)-dipath, then the maximum number of arcs in an (r, s)-dipath is equal to the minimum number of (r, s)-dicuts needed to cover A.* □

(8.3.24) Theorem. *If $(X, \leq)$ is a partially ordered set, then the maximum size of a chain is equal to the minimum number of antichains needed to cover X.* □

Similarly to the combination of the max-flow and the shortest path problems to the minimum-cost flow problem, the problems above can be combined to a "maximum-gain" flow problem.

(8.3.25) Maximum-Gain Flow Problem. *Given an acyclic digraph $D = (V, A)$, different nodes r, s of D such that each arc of D is contained in some (r,s)-dipath, a requirement function $d : A \to \mathbb{Q}_+$, a gain function $c : A \to \mathbb{Q}_+$, and a rational number t, find an (r,s)-flow x of value t satisfying $x \geq d$ of maximum gain $\sum_{a \in A} c(a)x(a)$.*

By the total unimodularity of the corresponding matrix one easily derives a good characterization (a min-max relation) for (8.3.25). It also yields that if d and t are integral, there is an integer optimum flow. It is not difficult to derive the minimum (r,s)-flow problem (8.3.14) and the longest path problem (8.3.21) for acyclic digraphs as special cases. Moreover, the maximum-gain flow problem can be reduced to the minimum-cost flow problem (8.3.13) in the same way as we reduced the minimum (r,s)-flow problem (8.3.14) to the maximum (r,s)-flow problem (8.3.7). Hence, also problem (8.3.25) is polynomially solvable.

Finally, we consider the combined case of flows satisfying upper and lower bounds, or, rather, we consider circulations. Given a directed graph $D = (V, A)$, a **circulation** is a function $x : A \to \mathbb{Q}_+$ such that for *each* node v of D one has

$$x(\delta^-(v)) = x(\delta^+(v)). \tag{8.3.26}$$

Now the following problem contains both the minimum-cost flow problem and the maximum-gain flow problem.

(8.3.27) Minimum-Cost Circulation Problem. *Given a digraph $D = (V, A)$, functions $d_1, d_2 : A \to \mathbb{Q}_+$ and a cost function $c : A \to \mathbb{Q}$, find a circulation x satisfying $d_1 \leq x \leq d_2$ such that $\sum_{a \in A} c(a)x(a)$ is as small as possible.*

This problem reduces to the minimum-cost (r,s)-flow problem (8.3.13) in case $c \geq 0$ and d_1 has only one nonzero component. Similarly the max-gain flow problem (8.3.25) is contained in (8.3.27).

Again a min-max relation follows from the total unimodularity of the incidence matrix of D. Moreover, if d_1 and d_2 are integral such that (8.3.27) has an optimum solution then there is an integer optimum circulation. Note that if $c = 0$, (8.3.27) reduces to the existence problem for circulations – see HOFFMAN (1960).

The minimum-cost circulation problem can be solved in polynomial time by adapting the out-of-kilter method of Yakovleva, Minty and Fulkerson for the minimum-cost flow problem. In fact, this algorithm also solves the following, even more general problem.

(8.3.28) Problem. *Given a digraph $D = (V, A)$, and functions $r_1, r_2 : V \to \mathbb{Q}$ and $d_1, d_2, c : A \to \mathbb{Q}$, find a function $x : A \to \mathbb{R}$ such that $d_1 \leq x \leq d_2$ and $r_1(v) \leq x(\delta^-(v)) - x(\delta^+(v)) \leq r_2(v)$ for each vertex v, and such that $\sum_{a \in A} c(a)x(a)$ is as small as possible.*

Clearly, problem (8.3.28) contains many of the problems treated in this section and in the previous section on bipartite graphs as special cases.

The total unimodularity of the incidence matrix of D gives a good characterization for problem (8.3.28) and the integrality of an optimum solution, if r_1, r_2, d_1, and d_2 are integer. Thus, problem (8.3.28) (and therefore all its special cases) can be viewed as explicit linear programs, and hence they can be solved with Khachiyan's version of the ellipsoid method in polynomial time. Integral optimum solutions (given integral constraints) can be obtained by calculating optimum vertex solutions.

*8.4 Trees, Branchings, and Rooted and Directed Cuts

In Section 8.3 we studied problems on connecting *one* specified pair of nodes in a digraph by dipaths or flows. We now look at problems in which we wish to connect *several* pairs of nodes at the same time. That is, instead of (r, s)-dipaths we consider sets of arcs which connect all given pairs $(r_1, s_1), \ldots, (r_t, s_t)$. (This type of problems differs from the one to be treated in Section 8.6 on multicommodity flows, where we consider sets of arcs which connect at least one of the given pairs $(r_1, s_1), \ldots, (r_t, s_t)$. As dual combinatorial objects we obtain in this section cuts that separate *at least one pair* among the given pairs, whereas in Section 8.6 on multicommodity flows, cuts are studied that separate *all given pairs* of nodes.)

Arborescences and Rooted Cuts

We first study arborescences and rooted cuts, a subject to which we gave an introduction in Section 7.2. Recall that an **arborescence** in a digraph $D = (V, A)$ is a set A' of arcs making up a spanning tree such that each node of D is entered by at most one arc in A'. It follows that there is exactly one node r that is not entered by any arc of A'. This node is called the **root** of A', and A' is called **rooted** in r, or an **r-arborescence**.

So an r-arborescence has exactly $|V| - 1$ arcs, and contains a unique (r, v)-dipath, for each node v of D. In fact, r-arborescences can be considered as the inclusionwise minimal sets of arcs that contain (r, v)-dipaths for all v in V.

Now consider the following analogue of the shortest dipath problem (8.3.2):

(8.4.1) Shortest r-Arborescence Problem. *Given a digraph $D = (V, A)$, a node $r \in V$, and a length function $l : A \to \mathbb{Q}$, find an r-arborescence of shortest length.*

Note that l may have negative values, so that the problem of finding a longest r-arborescence is included. CHU and LIU (1965), EDMONDS (1967a), BOCK (1971), and TARJAN (1977) designed polynomial algorithms to solve this problem. The

fastest method to solve (8.4.1) known at present is described in GABOW, GALIL, SPENCER and TARJAN (1986). It turned out that the the algorithms for the shortest r-arborescence problem also solve the following dual problem:

Given a digraph $D = (V, A)$, a node $r \in V$, and a length function $l : A \to \mathbb{Z}_+$, find a maximum family of (not necessarily distinct) r-cuts such that no arc a is contained in more than $l(a)$ of these r-cuts.

There is a difficulty with the posing of this problem since the output may take exponential space. For example, if all lengths $l(a)$ are equal to the same value L then the solution contains at least L r-cuts while the input has size $O(|V|^2 \log L)$. Therefore, we have to encode the output more compactly by listing only the distinct cuts together with their multiplicity. This leads to the following formulation of the problem.

(8.4.2) Rooted Cut Packing Problem. *Given a directed graph $D = (V, A)$, $r \in V$, and a length function $l : A \to \mathbb{Z}_+$, find r-cuts $C_1, \ldots, C_k$ and nonnegative integers $\lambda_1, \ldots, \lambda_k$ such that for each arc a of D the sum $\sum_{i, a \in C_i} \lambda_i$ is not more than $l(a)$ and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

It follows from the results below that there is always an optimum solution of (8.4.2) with at most $2|V| - 1$ distinct r-cuts. The solution of (8.4.2) can therefore be described in space polynomial in the input length of the problem. In fact, FULKERSON (1974) showed that for nonnegative integral l, the optimum values of (8.4.1) and (8.4.2) are equal, and hence the following theorem follows.

(8.4.3) Fulkerson's Optimum Arborescence Theorem. *For any digraph $D = (V, A)$, any node r in V, and any length function $l : A \to \mathbb{Z}_+$, the minimum length of an r-arborescence is equal to the maximum number of (not necessarily distinct) r-cuts such that each arc a of D is contained in at most $l(a)$ of them.* □

(Note that the cardinality version ($l = \mathbb{1}$) is trivial.) Fulkerson's theorem (8.4.3) can be equivalently stated as: the clutter of r-cuts has the $\mathbb{Z}_+$-MFMC property. In other words, both sides of the linear programming duality relation

$$\text{(8.4.4)} \qquad \min \sum_{a \in A} l_a x_a \qquad = \max \sum_{C\ r\text{-cut}} \lambda_C$$

$$x(C) \geq 1 \text{ for all } r\text{-cuts } C \subseteq A, \qquad \sum_{\substack{C\ r\text{-cut} \\ a \in C}} \lambda_C \leq l_a \text{ for all } a \in A,$$

$$x_a \geq 0 \text{ for all } a \in A, \qquad \lambda_C \geq 0 \text{ for all } r\text{-cuts C},$$

have integer optimum solutions, if $l \in \mathbb{Z}_+^A$. This in particular implies that the solution set of the minimization problem in (8.4.4) is the polyhedron $\text{conv}\{\chi^B \in \mathbb{R}^A \mid B\text{r-arborescence in } D\} + \mathbb{R}_+^A$, i. e., is equal to the dominant of the convex hull of incidence vectors of r-arborescences (see also Section 7.2). We have shown in Section 7.2 that the minimization problem in (8.4.4) can be solved in polynomial time using the ellipsoid method with a minimum capacity r-cut algorithm as

separation subroutine. As we can find an optimum vertex with the ellipsoid method, we can find an integer optimum solution, and thus we can solve the shortest r-arborescence problem (8.4.1) in polynomial time.

With the ellipsoid method, we can also find an optimum dual solution, but we do not know a direct way to find an integer optimum solution, although we know that integral optimum solutions to the minimization problem in (8.4.4) exist. There is, however, an ad-hoc trick to construct an integral optimum solution from a fractional one. The method goes as follows.

Suppose we have found an optimum fractional solution $\lambda_{C_1}, \ldots, \lambda_{C_k}$ to the maximization problem in (8.4.4). Without loss of generality $\lambda_{C_i} > 0$ for $i = 1, \ldots, k$. We can trivially find nonempty subsets $S_1, \ldots, S_k \subseteq V \setminus \{r\}$ such that $C_i = \delta^-(S_i)$, $i = 1, \ldots, k$. From the system $S_1, \ldots, S_k$ of node sets we want to construct a new system $S_1', \ldots, S_{k'}' \subseteq V \setminus \{r\}$ such that no two sets **cross**, i. e., such that $S_i' \cap S_j' \neq \emptyset$ implies $S_i' \subseteq S_j'$ or $S_j' \subseteq S_i'$. We proceed by the following **uncrossing technique**. If $S_i \cap S_j \neq \emptyset$, $S_i \not\subseteq S_j$, and $S_j \not\subseteq S_i$ then we set

(8.4.5)
$$\begin{aligned} \varepsilon &:= \min\{\lambda_{\delta^-(S_i)}, \lambda_{\delta^-(S_j)}\}, \quad \text{and we reset} \\ \lambda_{\delta^-(S_i)} &:= \lambda_{\delta^-(S_i)} - \varepsilon, \\ \lambda_{\delta^-(S_j)} &:= \lambda_{\delta^-(S_j)} - \varepsilon, \\ \lambda_{\delta^-(S_i \cup S_j)} &:= \lambda_{\delta^-(S_i \cup S_j)} + \varepsilon, \\ \lambda_{\delta^-(S_i \cap S_j)} &:= \lambda_{\delta^-(S_i \cap S_j)} + \varepsilon. \end{aligned}$$

This means, we replace the two sets S_i and S_j in the system $\{S_1, \ldots, S_k\}$ by $S_i \cup S_j$, $S_i \cap S_j$ and one of the sets S_i, S_j, and adjust the λ's such that the new λ's form an optimum dual solution. It is easy to see that after a finite number of steps a cross-free collection $S_1', \ldots, S_{k'}' \subseteq V$ is obtained with associated values $\lambda_1', \ldots, \lambda_{k'}'$. In fact, one can prove that $O(|V|^2)$ uncrossing steps (8.4.5) suffice to get such a collection – cf. the proof of Theorem (10.3.28). Now consider the following linear programming problem (restricted dual):

(8.4.6)
$$\begin{aligned} \max & \sum_{i=1}^{k'} \mu_{\delta^-(S_i')} \\ & \sum_{a \in \delta^-(S_i')} \mu_{\delta^-(S_i')} \leq l_a \quad \text{for all } a \in A, \\ & \mu_{\delta^-(S_i')} \geq 0 \quad i = 1, \ldots, k'. \end{aligned}$$

Since the values $\lambda_1', \ldots, \lambda_{k'}'$ constructed above form a feasible solution of (8.4.6) we know that the optimum value of (8.4.6) is the same as the optimum value of the minimization problem in (8.4.4). Moreover, since the node sets $S_1', \ldots, S_{k'}'$ are crossfree, the constraint matrix in (8.4.6) is totally unimodular – see EDMONDS and GILES (1977). This shows that (8.4.6) has an integer optimum solution, which we can compute in polynomial time with the ellipsoid method. This solution is of course also an integral optimum solution to the maximization problem in (8.4.4) and therefore an optimum solution to the rooted cut packing problem (8.4.2).

The Shortest Dipath Problem (8.3.2) and Theorem (8.3.4) are special cases of (8.4.1) and (8.4.3): to find a shortest (r,s)-dipath, add arcs (s,v) for all v in V, with length 0, and solve the shortest r-arborescence problem.

Furthermore, the shortest r-arborescence problem is easily seen to be equivalent to the following problems.

(8.4.7) Shortest Arborescence Problem. *Given a digraph $D = (V,A)$ and a length function $l : A \to \mathbb{Q}$, find an arborescence of minimum length.*

A **branching** in a directed graph $D = (V,A)$ is a set A' of arcs not containing circuits such that each node of D is entered by at most one arc in A' (so the arcs in A' make up a forest).

(8.4.8) Longest Branching Problem. *Given a digraph $D = (V,A)$ and a function $l : A \to \mathbb{Q}$, find a branching of maximum length.*

EDMONDS (1967a) not only designed a polynomial time algorithm to solve (8.4.8), but also gave a complete linear characterization of the branching polytope. He proved

$$\begin{aligned}(8.4.9)\qquad \operatorname{conv}\{\chi^B \in \mathbb{R}^A \mid B \subseteq A \text{ branching}\} &= \\ = \{x \in \mathbb{R}^A \mid\ & (1)\quad x(\delta^-(v)) \leq 1 && \text{for all } v \in V,\\ & (2)\quad x(A(W)) \leq |W| - 1 && \text{for all } \emptyset \neq W \subseteq V,\\ & (3)\quad x_a \geq 0 && \text{for all } a \in A\}.\end{aligned}$$

This relates to the theory discussed in Section 7.5. The polytope defined by the system of inequalities (2) and (3) is a matroid polytope, the vertices of which are the incidence vectors of the forests in D. Since the strong separation problem for matroid polytopes is solvable in polynomial time, and since the inequalities (1) in (8.4.9) are easy to check by substitution, we have a polynomial time algorithm to solve the separation problem for the inequality system (1), (2), (3). Hence, the polynomial time solvability of the longest branching problem (8.3.8) follows from the ellipsoid method. In fact, as we saw in Section 7.5, there is another matroid hidden in this problem. Inequalities (1) and (3) define a polytope whose vertices are the incidence vectors of the partition matroid on A defined by the partition $\delta^-(v)$, $v \in V$ – cf. (7.5.19). So the branching polytope (8.4.9) can be viewed as the intersection of two matroid polytopes and therefore our analysis of the 2-matroid intersection problem in Section 7.5 applies for this case.

We have seen a linear description of the convex hull of incidence vectors of r-arborescences (Section 7.2). It is easy to derive from (8.4.9) a linear description of the convex hull of the incidence vectors of *all* arborescences of a digraph D. One can simply add the equation $x(A) = n-1$ to the system (1), (2), (3) in (8.4.9) to obtain such a characterization. This gives another possible line of attack on the shortest arborescence problem (8.4.7) via the ellipsoid method.

Again, as we did with flows, these problems and results can be "dualized".

(8.4.10) Minimum Rooted Cut Problem. *Given a digraph $D = (V,A)$, a node r of D, and a capacity function $c : A \to \mathbb{Q}_+$, find an r-cut of minimum capacity.*

(8.4.11) Arborescence Packing Problem. *Given a digraph $D = (V, A)$, a node r of D, and a capacity function $c : A \to \mathbb{Z}_+$, find nonnegative integers $\lambda_1, \ldots, \lambda_k$ and r-arborescences $B_1, \ldots, B_k$ such that, for each arc a of D, the sum $\sum_{i,a\in B_i} \lambda_i$ is at most $c(a)$, and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

EDMONDS (1973) showed that both problems (8.4.10) and (8.4.11) have the same optimum value:

(8.4.12) Edmonds' Disjoint Arborescence Theorem. *For any digraph $D = (V, A)$ and any node r of D, the minimum cardinality of an r-cut is equal to the maximum number of pairwise disjoint r-arborescences.* □

Edmonds' theorem (8.4.12) is equivalent to the fact that the clutter of r-arborescences has the $\mathbb{Z}_+$-MFMC property. In other words, both sides of the linear programming duality relation

$$(8.4.13)\qquad \begin{aligned} \min \sum_{a\in A} c_a x_a \quad & = \quad \max \sum_B \lambda_B \\ x(B) \geq 1 \text{ for all } B \in \mathscr{A}, \quad & \quad \sum_{B \ni a} \lambda_B \leq c_a \text{ for all } a \in A, \\ x_a \geq 0 \text{ for all } a \in A, \quad & \quad \lambda_B \geq 0 \text{ for all } B \in \mathscr{A}, \end{aligned}$$

have integral optimum solutions, if $c \in \mathbb{Z}_+^A$ (where $\mathscr{A}$ is the set of all r-arborescences in D). This in particular implies that the solution set of the minimization problem in (8.4.13) is the polyhedron $\operatorname{conv}\{\chi^C \in \mathbb{R}^A \mid C \text{ } r\text{-cut in } D\} + \mathbb{R}_+^A$, i. e., is equal to the dominant of the convex hull of incidence vectors of r-cuts. This last statement also follows from Fulkerson's optimum arborescence theorem (8.4.3) via Lehman's theorem (8.1.5). However, the integrality of the dual solution does not follow from (8.4.3) via (8.1.5). We have seen in Section 7.2 that not only the dominant but also the convex hull ARB(D) of the incidence vectors of the r-arborescences of D can be described by a nice system of linear inequalities. In particular one can optimize over ARB(D) in polynomial time. This does not remain true for the convex hull of the incidence vectors of r-cuts. In fact, it is $\mathscr{NP}$-complete to find a maximum size r-cut.

It is not difficult to show that the minimum rooted cut problem is polynomially solvable: to this end just solve the minimum (r, s)-cut problem for each node $s \neq r$, and take the minimum of the optimum values. TARJAN (1974) and LOVÁSZ (1976) showed that the arborescence packing problem (8.4.11) can be solved in polynomial time if $c(a) = 1$ for all $a \in A$.

Problem (8.4.11) can be reduced to the case $c = \mathbb{1}$ by replacing each arc a by c_a parallel arcs. But this reduction is not polynomial. However, a polynomial time reduction can be obtained using the ellipsoid method as follows.

First consider the dual pair of linear programs (8.4.13), where $c \in \mathbb{Z}_+^A$. As mentioned above, for any objective function the primal problem in (8.4.13) is equivalent to finding a minimum r-cut and therefore it is easily solved. Hence we can also find a basic optimum dual solution in polynomial time, say y^*. So y^* has no more than $|A|$ positive components. Set

$$(8.4.14)\qquad c'_a := \left\lceil \sum_{B \ni a} (y^*_B - \lfloor y^*_B \rfloor) \right\rceil \quad \text{for all } a \in A.$$

Clearly, $y_B^* - \lfloor y_B^* \rfloor$, is an optimum solution for the maximization problem in (8.4.13) if we replace c by c' (in fact, $y^* - \lfloor y^* \rfloor$ is a feasible solution in (8.4.13) and satisfies the complementary slackness conditions with the same optimal primal solution). Conversely if, say z^*, is any optimal solution of the dual problem with capacities c', then $\lfloor y^* \rfloor + z^*$ is an optimum solution for the original capacities c. So it suffices to find an integral optimum solution for the capacities c'. But for these new capacities the trivial reduction of adding parallel arcs described above works in polynomial time, since $c'_a \leq |A|$ for all $a \in A$.

Observe that this algorithm is only strongly polynomial if we allow rounding as an elementary operation. As pointed out to us by A. Frank, the proof of (8.4.12) given by MADER (1983) can be turned into a strongly polynomial algorithm for the arborescence packing problem, which is strongly polynomial even in the stronger sense.

So, to a large extent there exists an analogy between (r,s)-dipaths and (r,s)-cuts on the one side and r-arborescences and r-cuts on the other side. There arise more complications if we do not want to fix r either. When pursuing analogy, we will arrive at the problem of connecting *all* pairs of nodes in a directed graph. Therefore, consider the following problem.

(8.4.15) Minimum Strongly Connected Subdigraph Problem. *Given a digraph $D = (V,A)$ and a length function $l : A \to \mathbb{Q}_+$, find a set A' of arcs such that (V,A') is strongly connected and A' has minimum length.*

However, this problem is $\mathcal{NP}$-complete (as the asymmetric traveling salesman problem is included). Also the problem of packing strongly connected subdigraphs turns out to be $\mathcal{NP}$-complete. Yet, the following problem can be solved in polynomial time, e.g., by applying $|V|$ $(|V|-1)$ times a minimum (r,s)-cut algorithm (in fact, $|V|$ applications suffice):

(8.4.16) Minimum Cut Problem. *Given a digraph $D = (V,A)$ and a capacity function $c : A \to \mathbb{Q}_+$, find a cut of minimum capacity.*

Trees and Cuts in Undirected Graphs

Some optimization problems for directed graphs turn out to be easier in their undirected version.

(8.4.17) Minimum Spanning Tree Problem. *Given an undirected graph $G = (V,E)$ and a length function $l : E \to \mathbb{Q}$, find a spanning tree of minimum length.*

This problem is the undirected analogue of (8.4.15). It is polynomially solvable, e.g., by the optimum arborescence methods treated above; much simpler, however, by the greedy algorithm, since spanning trees form the bases of the graphic matroid of a graph – see BORŮVKA (1926), KRUSKAL (1956), PRIM (1957), DIJKSTRA (1959), YAO (1975), CHERITON and TARJAN (1976). The – at present – theoretically fastest method for solving (8.4.17) is described in GABOW, GALIL, SPENCER and TARJAN (1986). An account of the history of spanning tree

algorithms can be found in GRAHAM and HELL (1985). Since lengths may be negative, the maximum spanning tree problem is included in (8.4.17).

One may ask for a min-max relation corresponding to (8.4.17). The most obvious one in terms of packing of cuts turns out not to hold, but a good characterization in terms of more general cuts was given by EDMONDS (1970).

A fractional version of the minimum spanning tree problem is as follows.

(8.4.18) Problem. *Given an undirected graph $G = (V, E)$ and a length function $l : E \to \mathbb{Q}_+$, find a nonnegative vector $x \in \mathbb{R}^E$ such that $x(\delta(W)) \geq 1$ for each set W of nodes with $\emptyset \neq W \neq V$ and such that $\sum_{e \in E} l(e)x(e)$ is as small as possible.*

Note that requiring x to be integer makes the problem equivalent to the minimum spanning tree problem, but that requiring x to be half-integer (i. e., $x_e \in \frac{1}{2}\mathbb{Z}$ for all $e \in E$) would make the problem $\mathcal{NP}$-complete as the symmetric traveling salesman problem would be included. However, requiring x to be just fractional gives again a polynomially solvable problem (which can be used as lower bound in a branch-and-bound procedure for the traveling salesman problem). In fact, the following more general problem is polynomially solvable.

(8.4.19) Network Synthesis Problem. *Given a natural number n, and nonnegative integers l_{ij}, r_{ij} for $1 \leq i < j \leq n$, design a graph G on n nodes with (possibly fractional) capacities c_{ij} on edge ij $(1 \leq i < j \leq n)$, such that between every pair i,j of nodes there exists an $[i,j]$-flow (subject to c) of value at least r_{ij}, and such that the cost $\sum_{1 \leq i < j \leq n} l_{ij} c_{ij}$ is as small as possible.*

By the max-flow min-cut theorem (7.1.4), problem (8.4.18) is the special case of (8.4.19) where $r_{ij} = 1$ for all i,j.

GOMORY and HU (1964) showed that the network synthesis problem is polynomially solvable if $l_{ij} = 1$ for all i,j. BLAND, GOLDFARB and TODD (1981) showed that the general problem is polynomially solvable with the ellipsoid method.

To see this, problem (8.4.19) can be viewed as the following linear program:

$$\text{(8.4.20)} \qquad \min \sum_{1 \leq i < j \leq n} l_{ij} x_{ij}$$

$$x(\delta(W)) \geq r_{ij} \quad \text{for all } W \subseteq V \text{ with } i \in W, j \in V \setminus W,$$
$$x_{ij} \geq 0 \quad \text{for } 1 \leq i < j \leq n.$$

The separation problem for the feasible region of (8.4.20) can be easily solved by running $\binom{|V|}{2}$ minimum $[i,j]$-cut algorithms. A similar approach works for the directed case.

Now consider the dual problems of minimizing cuts and packing spanning trees in undirected graphs.

(8.4.21) Minimum Cut Problem (undirected case). *Given a graph $G = (V, E)$ and a capacity function $c : E \to \mathbb{Q}_+$, find a cut of minimum capacity.*

Clearly, this problem can be solved in polynomial time, for instance, by solving $\frac{1}{2}|V|(|V| - 1)$ times a minimum $[r,s]$-cut problem. GOMORY and HU

(1961) – cf. HU (1969) – showed that the following problem (which includes the minimum cut problem) can be solved by applying not more than $|V|-1$ minimum $[r,s]$-cut algorithms.

(8.4.22) Multiterminal Flow Problem. *Given a graph $G=(V,E)$ and a capacity function $c: E \to \mathbb{Q}_+$, find, for all r, s in V, an $[r,s]$-cut of minimum capacity.*

In fact, minimum cuts for all pairs of nodes can be described in the following compact and elegant form. Let $G=(V,E)$ be an undirected graph and $c: E \to \mathbb{Q}_+$ a capacity function. A **Gomory-Hu tree** for G is a tree $T=(V,F)$ on the same set of nodes such that, for each edge $ij \in F$, the cut C_{ij} of G formed by those edges of G connecting the two components of $T-ij$ is a minimum $[i,j]$-cut in G. We define the capacity d_{ij} of $ij \in F$ as the capacity of C_{ij} in G. (Note that F is not necessarily a subset of E!) It follows that for any two nodes $r,s \in V$ a minimum $[r,s]$-cut in G can be found by selecting an edge $ij \in F$ on the unique $[r,s]$-path in T such that d_{ij} is minimum. Then the cut C_{ij} is a minimum $[r,s]$-cut in G. This implies that the Gomory-Hu tree is **flow-equivalent**, i. e., for any pair $r,s \in V$ the maximum value of an $[r,s]$-flow in G is the same as that in T.

(8.4.23) Theorem. *Each undirected graph with nonnegative capacities has a Gomory-Hu tree.* □

Such a tree can be found in polynomial time by choosing $|V|-1$ cuts recursively so that each new cut is a minimum cut for some pair of nodes which was not yet separated by the previously chosen cuts and which does not cross the previously chosen cuts. A pair of nodes i,j is then connected by an edge ij in T if and only if it is separated by exactly one of the cuts chosen, and d_{ij} is equal to the capacity of this cut – for details see HU (1969).

In contrast to the minimum cut problem, the following problem (even in its noncapacitated version) is $\mathcal{NP}$-complete (KARP (1972)):

(8.4.24) Maximum Cut Problem. *Given an undirected graph $G=(V,E)$ and a capacity function $c: E \to \mathbb{Q}$, find a cut of maximum capacity.*

It is customary to consider, in this case, the empty set also as a cut. Even though this problem is $\mathcal{NP}$-complete, it is worthwhile to see what the polyhedral approach gives. Let $G=(V,E)$ be an undirected graph, and let CUT(G) denote the convex hull of the incidence vectors of the cuts of G. As usual, by solving

$$\max c^T x, x \in \mathrm{CUT}(G)$$

we can solve the maximum cut problem for G. The cut polytope CUT(G) and the closely related bipartite subgraph polytope have been studied in BARAHONA, GRÖTSCHEL and MAHJOUB (1985) and BARAHONA and MAHJOUB (1986). It is easy to see that the inequalities

(8.4.25)

(i)	$0 \le x_e \le 1$	for all $e \in E$,
(ii)	$x(F)-x(C \setminus F) \le \lvert F\rvert -1$	for all circuits $C \subseteq E$ and all $F \subseteq C$ with $\lvert F\rvert$ odd

are valid for CUT(G). In fact, the inequalities (i) define facets of CUT(G), if e is not contained in a triangle, and (ii) defines a facet, for each $F \subseteq C$ with $|F|$ odd, if C has no chord. Moreover, the integral vectors satisfying (8.4.25) (i), (ii) are the vertices of CUT(G). So, $\max\{c^T x \mid x$ satisfies (8.4.25) (i), (ii), and x integer$\}$ is an integer programming formulation of the maximum cut problem. BARAHONA and MAHJOUB (1986) have shown that the solution set of (8.4.25) is equal to CUT(G) if and only if G has no subgraph contractible to K_5 (this result also follows from SEYMOUR (1981b)); hence in this case, (8.4.25) provides an LP-formulation of (8.4.24).

We now show that the separation problem for the system (8.4.25) can be solved in polynomial time for any graph $G = (V, E)$. So choose $y \in \mathbb{Q}^E$. We first check whether $0 \leq y_e \leq 1$ for all $e \in E$ by substitution. If not, a separating hyperplane is at hand. Otherwise we construct a new graph $H = (V' \cup V'', E' \cup E'' \cup E''')$ consisting of two disjoint copies $G' = (V', E')$ and $G'' = (V'', E'')$ of G and an additional edge set E''' that contains, for each $uv \in E$, the two edges $u'v''$, $u''v'$. The edges $u'v' \in E'$ and $u''v'' \in E''$ get the weight y_{uv}, while the edges $u'v'', u''v' \in E'''$ get the weight $1 - y_{uv}$. For each node $u \in V$, we calculate a shortest (with respect to the weights just defined) path in H from $u' \in V'$ to $u'' \in V''$. Such a path contains an odd number of edges of E''' and corresponds to a closed walk in G containing u. Clearly, if the shortest of these $[u', u'']$-path has length at least 1 then y satisfies (8.4.25) (ii), otherwise there exists a cycle C and a set $F \subseteq C$, $|F|$ odd such that y violates the corresponding inequality.

This separation algorithm – due to BARAHONA and MAHJOUB (1986) – yields, via the ellipsoid method, a polynomial time algorithm for the solution of linear programs over (8.4.25) for any graph, and thus, for graphs not contractible to K_5, a polynomial time algorithm for the max-cut problem. (For the special case of planar graphs, polynomial time algorithms, that are based on matching techniques (in fact, the T-join problem (8.5.17) comes up here) and planar duality, were given by ORLOVA and DORFMAN (1972) and HADLOCK (1975); and for graphs not contractible to K_5, a combinatorial polynomial algorithm, based on decomposition techniques and matching, was designed by BARAHONA (1983).) Linear programs over (8.4.25) give very good bounds for the maximum weight of a cut, as was demonstrated empirically in BARAHONA, GRÖTSCHEL, JÜNGER and REINELT (1986), where a cutting plane algorithm – using the simplex method – is described with which real-world problems (from physics and layout design) of considerable size could be solved to optimality. A (simpler) version of the separation algorithm described above can also be used to design a polynomial time (ellipsoidal) algorithm for the max-cut problem with nonnegative edge weights in weakly bipartite graphs – see GRÖTSCHEL and PULLEYBLANK (1981). This class of graphs contains the graphs not contractible to K_5, as was shown by FONLUPT, MAHJOUB and UHRY (1984).

Let us return to trees. The packing problem for spanning trees is also polynomially solvable – see EDMONDS (1968), KNUTH (1973), GREENE and MAGNANTI (1975):

(8.4.26) Spanning Tree Packing Problem. *Given an undirected graph, find as many pairwise edge-disjoint spanning trees as possible.*

Again we can formulate a weighted version:

(8.4.27) Capacitated Spanning Tree Packing Problem. *Given an undirected graph $G = (V, E)$ and a capacity function $c : E \to \mathbb{Z}_+$, find nonnegative integers $\lambda_1, \ldots, \lambda_k$ and spanning trees $T_1, \ldots, T_k$ such that, for each edge $e \in E$, the sum $\sum_{i, e \in T_i} \lambda_i$ is at most $c(e)$ and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

However, the optimum value of (8.4.27) can be less than the optimum value of the minimum cut problem (8.4.21) (it cannot be more). NASH-WILLIAMS (1961) and TUTTE (1961) derived a min-max relation corresponding to the capacitated spanning tree packing problem (8.4.27). It is not difficult to reduce the spanning tree covering problem to the spanning tree packing problem – cf. NASH-WILLIAMS (1964). These results extend to the packing of bases in a matroid – see the end of Section 10.2.

Dicuts and Dijoins

Recall that a set A' of arcs in a directed graph $D = (V, A)$ is called a **directed cut** or a **dicut** if $A' = \delta^-(V')$ for some set V' of nodes with $\emptyset \neq V' \neq V$ and $\delta^+(V') = \emptyset$. A **dicut covering**, or a **dijoin** is a set of arcs intersecting each dicut. It follows that a set A' of arcs is a dijoin if and only if the addition to D of all arcs (v, u) for (u, v) in A' makes D strongly connected (equivalently, if and only if contracting the arcs in A' makes D strongly connected). Two natural problems then are to ask for minimum dijoins and minimum dicuts.

(8.4.28) Minimum Dicut Problem. *Given a digraph $D = (V, A)$ and a capacity function $c : A \to \mathbb{Q}_+$, find a dicut of minimum capacity.*

(8.4.29) Minimum Dijoin Problem. *Given a digraph $D = (V, A)$ and a length function $l : A \to \mathbb{Q}_+$, find a dijoin of minimum length.*

LUCCHESI and YOUNGER (1978) showed that the optimum values of these problems are equal to the optimum values of the following "dual" problems.

(8.4.30) Dijoin Packing Problem. *Given a digraph $D = (V, A)$ and a capacity function $c : A \to \mathbb{Z}_+$, find nonnegative rationals $\lambda_1, \ldots, \lambda_k$ and dijoins $J_1, \ldots, J_k$ such that $\sum_{i, a \in J_i} \lambda_i \leq c(a)$ for each arc a, and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

(8.4.31) Dicut Packing Problem. *Given a digraph $D = (V, A)$ and a length function $l : A \to \mathbb{Z}_+$, find nonnegative rationals $\lambda_1, \ldots, \lambda_k$ and dicuts $C_1, \ldots C_k$ such that $\sum_{i, a \in C_i} \lambda_i \leq l(a)$ for each arc a, and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

In the framework of blocking clutters, Lucchesi and Younger's result states that the clutter of dicuts has the $\mathbb{Q}_+$-MFMC property. In terms of polyhedra it is the following result.

(8.4.32) Theorem. *Let $D = (V, A)$ be a digraph. Then the dominant of the incidence vectors of dijoins is determined by*

(i) $x_a \geq 0$ *for all* $a \in A$,

(ii) $x(C) \geq 1$ *for each dicut* $C \subseteq A$.

□

By Lehman's theorem (8.1.5) we also have an analogous polyhedral description of the dominant of dicuts by interchanging the roles of dicuts and dijoins.

Lucchesi and Younger also showed that (8.4.31), always has an integral optimum solution. As the optimum value is equal to that of (8.4.29), in a more combinatorial way it can be formulated as follows.

(8.4.33) Lucchesi-Younger Theorem. *For any digraph, the minimum cardinality of a dijoin is equal to the maximum number of pairwise disjoint dicuts.* □

This is equivalent to saying that the clutter of dicuts has the $\mathbb{Z}_+$-MFMC property, and to saying that the system in (8.4.32) is TDI. However, SCHRIJVER (1980b) showed by an example that the $\mathbb{Z}_+$-MFMC does not always hold for the clutter of dijoins. This is another example showing that Lehman's theorem cannot be extended to the $\mathbb{Z}_+$-MFMC property. But if D is acyclic and each pair of source and sink is connected by a directed path, then the clutter of dijoins has the $\mathbb{Z}_+$-MFMC property – see FEOFILOFF and YOUNGER (1985), SCHRIJVER (1982). Another such class is formed by those digraphs that can be obtained from oriented trees by adding some of the arcs (u, v) for which the tree contains a (u, v)-dipath – see SCHRIJVER (1983).

What does this mean algorithmically? The minimum dicut problem (8.4.28) can easily be solved in polynomial time, e. g., by adding an arc (v, u) to D for each arc (u, v) of D, where the new arc has high capacity. Then (8.4.28) is reduced to the minimum cut problem (8.4.16). This implies by Theorem (8.1.10) that also the other three problems (8.4.29), (8.4.30), and (8.4.31) are solvable in polynomial time. We know by the Lucchesi-Younger theorem that the dicut packing problem (8.4.31) has integral optimum solutions. Although Theorem (8.1.10) does not directly supply these, we can construct from the fractional optimum solution an integral one by applying uncrossing techniques similar to those described earlier in this section for the rooted cut packing problem (8.4.2). LUCCHESI (1976), FRANK (1981b), and A. V. Karzanov designed combinatorial polynomial algorithms for (8.4.29) and (8.4.31). It is not known whether optimum integral packings of dijoins can be found in polynomial time. In the special cases mentioned above when the clutter of dijoins has the $\mathbb{Z}_+$-MFMC property, the proofs yield polynomial time algorithms to find an optimum integral packing of dijoins.

The following results (SCHRIJVER (1982)) generalize the above results on dijoins. Let $D = (V, A)$ and $D' = (V, A')$ be digraphs, both with node set V. Call a set A'' of arcs of D a **strong connector (for D')** if the digraph $(V, A' \cup A'')$ is strongly connected. A set A'' of arcs of D is called a **strong cut (induced by D')** if there exists a set V' of nodes with $\emptyset \neq V' \neq V$, such that no arc of D' enters

V', and A'' is the set of arcs of D entering V'. Now asking for a strong cut of minimum capacity can be reduced to the minimum cut problem (8.4.16), and is hence polynomially solvable. But the problem of finding a strong connector of minimum length contains the minimum strongly connected subdigraph problem (8.4.15) as special case (by taking $A' = \emptyset$), and is hence $\mathcal{NP}$-complete.

However, in some special cases one does have a good characterization.

(8.4.34) Theorem. *Let $D' = (V, A')$ be a digraph such that D' is acyclic and each source of D' is connected to each sink of D' by a dipath. Then for each digraph $D = (V, A)$ we have the following.*

(a) *For any length function $l : A \to \mathbb{Z}_+$, the minimum length of a strong connector for D' is equal to the maximum number of (not necessarily distinct) strong cuts such that each arc a of D is contained in at most $l(a)$ of these strong cuts.*

(b) *For any capacity function $c : A \to \mathbb{Z}_+$, the minimum capacity of a strong cut is equal to the maximum number of (not necessarily distinct) strong connectors such that each arc of D is contained in at most $c(a)$ of these strong connectors.* □

This theorem means that under the conditions on D' stated above, the clutter of minimal strong connectors as well as the clutter of minimal strong cuts have the $\mathbb{Z}_+$-MFMC property. The problems and equalities contained in (8.4.34) (a), (b) include those on shortest paths, maximum flows, minimum (r, s)-cuts, minimum r-arborescences, maximum arborescence packings, minimum r-cuts, r-cut packings, edge coverings in bipartite graph, stable sets in bipartite graphs, and edge colorings in bipartite graphs.

As remarked above, the minimum strong cut problem can be solved in polynomial time, hence we obtain by the same arguments as for dijoins that the minimum strong connector problem can be solved in polynomial time for pairs of digraphs satisfying the assumptions of Theorem (8.4.34). However, to obtain integral dual solutions we refer to the special algorithm described in SCHRIJVER (1982). Schrijver's algorithm is not strongly polynomial since it uses rounding of rational numbers to integers.

The problem of minimum dijoins is related to the following. A set A' of arcs of a digraph $D = (V, A)$ is called a **feedback arc set** if each directed cycle of D contains at least one arc in A', i. e., if the digraph $(V, A \setminus A')$ is acyclic.

(8.4.35) Minimum Feedback Arc Set Problem. *Given a digraph $D = (V, A)$ and a length function $l : A \to \mathbb{Q}_+$, find a feedback arc set of minimum length.*

In general, this problem (equivalent to the **acyclic subgraph problem**, where a maximum length set of arcs containing no directed cycle is sought) is $\mathcal{NP}$-complete. However, for a planar digraph we can construct the planar dual. Then directed cycles and feedback arc sets pass into dicuts and dijoins, respectively. Hence the minimum feedback arc set problem passes into the minimum dijoin problem, which is polynomially solvable. The Lucchesi-Younger theorem (8.4.32) then gives the optimum value of the following problem.

(8.4.36) Directed Cycle Packing Problem. *Given a digraph $D = (V, A)$ and a capacity function $c : A \to \mathbb{Z}_+$, find nonnegative integers $\lambda_1, \ldots, \lambda_k$ and dicycles $C_1, \ldots C_k$ such that $\sum_{i, a \in C_i} \lambda_i \leq c(a)$ for each arc a and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

For general digraphs this problem is $\mathcal{NP}$-complete (also if we would skip the condition of λ_i being integer).

The Lucchesi-Younger theorem thus gives as a corollary:

(8.4.37) Theorem. *In a planar digraph, the minimum cardinality of a feedback arc set is equal to the maximum number of pairwise arc-disjoint directed cycles.* □

Again, one easily derives a weighted version from this theorem. Polyhedrally this can be stated as follows. Consider the following system of inequalities for a digraph $D = (V, A)$:

(8.4.38)
$$\begin{array}{lll} \text{(i)} & x_a \geq 0 & \text{for each } a \in A, \\ \text{(ii)} & x(C) \geq 1 & \text{for each dicycle } C \subseteq A. \end{array}$$

(8.4.39) Corollary. *If D is planar, the dominant of the incidence vectors of the feedback arc sets of D is given by* (8.4.38) (i), (ii). □

Above, planarity is a sufficient but not a necessary condition. E. g., the corollary holds trivially for all acyclic digraphs, and one can glue together planar and acyclic digraphs in certain ways to get less trivial examples – for more detail see for instance GRÖTSCHEL, JÜNGER and REINELT (1985) and BARAHONA and MAHJOUB (1985).

Since we can find a shortest dicycle in polynomial time, the separation problem for the solution set of (8.4.38) can be solved in polynomial time. Hence for all digraphs for which the dominant of the incidence vectors of the feedback arc sets is given by (8.4.38), a minimum feedback arc set can be found in polynomial time. (These and further observations have been used in GRÖTSCHEL, JÜNGER and REINELT (1984) to design an empirically efficient cutting plane algorithm for the closely related ($\mathcal{NP}$-hard) linear ordering problem with which large practical problems can be solved to optimality.) Blocking theory, in particular Theorem (8.1.5), implies that for these digraphs we also have a description of the dominant of the incidence vectors of dicycles – see the remarks after (8.3.6).

*8.5 Matchings, Odd Cuts, and Generalizations

Flows, paths, trees, and branchings represent one stream of generalization of the problems and results on bipartite graphs mentioned in Section 8.2. There is another important stream of generalization, namely to matchings in arbitrary, not necessarily bipartite, graphs. A preview of these results was given in Section 7.3. Now we go into the subject in greater depth.

TUTTE (1947) laid the foundation of optimum matching theory for nonbipartite graphs by giving a good characterization for the existence problem for perfect matchings – see Theorem (7.3.2). Further research in matching theory followed – cf. TUTTE (1952, 1954), GALLAI (1950, 1963, 1964), ORE (1957, 1959), BERGE (1958). The papers of EDMONDS (1965a, 1965b) brought the breakthrough in the algorithmic side of matching, giving a polynomial algorithm for the maximum (weighted) matching problem and an explicit linear characterization of the polyhedra related to matching problems, thus describing the matching problem as a linear program – cf. Theorem (7.3.5). Although these programs have an exponential number of constraints, Edmonds was able to solve them in polynomial time by using structural properties of the underlying combinatorial problem.

Matching

Let us start with one of the most simple, but basic problems.

(8.5.1) (Cardinality) Matching Problem. *Given an undirected graph, find a matching of maximum cardinality.*

BERGE (1958) derived the following result from Tutte's characterization (7.3.2) of graphs with perfect matchings.

(8.5.2) Tutte-Berge Theorem. *For any undirected graph $G = (V, E)$, the maximum size $\nu(G)$ of a matching is equal to*

$$\min_{V' \subseteq V} \frac{|V| + |V'| - \text{odd}(V \setminus V')}{2}$$

where $\text{odd}(V \setminus V')$ *denotes the number of components in the subgraph* $G - V'$ *of* G *with an odd number of nodes.* □

The Tutte-Berge theorem shows that the matching problem is in $\mathcal{NP} \cap \text{co-}\mathcal{NP}$. EDMONDS (1965a) was the first to give a polynomial algorithm to solve (8.5.1), thereby showing that it is in $\mathcal{P}$. More recently, EVEN and KARIV (1975) and MICALI and VAZIRANI (1980) published improvements yielding an algorithm of complexity $|V|^{1/2}|E|$. Note that Edmonds' result also provides a polynomial test for a graph to have a perfect matching. Since Gallai's identities (8.2.2) have a polynomially constructive proof, it also implies the polynomial solvability of the minimum edge covering problem. With Gallai's identities a similar min-max equality for the edge covering number $\rho(G)$ can be derived from the Tutte-Berge theorem.

The weighted matching problem is also polynomially solvable, as was shown by EDMONDS (1965b).

(8.5.3) Weighted Matching Problem. *Given an undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{Q}_+$, find a matching of maximum weight.*

Edmonds obtained, as a by-product, a good characterization of this problem in terms of polyhedra. The **matching polytope** of $G = (V, E)$ is the convex hull of the incidence vectors of matchings in G.

(8.5.4) Theorem. *The matching polytope is determined by the following inequalities:*

$$\begin{aligned} &\text{(i)} && x(e) \geq 0 && \text{for all } e \in E, \\ &\text{(ii)} && x(\delta(v)) \leq 1 && \text{for all } v \in V, \\ &\text{(iii)} && x(E(W)) \leq \lfloor \tfrac{1}{2}|W| \rfloor && \text{for all } W \subseteq V, |W| \text{ odd.} \end{aligned}$$

□

It is not difficult to see that this theorem is equivalent to Edmonds' perfect matching polytope theorem (7.3.5).

This indeed provides a good characterization for the weighted matching problem, as now it can be described as a linear program with constraints (8.5.4) (i), (ii), (iii), and hence the duality theorem of linear programming yields a good characterization. We should note here that, although there are exponentially many constraints in (8.5.4), an optimum dual solution can be described by specifying the dual variables belonging to at most $|E|$ of the constraints – the other dual variables are zero.

The Tutte-Berge theorem (8.5.2) is equivalent to saying that the dual of the linear program of maximizing $\mathbb{1}^T x$ over (8.5.4) (i), (ii), (iii) has an integral optimum solution. CUNNINGHAM and MARSH (1979) generalized this by showing that for any integral weight function w, the dual program has an integral optimum solution, and they gave a polynomial time method to find this solution.

Using elementary constructions it is easy to see that the weighted matching problem (8.5.3) is equivalent to the:

(8.5.5) Weighted Perfect Matching Problem. *Given an undirected graph $G = (V, E)$ with $|V|$ even, and a weight function $w : E \to \mathbb{Q}$, find a perfect matching of maximum weight.*

Since weights may be negative, this includes the minimum weighted perfect matching problem.

As mentioned, the characterization (8.5.4) of the matching polytope is equivalent to the characterization of the perfect matching polytope (7.3.5). Recall that the perfect matching polytope is determined by the inequalities:

$$\begin{aligned} (8.5.6) \qquad x(e) &\geq 0 \quad \text{for all } e \in E, \\ x(\delta(v)) &= 1 \quad \text{for all } v \in V, \\ x(\delta(W)) &\geq 1 \quad \text{for all } W \subseteq V, |W| \text{ odd.} \end{aligned}$$

Also recall that a set of edges of the form $\delta(W)$ with $|W|$ odd is called an **odd cut**. Describing the weighted perfect matching problem as a linear program with constraints (8.5.6) will give as dual program a packing problem of odd cuts. From the above mentioned results of Cunningham and Marsh it follows that, for an integer weight function w, the dual program has a half-integer optimum solution, and that this solution can be found in polynomial time.

So, following the lines of the previous sections, we formulate a polar to the weighted matching problem:

(8.5.7) Minimum Odd Cut Problem. *Given an undirected graph $G = (V, E)$ with $|V|$ even, and a capacity function $c : E \to \mathbb{Q}_+$, find an odd cut of minimum capacity.*

PADBERG and RAO (1982) gave a polynomial algorithm for this problem, using the Gomory-Hu method for the minimum cut problem (8.4.21) – see Section 7.3 and (8.4.23).

b-Matching

We next treat some further, more general problems. If $G = (V, E)$ is an undirected graph, and $b : V \to \mathbb{Z}_+$, then a **b-matching** is a function $x : E \to \mathbb{Z}_+$ such that $x(\delta(v)) \leq b(v)$ for each node v. The b-matching is **perfect** if we have equality for all v. So (perfect) 1-matchings are just the incidence vectors of (perfect) matchings.

(8.5.8) Weighted (Perfect) b-Matching Problem. *Given an undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{Q}$, and a function $b : V \to \mathbb{Z}_+$, find a (perfect) b-matching x with $\sum_{e \in E} w(e)x(e)$ as large as possible.*

So this problem reduces to the weighted (perfect) matching problem if $b = \mathbb{1}$. The weighted b-matching problem could be reduced to the weighted matching problem by replacing each node v by a stable set of $b(v)$ nodes, where two of the new nodes are adjacent iff their originals are adjacent. In this way one obtains from (8.5.4) that the **b-matching polytope** (i. e., the convex hull of the b-matchings), is given by:

$$\begin{aligned} x(e) &\geq 0 && \text{for all } e \in E, \\ x(\delta(v)) &\leq b(v) && \text{for all } v \in V, \\ x(E(W)) &\leq \lfloor \tfrac{1}{2} b(W) \rfloor && \text{for all } W \subseteq V,\ b(W) \text{ odd.} \end{aligned} \tag{8.5.9}$$

Hence also for the weighted b-matching problem a good characterization exists. PULLEYBLANK (1980, 1981) showed that for integral weight functions w, the linear program corresponding to (8.5.8) (with constraints (8.5.9)) has an integral optimum dual solution.

However, splitting a node v into $b(v)$ new nodes is, in general, not a polynomial time procedure (as $b(v)$ is exponential in the encoding length of $b(v)$). So Edmonds' weighted matching algorithm cannot be extended straightforwardly in this way. Yet, the weighted b-matching problem is polynomially solvable, as was shown by Cunningham and Marsh (see MARSH (1979)) – see also EDMONDS and JOHNSON (1970), PULLEYBLANK (1973), and PADBERG and RAO (1982).

Again, the b-matching problem is trivially polynomially equivalent to the perfect b-matching problem. Moreover, from (8.5.9) it follows that the **perfect b-matching polytope** (i. e., the convex hull of the perfect b-matchings) is given by the linear constraints:

$$\begin{aligned} x(e) &\geq 0 && \text{for all } e \in E, \\ x(\delta(v)) &= b(v) && \text{for all } v \in V, \\ x(\delta(W)) &\geq 1 && \text{for all } W \subseteq V,\ b(W) \text{ odd.} \end{aligned} \tag{8.5.10}$$

As a dual problem to the b-matching problems one could take the following.

(8.5.11) Minimum Weighted T-Cut Problem. *Given an undirected graph $G = (V, E)$, $T \subseteq V$ with $|T|$ even, and a capacity function $c : E \to \mathbb{Q}_+$, find a T-cut of minimum capacity.*

This indeed contains the separation problem for the perfect b-matching polytope, by taking $T := \{v \mid b(v) \text{ odd}\}$. (If $|T|$ is odd there are no perfect b-matchings.) PADBERG and RAO (1982) showed that the minimum T-cut problem is polynomially solvable. This algorithm was described in Section 7.3. Hence by the ellipsoid method, the (perfect) b-matching problem is solvable in polynomial time.

Often one is interested in b-matchings in which a certain edge may not be used more than once (in which case we just get a subgraph problem), or more than a given number of times. If $G = (V, E)$ is an undirected graph, and $c : E \to \mathbb{Z}_+$, then a b-matching x is called **c-capacitated** if $x(e) \leq c(e)$ for each edge e of G.

(8.5.12) Capacitated Weighted (Perfect) b-Matching Problem. *Given an undirected graph $G = (V, E)$, a function $b : V \to \mathbb{Z}_+$, a capacity function $c : E \to \mathbb{Z}_+$, and a function $w : E \to \mathbb{Q}_+$ find a c-capacitated (perfect) b-matching x such that $\sum_{e \in E} w(e)x(e)$ is as large as possible.*

Cunningham and Marsh (see MARSH (1979)), EDMONDS and JOHNSON (1970), and PULLEYBLANK (1973) gave polynomial algorithms for this problem. There is a way of reducing (8.5.12) to the weighted perfect b-matching problem. First, the capacitated weighted b-matching problem can be reduced to the capacitated weighted perfect b-matching problem – this is easy, by taking a disjoint copy G' of G, and joining a node in G to its copy in G' by an edge of high capacity and zero weight. Next, the capacitated weighted perfect b-matching problem can be reduced to the (uncapacitated) weighted perfect b-matching problem, by replacing each edge $e = \{u, v\}$ of G by three edges $\{u, u_e\}$, $\{u_e, v_e\}$, $\{v_e, v\}$ in series (where for each edge e, u_e and v_e are two new nodes of degree two), with weights $w(e)$, 0, $w(e)$, respectively. If we define $b(u_e) := b(v_e) := c(e)$, the perfect b-matching problem for the new graph is equivalent to the capacitated perfect b-matching problem in the original graph – see TUTTE (1952).

The same construction yields that the following inequalities define the c-capacitated perfect b-matching polytope:

$$
\begin{aligned}
(8.5.13) \qquad 0 \leq x(e) &\leq c(e) && \text{for all } e \in E,\\
x(\delta(v)) &= b(v) && \text{for all } v \in V,\\
x(\delta(W) \setminus F) + c(F) - x(F) &\geq 1 && \text{for all } W \subseteq V \text{ and } F \subseteq \delta(W)\\
& && \text{with } b(W) + c(F) \text{ odd.}
\end{aligned}
$$

Similarly, the c-capacitated b-matching polytope is given by:

$$
\begin{aligned}
(8.5.14) \qquad 0 \leq x(e) &\leq c(e) && \text{for all } e \in E,\\
x(\delta(v)) &\leq b(v) && \text{for all } v \in V,\\
x(E(W)) + x(F) &\leq \lfloor \tfrac{1}{2}(b(W) + c(F)) \rfloor && \text{for all } W \subseteq V \text{ and } F \subseteq \delta(W)\\
& && \text{with } b(W) + c(F) \text{ odd.}
\end{aligned}
$$

By taking $c \equiv \infty$ (or very large), the characterizations (8.5.9) and (8.5.10) follow. Descriptions (8.5.13) and (8.5.14) give good characterizations for the capacitated weighted (perfect) b-matching problem.

PADBERG and RAO (1982) proved that the separation problem for the capacitated b-matching polytope is polynomially solvable. Their algorithm is based on the same reduction to a T-cut problem as outlined above.

Also the problem of finding a minimum weighted edge covering can be reduced by the two constructions given above directly to the weighted matching problem. Indeed, the weighted edge covering problem is equivalent to the maximum weighted 1-capacitated b-matching problem, with $b(v)$ equal to one less than the degree of node v (so each node can be split into $b(v)$ copies in polynomial time). Moreover, this yields the following characterization of the **edge covering polytope** (i. e., the convex hull of the incidence vectors of edge coverings):

$$\begin{aligned} (8.5.15) \qquad 0 \le x(e) &\le 1 && \text{for all } e \in E, \\ x(E(W) \cup \delta(W)) &\ge \lceil \tfrac{1}{2}|W| \rceil && \text{for all } W \subseteq V \text{ with } |W| \text{ odd.} \end{aligned}$$

One of the most general problems in this direction (comparable with (8.2.13) and (8.3.28)) is the following:

(8.5.16) Problem. *Given an undirected graph $G = (V, E)$, and functions $b_1, b_2 : V \to \mathbb{Z}$, $c_1, c_2 : E \to \mathbb{Z}$, and $w : E \to \mathbb{Q}$ find a function $x : E \to \mathbb{Z}$ such that $c_1(e) \le x(e) \le c_2(e)$ for each edge e of G and $b_1(v) \le x(\delta(v)) \le b_2(v)$ for each node v of G, and such that $\sum_{e \in E} w(e)x(e)$ is as large as possible.*

It is not difficult to reduce this problem to the case with $c_1 \equiv 0$ and $b_1 \ge 0$. By some further elementary constructions, this problem can be reduced in polynomial time to the capacitated b-matching problem, and is hence polynomially solvable. Also the corresponding linear inequalities can be found in this way – cf. LOVÁSZ (1970), TUTTE (1974, 1978), SCHRIJVER and SEYMOUR (1977). For a survey of capacitated and uncapacitated b-matching polyhedra (characterizing also facets and minimal TDI-systems) see COOK and PULLEYBLANK (1987).

T-Joins and T-Cuts

In Section 7.3 we found it handy to extend the notion of odd cuts and considered T-cuts. Matching theory yields several min-max relations and polyhedra corresponding to this more general class of cuts.

Let there be given an undirected graph $G = (V, E)$ and a subset T of V with $|T|$ even. A set F of edges is called a **T-join** if T coincides with the set of nodes of odd degree in the graph $G' = (V, F)$. Recall that a **T-cut** is a set of edges of the form $\delta(W)$, where W is a set of nodes with $|T \cap W|$ odd.

T-joins and T-cuts have turned out to form a suitable generalization of many interesting special cases. For instance, if $T = \{r, s\}$, then (inclusionwise) minimal T-joins are just paths from r to s. If $T = V$, the T-joins of size $\frac{1}{2}|V|$ are the perfect matchings. Later on we shall see that also the Chinese postman problem fits into this framework.

It is not difficult to show that the minimal T-cuts are exactly the minimal sets of edges intersecting all T-joins, and conversely, minimal T-joins are exactly the minimal sets of edges intersecting all T-cuts, i. e., minimal T-cuts and minimal T-joins form blockers of each other. Note that each T-join is the disjoint union of $\frac{1}{2}|T|$ paths, each connecting two nodes in T, and a number of circuits.

We formulated the minimum T-cut problem in (8.5.11). Now we also study its counterpart.

(8.5.17) Minimum Weighted T-Join Problem. *Given an undirected graph $G = (V, E)$, a subset T of V with $|T|$ even, and a length function $l : E \to \mathbb{Q}$, find a T-join of minimum length.*

This problem contains as special cases several problems discussed before. If $T = V$ problem (8.5.17) contains the weighted perfect matching problem (8.5.5). If $T = \{r, s\}$ for nodes r and s of G, and each circuit of G has nonnegative length, problem (8.5.17) reduces to finding an $[r, s]$-path of minimum length. If $T = \emptyset$, problem (8.5.17) is equivalent to finding a subgraph with all degrees even, with minimum length.

EDMONDS and JOHNSON (1973) showed that the minimum T-join problem is polynomially solvable, by reducing it to the weighted perfect matching problem. A simple reduction to describe is splitting each node $v \in V$ into a clique of size t_v, where $t_v = \deg(v)$ or $t_v = \deg(v) + 1$ such that t_v is odd if and only if $v \in T$. From a complexity point of view the following reduction turns out to be more satisfactory. First let $E' := \{e \in E \mid l_e < 0\}$. Then for all $e \in E$ replace the edge length l_e by $l'_e := |l_e|$ and replace T by $T' := T \triangle \{v \in V \mid v \text{ is incident with an odd number of edges in } E'\}$. (Here $\triangle$ denotes the symmetric difference.) Then J is a minimum length T-join with respect to l if and only if $J \triangle E'$ is a minimum length T'-join with respect to l'. In this way the minimum T-join problem is reduced to the case with nonnegative edge length. So we may assume $l \geq 0$. Next for each pair $r, s \in T$ we determine a shortest $[r, s]$-path P_{rs} of length λ_{rs} (say). Consider the complete graph on T with edge weights λ_{rs} and find a minimum weight perfect matching in this graph, say $r_1s_1, \ldots, r_ks_k$. Then $J := P_{r_1s_1} \triangle \ldots \triangle P_{r_ks_k}$ is a minimum length T-join in the original graph G.

The problem dual to the T-join problem is the following.

(8.5.18) T-Cut Packing Problem. *Given an undirected graph $G = (V, E)$, a subset T of V with $|T|$ even, and a length function $l : E \to \mathbb{Z}_+$, find T-cuts $E_1, \ldots, E_k$ and nonnegative rationals $\lambda_1, \ldots, \lambda_k$ such that $\sum_{e \in E_i} \lambda_i \leq l(e)$ for each edge $e \in E$, and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

And similarly, the dual to the T-cut problem is:

(8.5.19) T-Join Packing Problem. *Given an undirected graph $G = (V, E)$, a subset T of V with $|T|$ even, a capacity function $c : E \to \mathbb{Z}_+$, find T-joins $J_1, \ldots, J_k$ and nonnegative rationals $\lambda_1, \ldots, \lambda_k$ such that $\sum_{e \in J_i} \lambda_i \leq c(e)$ for each edge $e \in E$, and such that $\lambda_1 + \ldots + \lambda_k$ is as large as possible.*

If $|T| = 2$, then the T-join packing problem contains the maximum flow problem (8.3.7) for undirected graphs – see (8.3.9) – while, if $T = V$, then it includes the fractional edge coloring problem – see Section 7.4.

Edmonds and Johnson showed that if l is nonnegative the optimum value of (8.5.17) is equal to the optimum value of (8.5.18). It can be derived from their proof – cf. LOVÁSZ (1975) – that the optimum λ_i can be taken to be half-integral. SEYMOUR (1981a) showed that there are integer optimum λ_i if for each circuit of G the sum of its edge lengths is even. From this the half-integrality result follows by multiplying l by 2. Seymour's results can be stated combinatorially as follows.

(8.5.20) Theorem. *If G is a bipartite graph and T is a set of nodes of G of even size, then the minimum size of a T-join is equal to the maximum number of pairwise disjoint T-cuts.* □

The graph $G = K_4$ shows that we may not delete "bipartite", but there are other classes of graphs, like series-parallel graphs, for which a similar theorem holds – see SEYMOUR (1977).

The result of Edmonds and Johnson implies that the clutter of T-cuts has the $\mathbb{Q}_+$-MFMC property. Hence the dominant of the incidence vectors of the T-joins of G is given by the linear inequalities

$$\begin{aligned} x_e &\geq 0 \quad \text{for all } e \in E, \\ x(F) &\geq 1 \quad \text{for all } T\text{-cuts } F \subseteq E. \end{aligned} \tag{8.5.21}$$

Moreover, the convex hull of the incidence vectors of T-joins is given by

$$\begin{aligned} 0 \leq x_e &\leq 1 \quad \text{for all } e \in E, \\ x(\delta(W) \setminus F) + |F| - x(F) &\geq 1 \quad \text{for all } W \subseteq V \text{ and } F \subseteq \delta(W) \\ &\qquad \text{with } |W \cap T| + |F| \text{ odd}. \end{aligned} \tag{8.5.22}$$

By (8.1.5) we obtain from (8.5.21) that the clutter of T-joins also has the $\mathbb{Q}_+$-MFMC property. Hence the dominant of the incidence vectors of T-cuts is given by

$$\begin{aligned} x_e &\geq 0 \quad \text{for all } e \in E, \\ x(J) &\geq 1 \quad \text{for all } T\text{-joins } J \subseteq F. \end{aligned} \tag{8.5.23}$$

Both polyhedral results (8.5.21) and (8.5.22) can be used to show the polynomial time solvability of the T-join problem. In fact, the separation problem for (8.5.21) is just the minimum T-cut problem, and hence we can optimize in polynomial time over (8.5.21), which solves the T-join problem for nonnegative edge lengths. We saw above that this implies that we can solve the minimum T-join problem for any length function. Hence the optimization and also the separation problem for (8.5.22) is solvable in polynomial time. The separation problem for (8.5.22) can be directly reduced to a T'-cut problem on a larger graph.

By Theorem (8.1.10), also the T-cut packing problem can be solved in polynomial time. To find an integral optimum solution to (8.5.18) is $\mathcal{NP}$-complete. However, half-integral optimum solutions can be found in polynomial time. This again follows from the fact that if every cycle has even length with respect to l then optimum integral solutions can be found in polynomial time. Like for packing rooted cuts and dicuts, the algorithm is based on uncrossing techniques, but it is more involved, as in the case of T-cuts we cannot appeal to total unimodularity. But from LOVÁSZ (1975) and SEYMOUR (1979) a polynomial time algorithm can be derived.

By blocking theory, Theorem (8.4.20) implies that the optimum value of the weighted T-cut problem (8.5.11) is equivalent to the optimum value of the T-join packing problem (8.5.19). If $|T| = 2$ this extends the (fractional version of the) max-flow min-cut theorem (8.3.9) for undirected graphs.

A (fractional) optimum solution to the T-join packing problem (8.5.19) can be obtained in polynomial time by Theorem (8.1.10). As the integer version of this problem contains the edge coloring problem, it is $\mathcal{NP}$-complete. If $G = (V, E)$ is a planar graph and $T \subseteq V$ is a set of nodes such that all nodes are contained in a cycle bounding one face of a plane embedding of G, then Barahona and Conforti (personal communication) showed that the minimum weight of a T-cut of G equals the maximum weight of an integral T-join packing of G. So in this case the clutter of T-joins has the $\mathbb{Z}_+$-MFMC-property.

Chinese Postmen and Traveling Salesmen

Two optimization problems which are important in practice and related to matching theory are the following.

(8.5.24) Chinese Postman Problem. *Given a connected undirected graph $G = (V, E)$ and a length function $l : E \to \mathbb{Q}$, find a closed walk in G such that every edge is traversed at least once and such that the total length is as small as possible.*

(8.5.25) (Symmetric) Traveling Salesman Problem. *Given an undirected graph $G = (V, E)$ and a length function $l : E \to \mathbb{Q}$, find a Hamiltonian cycle in G with minimum length.*

The Chinese postman problem was formulated by MEI-KO KWAN (1960), and EDMONDS and JOHNSON (1973) gave a polynomial time algorithm, by the following easy reduction to the minimum weighted T-join problem. One first finds a minimum length T-join where T is the set of nodes with odd degree. If we duplicate in G the edges of this T-join we obtain an Eulerian graph G'. Any Eulerian trail in G' gives an optimum Chinese postman walk.

The Chinese postman problem is also solvable in polynomial time for directed graphs (with essentially the same idea). But for mixed graphs (i. e., graphs with both directed and undirected edges) the Chinese postman problem is $\mathcal{NP}$-complete– see PAPADIMITRIOU (1976).

The traveling salesman problem is $\mathcal{NP}$-complete, and therefore it is unlikely that a complete linear description of the associated polytope can be obtained explicitly. But the polyhedral approach was quite successful with respect to designing practically efficient cutting plane algorithms for the solution of traveling salesman problems – see PADBERG and GRÖTSCHEL (1985) for these algorithmic aspects. The theoretical investigations of the traveling salesman polytope

$$\mathrm{TSP}(G) := \mathrm{conv}\{\chi^H \in \mathbb{R}^E \mid H \subseteq E \text{ a Hamiltonian cycle (tour) in } G\}$$

have been summarized in GRÖTSCHEL and PADBERG (1985). We outline here some of the results.

Note that every tour in a graph passes through every node exactly once, thus every incidence vector of a tour satisfies

$$\begin{aligned} 0 \leq x_e \leq 1 \quad & \text{for all } e \in E, \\ x(\delta(v)) = 2 \quad & \text{for all } v \in V. \end{aligned} \tag{8.5.26}$$

The integral solutions of (8.5.26) are the incidence vectors of the 1-capacitated perfect 2-matchings contained in G, in other words, the incidence vectors of the disjoint unions of circuits such that every node of G is on exactly one circuit. But the polytope defined by (8.5.26) also has fractional vertices. From (8.5.13) we know how to cut these off. By adding the inequality system

$$x(E(W)) + x(T) \leq |W| + \frac{1}{2}(|T| - 1) \quad \text{for all } W \subseteq V \text{ and all } T \subseteq \delta(W) \text{ with } |T| \text{ odd} \tag{8.5.27}$$

to (8.5.26) we obtain the convex hull of the 1-capacitated perfect 2-matchings of G. In order to get the traveling salesman polytope, the 2-matchings, that are not tours, have to be cut off. It is obvious that every such 2-matching violates one of the following **subtour elimination constraints**

$$x(\delta(W)) \geq 2 \quad \text{for all } W \subseteq V \text{ with } 2 \leq |W| \leq |V| - 2. \tag{8.5.28}$$

The integral points in the polytope $P(G)$ defined by (8.5.26), (8.5.27), and (8.5.28) are exactly the incidence vectors of tours. However, $P(G)$ has further fractional vertices. What is important about $P(G)$ is that linear programs over $P(G)$ can be solved in polynomial time and that the traveling salesman problem can be viewed as the integer linear program $\min\{c^T x \mid x \in P(G) \text{ and } x \text{ integral}\}$.

To see the polynomial time solvability of $\min\{c^T x \mid x \in P(G)\}$ observe first that for any $y \in \mathbb{Q}^E$ the system (8.5.26) can be checked by substitution. So we may assume that y satisfies (8.5.26). The separation problem for the 2-matching inequalities (8.5.27) is a T-cut problem (as described earlier) and so it can be solved in polynomial time. (In fact, the design of an efficient algorithm for the separation problem for (8.5.27), to be used in a cutting plane algorithm for the traveling salesman problem, was the motivation for the paper PADBERG and RAO (1982), which solved the T-cut problem and the separation problem for

general matching polyhedra. So a computational problem was the stimulus for this pre-ellipsoid separation algorithm.)

To solve the separation problem for (8.5.28), we consider the values y_e, $e \in E$, as capacities on the edges of G. Then we solve the minimum cut problem (8.4.21) for this graph. If the minimum capacity $y(\delta(W^*))$ of a cut is smaller than 2, then $x(\delta(W^*)) \geq 2$ is a violated subtour elimination constraint; otherwise y satisfies all inequalities (8.5.28). So again, the separation problem can be reduced to a cut problem that is solvable in polynomial time. Since we can solve the separation problems for (8.5.26), (8.5.27), and (8.5.28) in polynomial time, we can optimize over $P(G)$ in polynomial time.

There is a further "handy looking" class of inequalities, valid for TSP(G), which can be defined as follows. A **clique tree** is a connected graph C for which the maximal cliques satisfy the following properties:

- The cliques are partitioned into two sets, the set of **handles** $H_1, \ldots, H_r$ and the set of **teeth** $T_1, \ldots, T_s$.
- No two teeth intersect.
- No two handles intersect.
- Each tooth contains at least two nodes, at most $n-2$ nodes, and at least one node belonging to no handle.
- For each handle, the number of teeth intersecting it is odd and at least three.
- If a tooth T and a handle H have a nonempty intersection, then $C-(H \cap T)$ is disconnected.

Figure 8.1 shows a clique tree with 19 teeth and 7 handles schematically. The ellipses containing a star correspond to teeth, the other ellipses to handles.

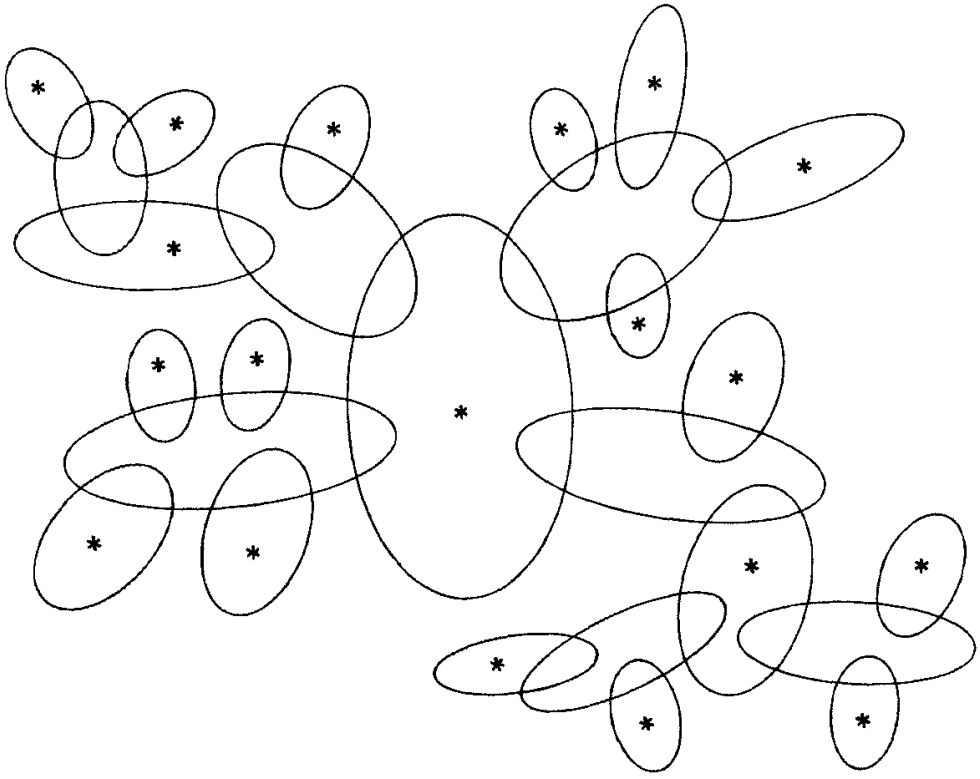


Figure 8.1

GRÖTSCHEL and PULLEYBLANK (1986) showed that for each clique tree (with handles $H_1, \ldots, H_r$ and teeth $T_1, \ldots, T_s$) the following **clique tree inequality**

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j)) \leq \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{s} (|T_j| - t_j) - \frac{s+1}{2} \tag{8.5.29}$$

defines a facet of TSP(G), if G is a complete graph. In (8.5.29), for every tooth T_j, the integer t_j denotes the number of handles that intersect T_j, $j = 1, \ldots, s$. At present it is unknown whether the separation problem for (8.5.29) can be solved in polynomial time. (The only solved special case is the class of 2-matching inequalities (8.5.27).) In PADBERG and GRÖTSCHEL (1985) some heuristics are described that check whether a point satisfies some of the inequalities (8.5.29) or not and that have been used in cutting plane procedures.

Several further classes of facet defining inequalities, much more complicated than the ones given above, are known. It follows, however, from a result of KARP and PAPADIMITRIOU (1980) that, if $\mathcal{NP} \neq$ co-$\mathcal{NP}$, there is no way to describe a list of inequalities sufficient to define TSP(G) – cf. also Section 2.2.

On the other hand, it has turned out that viewing the traveling salesman problem polyhedrally and using the inequalities described above in a simplex method based cutting plane algorithm (with final branch and bound phase), gives an empirically quite efficient procedure to solve this problem – see, for instance, CROWDER and PADBERG (1980) and PADBERG and RINALDI (1987). In the latter paper, a traveling salesman problem with 2392 cities (and thus an integer linear program with almost 3 million variables) has been solved to optimality. Such results demonstrate the computational success of this approach.

Moreover, the study of problems (in particular matching problems) related to the traveling salesman problem has resulted in a number of further interesting results. One instance is formed by the 2-matchings without small circuits. Note that the convex hull of the incidence vectors of perfect 2-matchings (no capacities) is given by the system (8.5.26) without the upper bounds $x_e \leq 1$, $e \in E$. The vertices of this polytope have coefficients that are 0, 1, or 2. The edges of value 2 form a 1-matching, the edges of value 1 disjoint circuits of odd length. CORNUÉJOLS and PULLEYBLANK (1980) have given a polynomial time algorithm for finding maximum weight 2-matchings without triangles, and they showed that, by adding the inequalities $x(E(W)) \leq 2$ for all $W \subseteq V$ with $|W| = 3$ to the system, the convex hull of perfect 2-matchings without triangles is obtained. Since the number of constraints in this case is polynomial in $|E|$, we can find a minimum perfect 2-matching without triangles in polynomial time through an explicit linear program by Khachiyan's method. It is interesting to note that by excluding circuits of length at most 5 or 7 or 9 etc. the problem becomes $\mathcal{NP}$-complete (C. H. Papadimitriou – see CORNUÉJOLS and PULLEYBLANK (1980)).

HARTVIGSEN (1984) designed a polynomial time algorithm for finding a maximum weight $\mathbb{1}$-capacitated 2-matching without triangles (this is an interesting relaxation of the TSP). A system of inequalities describing the convex hull of $\mathbb{1}$-capacitated 2-matchings without triangles is not known at present. Hartvigsen conjectures that for any graph $G = (V, E)$ such a system is given by $0 \leq x_e \leq 1$ for all $e \in E$, $x(\delta(v)) \leq 2$ for all $v \in V$, together with certain clique tree inequalities (8.5.29) with teeth of size two or three.

*8.6 Multicommodity Flows

In Section 8.3 we studied problems involving the flow of a single "commodity". That is, only one pair of nodes was joined by the flow. Obviously, many real world problems involve multicommodity flows, i. e., flows of multiple, differentiated commodities. For instance, in a telephone network a great number of connections must be handled simultaneously.

To be more precise, in this section we study problems on flows and paths each of which connects one of a given collection of node-pairs $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$. This differs from the approach in Section 8.4, where we considered problems on connecting all given node-pairs by one set of arcs or edges. As blocking problems we shall meet here problems on cuts separating all given pairs, whereas in Section 8.4 the blocking problems involved cuts separating at least one of the given pairs.

Let $G = (V, E)$ be an undirected graph, and let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be pairs of nodes of G. These pairs are called the **commodities**. Then a **multicommodity flow** is given by an orientation of G for each $i = 1, \ldots, k$, forming the digraph (V, A_i), together with an (r_i, s_i)-flow x_i in (V, A_i). The ***i*-th value** of this multicommodity flow is the value of the flow x_i, for $i = 1, \ldots, k$, and the **total value** is the sum of these values. If a capacity function $c : E \to \mathbb{Q}_+$ is given, then the multicommodity flow is said to be **subject to** c if for each edge e of G, the sum of the flows through e (in both directions) does not exceed $c(e)$.

For the problems to be dealt with below, it is convenient to have an alternative, but in a sense equivalent, representation of multicommodity flows. Let $G = (V, E)$ be an undirected graph, and let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be (different) pairs of nodes of G, called the **commodities**. Then a **multicommodity flow** is given by paths $P_1, \ldots, P_s$ and nonnegative rational numbers $\lambda_1, \ldots, \lambda_s$ such that for each $j = 1, \ldots, s$, P_j is an $[r_i, s_i]$-path for some $i = 1, \ldots, k$. For $i = 1, \ldots, k$, the ***i*-th value** of the multicommodity flow is equal to the sum of those λ_j for which P_j is an $[r_i, s_i]$-path. The **total value** of the multicommodity flow is the sum of these values, i. e., is equal to $\lambda_1 + \ldots + \lambda_s$. If a capacity function $c : E \to \mathbb{Q}_+$ is given, the multicommodity flow is said to be **subject to** c if, for each edge e of G, the sum of the λ_j for which P_j contains e is at most $c(e)$.

It is easy to see that both representations of a multicommodity flow are equivalent to the following extent: given a multicommodity flow in one representation one can find in polynomial time a multicommodity flow in the other representation, with the same values, and subject to the same capacity functions. Moreover, if the x_i in the first representation are (half)-integer, one can take the λ_j in the second representation (half)-integer, and conversely.

Now consider the following problem.

(8.6.1) Multicommodity Flow Problem. *Given an undirected graph $G = (V, E)$, a collection of commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, and a capacity function $c : E \to \mathbb{Q}_+$,*

(a) *find a multicommodity flow subject to c with maximum total value,*

(b) *given in addition "demands" $\rho_1, \ldots, \rho_k$, find, if possible, a multicommodity flow subject to c for which the i-th value is equal to ρ_i, for $i = 1, \ldots, k$.*

This problem is solvable in polynomial time, as it can be formulated as a linear program. For every edge $uv \in E$, we assign variables $x^1_{uv}, x^2_{uv}, \ldots, x^k_{uv}$ and $x^1_{vu}, x^2_{vu}, \ldots, x^k_{vu}$ where x^i_{uv} represents the value of the i-th flow through the edge uv in the direction from u to v. Now part (a) of (8.6.1) is equivalent to the linear program

$$\max \sum_{i=1}^{k} \sum_{r_i v \in E} (x^i_{r_i v} - x^i_{v r_i})$$

$$\begin{aligned} \textstyle\sum_{uv \in E} (x^i_{uv} - x^i_{vu}) &= 0 && \text{for all } u \in V \text{ and all } i = 1, \ldots, k, \\ \textstyle\sum_{i=1}^{k} (x^i_{uv} + x^i_{vu}) &\leq c_{uv} && \text{for all } uv \in E, \\ x^i_{vu}, x^i_{uv} &\geq 0 && \text{for all } uv \in E \text{ and all } i = 1, \ldots, k. \end{aligned}$$

Similarly, one can formulate part (b) of (8.6.1) as a nonemptiness problem. Hence, multicommodity flow problems can be solved in polynomial time. On the other hand, we may view version (a) of the multicommodity flow problem (8.6.1) as a (fractional) packing problem for the clutter consisting of all $[r_i, s_i]$-paths, $i = 1, \ldots, k$. We call the elements of this clutter **multicommodity paths** (with respect to $\{r_i, s_i\}$, $i = 1, \ldots, k$). The blocker of this clutter consists of all minimal sets E' of edges separating all pairs of $\{r_i, s_i\}$. We call these minimal sets **multicommodity cuts** (with respect to $\{r_i, s_i\}$, $i = 1, \ldots, k$). (Note that multicommodity cuts are not necessarily cuts in the usual sense.) This pair of clutters does not have the max-flow min-cut property in general. They yield three more combinatorial optimization problems.

(8.6.2) Shortest Multicommodity Path Problem. *Given a graph $G = (V, E)$, a collection of commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, and a length function $c : E \to \mathbb{Q}_+$, find a multicommodity path of minimum length.*

(8.6.3) Minimum Multicommodity Cut Problem. *Given a graph $G = (V, E)$, a collection of commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, and a capacity function $c : E \to \mathbb{Q}_+$, find a multicommodity cut of minimum capacity.*

(8.6.4) Multicommodity Cut Packing Problem. *Given a graph $G = (V, E)$, a collection of commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, and a length function $c : E \to \mathbb{Q}_+$, find a collection of multicommodity cuts $C_1, \ldots, C_t$ and nonnegative rationals $\lambda_1, \ldots, \lambda_t$ such that $\sum_{i, e \in C_i} \lambda_i \leq c(e)$ for all $e \in E$ and such that $\lambda_1 + \ldots + \lambda_t$ is as large as possible.*

Problem (8.6.2) is trivially solvable in polynomial time: we simply find, for each $i \in \{1, \ldots, k\}$, a shortest $[r_i, s_i]$-path; the shortest among these shortest paths will be a shortest multicommodity path. We do not know the complexity of problem (8.6.4), while JOHNSON, PAPADIMITRIOU, SEYMOUR and YANNAKAKIS (1984) showed that (8.6.3) is $\mathcal{NP}$-hard, even for the case of three commodities.

Similar techniques solve the directed versions of the multi-commodity flow problem. The **integer** and **half-integer multicommodity flow problems** (obtained from (8.6.1) by adding the condition that the multicommodity flow must be integer or half-integer), are $\mathcal{NP}$-hard. We close this chapter by discussing some

special cases, where the integral and half-integral multicommodity flow problems can be solved in polynomial time. They are restricted to the undirected case.

First note the following: if there exists a multicommodity flow subject to c for which the i-th value is at least ρ_i $(i = 1, \ldots, k)$, then:

(8.6.5) for each subset W of v, $\sum_{i \in \sigma(W)} \rho_i \leq c(\delta(W))$,

where $\sigma(W) := \{i = 1, \ldots, k \mid |\{r_i, s_i\} \cap W| = 1\}$.

(8.6.6) Theorem. *In the following cases,* (8.6.5) *is also* **sufficient** *for the solvability of* (8.6.1) (b)*:*

(a) $|\{r_1, s_1, \ldots, r_k, s_k\}| \leq 4$.

(b) *The graph* $(\{r_1, s_1, \ldots, r_k, s_k\}, \{r_1 s_1, \ldots, r_k s_k\})$ *is a circuit of length five or is the union of two stars.*

(c) *G is planar, and G has two faces F_1 and F_2 so that for each $i = 1, \ldots, k$, $r_i, s_i \in F_1$ or $r_i, s_i \in F_2$.*

(d) *The graph* $(V, E \cup \{r_1 s_1, \ldots, r_k s_k\})$ *is planar.* □

For proofs, we refer for (a) to HU (1963) and SEYMOUR (1980b); for (b) to LOMONOSOV (1979); for (c) to OKAMURA and SEYMOUR (1981) and OKAMURA (1983); for (d) to SEYMOUR (1981a).

It was shown moreover, that, in each of the cases (a), ..., (d) of (8.6.6), if all capacities and demands are integral, and (8.6.5) is satisfied, then there exists a half-integral multicommodity flow. More generally, if (8.6.5) is satisfied and if the capacities and demands are integral, so that

$$(8.6.7) \qquad \sum_{\substack{e \in E \\ e \ni v}} c(e) + \sum_{\substack{i=1 \\ v \in \{r_i, s_i\}}}^{k} \rho_i \quad \text{is even, for each } v \in V,$$

then there exists an integral multicommodity flow.

Polyhedrally, the above can be described as follows. For a given graph $G = (V, E)$ and commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, define the cone C in the space $\mathbb{R}^k \times \mathbb{R}^E$ by

$$(8.6.8)\ C := \{(\rho_1, \ldots, \rho_k, c^T)^T \mid \rho_1, \ldots, \rho_k \geq 0, c \in \mathbb{R}^E_+, \text{ and there exists a multicommodity flow subject to } c \text{ with } i\text{-th value } \rho_i\ (i = 1, \ldots, k)\}.$$

By definition, C is the convex cone generated by the following vectors:

$$(8.6.9) \qquad \text{(i)} \quad \begin{pmatrix} \varepsilon_i \\ \chi^P \end{pmatrix} \text{ for } i = 1, \ldots, k; \text{ and where } P \text{ is the edge-set of an } [r_i, s_i]\text{-path in } G;$$

$$\text{(ii)} \quad \begin{pmatrix} 0 \\ \varepsilon_e \end{pmatrix} \text{ for all } e \in E.$$

Here we use the following notation: ε_i is the i-th unit vector in $\mathbb{R}^k$; ε_e is the e-th unit vector in $\mathbb{R}^E$; χ^P is the incidence vector of P in $\mathbb{R}^E$.

Sufficiency of (8.6.5) (for each choice of capacities and demands) means that C is determined by the following linear inequalities:

$$\begin{aligned} &\rho_i \geq 0 \text{ for } i = 1, \ldots, k, \\ &c_e \geq 0 \text{ for all } e \in E, \\ &\sum_{i \in \sigma(W)} \rho_i \leq \sum_{e \in \delta(W)} c_e \text{ for all } W \subseteq V. \end{aligned} \tag{8.6.10}$$

So we know that in each of the cases (a), ..., (d) of (8.6.6), C is determined by (8.6.10). Moreover, if $(\rho^T, c^T)^T \in C$ is integral, it is a half-integral combination of the generators (8.6.9). If $(\rho^T, c^T)^T$ also satisfies (8.6.7), it is an integral combination of the generators (8.6.9).

Note that the optimization problem is polynomially solvable for the class of cones C defined in (8.6.8): we have to test whether the inner product of any given vector $(t^T, l^T)^T \in \mathbb{R}^k \times \mathbb{R}^E$ with each of the vectors (8.6.9) is nonnegative. To do this, we first test whether l is nonnegative; if not, $(t^T, l^T)^T$ has negative inner product with one of the vectors (8.6.9) (ii); if so, we next test, for each $i = 1, \ldots, k$, whether $-t_i$ is not larger than the length of the shortest $[r_i, s_i]$-path (length with respect to l).

Hence also the separation problem for the cones C is polynomially solvable. It implies that for each $(\rho^T, c^T)^T \in \mathbb{R}^k \times \mathbb{R}^E$ we can find linearly independent vectors $z_1, \ldots, z_t$ among the vectors defined in (8.6.9), together with scalars $\mu_1, \ldots, \mu_t$ so that

$$(\rho^T, c^T)^T = \mu_1 z_1 + \ldots + \mu_t z_t, \tag{8.6.11}$$

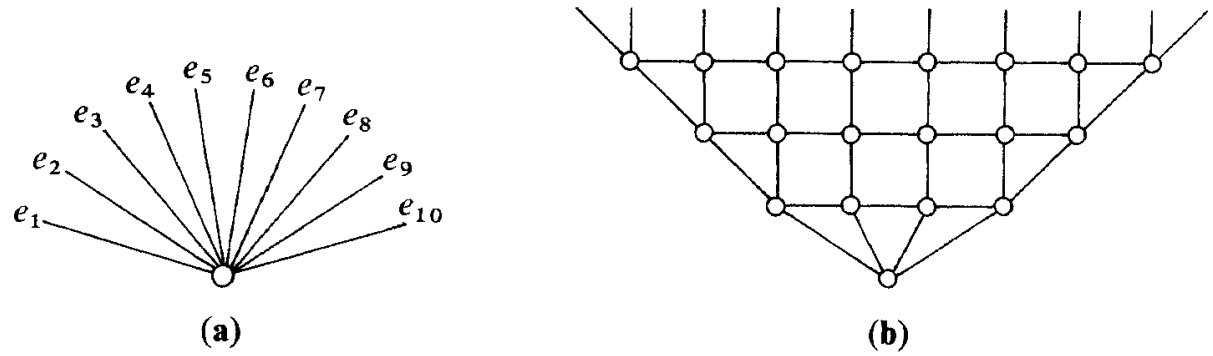
or decide that $(\rho^T, c^T)^T$ does not belong to C. Therefore, if we are in one of the cases (a), ..., (d) of (8.6.6), and if $(\rho^T, c^T)^T \notin C$, we can find in polynomial time a violated inequality among the inequalities (8.6.10). So if $(\rho^T, c^T)^T \geq 0$, then we find a subset F of E violating (8.6.5).

As mentioned above, if we are in one of the cases (a), ..., (d) of (8.6.6) and if $(\rho^T, c^T)^T$ is integral and satisfies (8.6.7), then we can decompose $(\rho^T, c^T)^T$ as (8.6.11) with integral μ_i. We describe a polynomial time algorithm to find such a decomposition.

First we decompose $(\rho^T, c^T)^T$ as in (8.6.11) with possibly fractional μ_i, where $z_1, \ldots, z_t$ are linearly independent vectors from (8.6.9) (this is possible in polynomial time). Without loss of generality, $z_1, \ldots, z_p$ are of type (8.6.9) (i), and $z_{p+1}, \ldots, z_t$ are of type (8.6.9) (ii). Next we replace $(\rho^T, c^T)^T$ by

$$(\tilde{\rho}^T, \tilde{c}^T)^T := (\rho^T, c^T)^T - \lfloor \mu_1 \rfloor z_1 - \ldots - \lfloor \mu_p \rfloor z_p - 2\lfloor \tfrac{1}{2}\mu_{p+1} \rfloor z_{p+1} - \ldots - 2\lfloor \tfrac{1}{2}\mu_t \rfloor z_t. \tag{8.6.12}$$

Then $(\tilde{\rho}^T, \tilde{c}^T)^T$ belongs to C, and it is integral and satisfies (8.6.7). (Observe that (8.6.12) involves rounding of rationals to integers. So the procedure we describe here is not strongly polynomial in the stricter sense.) So if we can decompose $(\tilde{\rho}^T, \tilde{c}^T)^T$ into an integral combination of vectors (8.6.9), we are finished.



(a) (b)

Figure 8.2

Note that

$$\|(\tilde{\rho}^T, \tilde{c}^T)^T\|_1 = (\mu_1 - \lfloor \mu_1 \rfloor)\|z_1\|_1 + \ldots + (\mu_p - \lfloor \mu_p \rfloor)\|z_p\|_1 + (\mu_{p+1} - 2\lfloor \tfrac{1}{2}\mu_{p+1} \rfloor)\|z_{p+1}\|_1 + \ldots + (\mu_t - 2\lfloor \tfrac{1}{2}\mu_t \rfloor)\|z_t\|_1 \leq 4(|E| + k)^2. \tag{8.6.13}$$

So in polynomial time we can replace each commodity $\{r_i, s_i\}$ by $\tilde{\rho}_i$ identical commodities $\{r_i, s_i\}$, and each edge of G by $\tilde{c}_e$ parallel edges. If we take, in the new structure, all capacities and demands equal to 1, we have a problem equivalent to the original problem. Note that if we add to this new graph an edge rs for each commodity $\{r, s\}$, then condition (8.6.7) implies that the graph obtained is Eulerian.

In this way our problem is reduced to the following:

(8.6.14) Problem. *Given an Eulerian graph $G = (V, E)$ (possibly with loops and multiple edges) and a subset D of E such that for each subset W of V we have*

$$|\delta_{E \setminus F}(W)| \geq |\delta_F(W)|, \tag{8.6.15}$$

partition E into circuits $C_1, \ldots, C_q$ so that $|C_i \cap D| \leq 1$ for all $i = 1, \ldots, q$.

Here $\delta_L(W) := \{e \in L \mid |e \cap W| = 1\}$ for $L \subseteq E$.

Suppose we are in one of the cases (a) and (b) of (8.6.6) (i. e., $|\cup D| \leq 4$, or D forms a circuit of length 5 (with parallel edges), or there are two nodes v, w so that each edge of D contains either v or w). If there exists a node v in which two edges of $E \setminus D$ meet, say uv and wv, we could "try" to replace the two edges by one edge uw: for each choice of such a pair of edges we test if, after the replacement, condition (8.6.5) is still satisfied (this can be done in polynomial time, as (8.6.10) can be tested in polynomial time). As soon as we find a pair uv, wv which we can replace by uw, we solve (8.6.14) for the smaller graph, which directly gives a solution for the original graph. If no such pair of edges exists, it follows that each of the circuits C_i must either be a loop, or consists of one edge in D and one edge in $E \setminus D$. In that case, solving (8.6.14) is trivial. It is not difficult to see that this yields a polynomial time procedure.

The cases (c) and (d) of (8.6.6) are slightly more difficult. We sketch a method. First we argue that we may assume that all nodes of G have degree at most 4. Indeed, we can replace any node of degree $2k$ ($k \geq 3$) by $k^2 - k - 1$ nodes of degree 4 as indicated for the case $k = 5$ in Figure 8.2.

One may check that if we choose any partition of the edges $e_1, \ldots, e_{2k}$ into k pairs of edges, there are k edge-disjoint paths in Figure 8.2 (b) so that each of these paths connects two edges of some pair of edges in the chosen partition. So (8.6.14) is solvable in the original graph, if and only if it is solvable in the new graph. Note moreover, that Figure 8.2 (b) is planar, so these replacements maintain cases (c) or (d). So we may assume that all nodes of G have degree at most 4.

Now consider any planar embedding of G (in case (c), the edges in D may cross in the faces F_1 or F_2). Let uv and vw be two "neighboring" edges in $E \setminus D$ (i. e., they occur consecutively, if we follow the edges incident with v in clockwise orientation). Again we try to replace uv and vw by one edge uw. If (8.6.15) still holds after the replacement, we recursively solve (8.6.14) for the smaller graph, which gives directly a solution for the original graph. Suppose next that no such pair of edges exists. By Theorem (8.6.6), we know that there exist circuits as described in (8.6.14). The edges in any node of degree 2 should go into the same circuit. Let v be a node of degree 4. If all edges incident with v belong to $E \setminus D$, then an edge and its "opposite" edge (in the planar embedding) should go into the same circuit. If all but one edge incident with v belong to $E \setminus D$, then again opposite edges should go into the same circuit. This implies that we can determine in polynomial time the sets $C_1 \setminus F, \ldots, C_q \setminus F$. Hence, we easily find also the sets $C_1, \ldots, C_q$.

Summing up, we get:

(8.6.16) Theorem. *For any given graph $G = (V, E)$ and given commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, satisfying one of the conditions* (a), ..., (d) *of* (8.6.6), *and any given integral demands $\rho_1, \ldots, \rho_k$ and integral capacities c_e ($e \in E$), satisfying* (8.6.5) *and* (8.6.7), *one can find in polynomial time an integral multicommodity flow subject to c satisfying the demands.* □

(8.6.17) Corollary. *For any given graph $G = (V, E)$ and given commodities $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$, satisfying one of the conditions* (a), ..., (d) *of* (8.6.6), *and any given integral demands $\rho_1, \ldots, \rho_k$ and integral capacities c_e ($e \in E$), satisfying* (8.6.5), *one can find in polynomial time a half-integral multicommodity flow subject to c satisfying the demands.*

Proof. Multiply all capacities and demands by 2. Then the premise of Theorem (8.6.16) is satisfied. Find a feasible integral multicommodity flow x. Then $\frac{1}{2}x$ is feasible for the original capacities and demands. □

*Chapter 9

Stable Sets in Graphs

In this chapter we survey the results of the polyhedral approach to a particular $\mathcal{NP}$-hard combinatorial optimization problem, the stable set problem in graphs. (Alternative names for this problem used in the literature are vertex packing, or coclique, or independent set problem.) Our basic technique will be to look for various classes of inequalities valid for the stable set polytope, and then develop polynomial time algorithms to check if a given vector satisfies all these constraints. Such an algorithm solves a relaxation of the stable set problem in polynomial time, i. e., provides an upper bound for the maximum weight of a stable set. If certain graphs have the property that every facet of the stable set polytope occurs in the given family of valid inequalities, then, for these graphs, the stable set problem can be solved in polynomial time. It turns out that there are very interesting classes of graphs which are in fact characterized by such a condition, most notably the class of perfect graphs. Using this approach, we shall develop a polynomial time algorithm for the stable set problem for perfect graphs. So far no purely combinatorial algorithm has been found to solve this problem in polynomial time.

Let us mention that all algorithms presented in this chapter can be made strongly polynomial using Theorem (6.6.5), with the natural exception of the algorithm designed to prove Theorem (9.3.30), which optimizes a linear objective function over a nonpolyhedral set.

*9.1 Odd Circuit Constraints and t-Perfect Graphs

Throughout this chapter, $G = (V, E)$ denotes a graph with node set $V = \{1, 2, \ldots, n\}$. Let $w : V \to \mathbb{Q}_+$ be any weighting of the nodes of G, and let $\alpha(G, w)$ denote the maximum weight of a stable set in G. It is well known that to determine $\alpha(G, w)$ is $\mathcal{NP}$-hard, even in the special case when $w = \mathbb{1}$ – see, for instance, GAREY and JOHNSON (1979).

Similarly as, say, in the case of matchings – see Sections 7.3 and 8.5 – we introduce the **stable set polytope**

$$\mathrm{STAB}(G) := \mathrm{conv}\{\chi^S \in \mathbb{R}^V \mid S \subseteq V \text{ stable set}\}$$

defined as the convex hull of the incidence vectors of all stable sets of nodes of G. Then $\alpha(G, w)$ is equal to the maximum value of the linear function $w^T x$ for $x \in \mathrm{STAB}(G)$. For this observation to be of any use, however, we need

information about inequalities defining STAB(G). So let us collect inequalities valid for STAB(G), and see if they are enough to describe this polytope.

The following sets of linear inequalities are obviously all valid for STAB(G) :

$$x_i \geq 0 \quad \text{for all } i \in V, \tag{9.1.1}$$

$$x_i + x_j \leq 1 \quad \text{for all } ij \in E. \tag{9.1.2}$$

It is also easy to see that the integral solutions of (9.1.1), (9.1.2) are exactly the incidence vectors of stable sets of nodes of G. Theorem (8.2.8) implies :

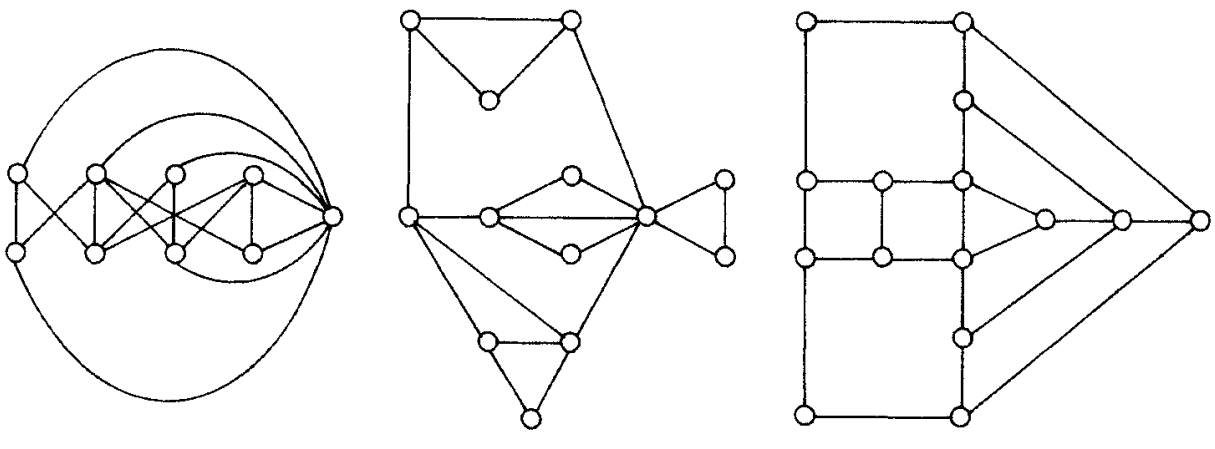
(9.1.3) Proposition. *The inequalities* (9.1.1), (9.1.2) *are sufficient to describe* STAB(G) *if and only if G is bipartite and has no isolated nodes.* □

We can take care of isolated nodes by adding, for each isolated node i, the inequality $x_i \leq 1$. So in particular, we see that the stable set problem for bipartite graphs can be solved using linear programming, since (9.1.1), (9.1.2) is an explicit system of linear inequalities, whose encoding length is polynomially bounded in the encoding length of G. (Combinatorial polynomial time methods for the stable set problem for bipartite graphs were mentioned in Section 8.2.)

The minimal graphs for which inequalities (9.1.1) and (9.1.2) are not sufficient to describe STAB(G) are the odd circuits. In fact, if $G = (V, E)$ is an odd circuit, the point $(\frac{1}{2}, \ldots, \frac{1}{2})^T \in \mathbb{R}^V$ satisfies all inequalities in (9.1.1) and (9.1.2) but is not in STAB(G). This suggests a new class of inequalities valid for STAB(G), the so-called **odd circuit constraints**:

$$\sum_{i \in V(C)} x_i \leq \frac{|V(C)| - 1}{2} \quad \text{for each odd circuit } C. \tag{9.1.4}$$

Let us call a graph **t-perfect** if (9.1.1), (9.1.2), and (9.1.4) are enough to describe STAB(G) (the "t" stands for "trou", the French word for hole). The study of these graphs was suggested by CHVÁTAL (1975). Although t-perfect graphs do not seem to occur in such an abundance as perfect graphs (to be described in the next section), there are some interesting classes of these graphs known.



(a) (b) (c)

Figure 9.1

(9.1.5) Examples.

I. Bipartite graphs. This follows trivially from Proposition (9.1.3), as odd circuit inequalities do not occur at all.

II. Almost bipartite graphs. A graph is **almost bipartite** if it has a node v such that all odd circuits go through v (equivalently, $G - v$ is bipartite) – see Figure 9.1 (a) for an example. The t-perfectness of this class was shown by FONLUPT and UHRY (1982). It is trivial to check if a graph is almost bipartite, and also the stable set problem for such graphs is easily reduced to the stable set problem for bipartite graphs.

III. Series-parallel graphs. A graph is **series-parallel** if it can be obtained from a forest by repeated application of the following operations: adding an edge parallel to an existing edge and replacing an edge by a path. DIRAC (1952) and DUFFIN (1965) characterized these graphs as those containing no subdivision of K_4. It is easy to check whether a graph is series-parallel. CHVÁTAL (1975) conjectured that series-parallel graphs are t-perfect. This was proved by BOULALA and UHRY (1979), who also designed a combinatorial polynomial time algorithm for the weighted stable set problem in series-parallel graphs. Series-parallel graphs include **cacti** which are graphs in which every block is either a bridge (i. e., an edge that is a cut) or a chordless circuit. A series-parallel graph is shown in Figure 9.1 (b).

IV. Nearly bipartite planar graphs. These are planar graphs in which at most two faces are bounded by an odd number of edges – see Figure 9.1 (c).Using a polynomial time planarity testing algorithm (HOPCROFT and TARJAN (1974)), these graphs can also be recognized in polynomial time. Their t-perfectness follows from the results of Gerards and Schrijver (see below).

V. Strongly t-perfect graphs. These are graphs that do not contain a subdivision of K_4 such that all four circuits corresponding to triangles in K_4 are odd. GERARDS and SCHRIJVER (1986) proved that these graphs are t-perfect. In fact, they proved that these graphs are characterized by the following property stronger than t-perfectness:

Let $a: V \to \mathbb{Z}$ and $b: V \to \mathbb{Z}$ be two integral weightings of the nodes, and let $c: E \to \mathbb{Z}$ and $d: E \to \mathbb{Z}$ be two integral weightings of the edges. Consider the inequalities

$$a_v \leq x_v \leq b_v \qquad \text{for each } v \in V, \tag{9.1.6}$$

$$c_{uv} \leq x_u + x_v \leq d_{uv} \qquad \text{for each } uv \in E, \tag{9.1.7}$$

and, for each circuit $C = (v_1, \ldots, v_k)$ and each choice of $\varepsilon_i = \pm 1$ $(i = 1, \ldots, k)$, the inequality

$$\sum_{i=1}^{k} \tfrac{\varepsilon_i + \varepsilon_{i-1}}{2} x_{v_i} \leq \left\lfloor \tfrac{1}{2} \left(\sum_{\varepsilon_i = 1} d_{v_i v_{i+1}} - \sum_{\varepsilon_i = -1} c_{v_i v_{i+1}} \right) \right\rfloor \tag{9.1.8}$$

taking indices mod k. *Then the solution set of* (9.1.6), (9.1.7), (9.1.8) *has integral vertices.*

So the inequalities (9.1.6), (9.1.7), (9.1.8) describe the convex hull of the integer solutions of (9.1.6), (9.1.7).

GERARDS, LOVÁSZ, SEYMOUR, SCHRIJVER and TRUEMPER (1987) proved that every strongly t-perfect graph can be "glued together" from almost bipartite graphs and nearly bipartite planar graphs. We do not go into the details of this decomposition, but remark that it yields combinatorial polynomial time procedures to recognize whether a graph is strongly t-perfect and to find a maximum weight stable set.

Not every t-perfect graph is strongly t-perfect, as is shown by the graph obtained from K_4 by subdividing the lines of a 4-circuit – see Figure 9.2. □

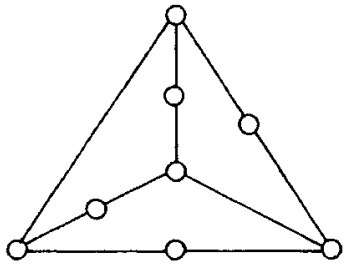


Figure 9.2

Let G be any graph. We set

$$\text{(9.1.9)} \qquad \text{CSTAB}(G) := \{ x \in \mathbb{R}^V \mid x \text{ satisfies (9.1.1), (9.1.2), (9.1.4)} \}$$

(CSTAB stands for, say, **circuit-constrained stable set polytope**). The main result in this section is the following.

(9.1.10) Theorem. *The strong optimization problem for* CSTAB(G) *can be solved in polynomial time for any graph G. Moreover, an optimum vertex solution can be found in polynomial time.*

By Theorem (6.4.9) and Lemma (6.5.1), it suffices to prove that the strong separation problem can be solved in polynomial time. Thus (9.1.10) is implied by the next lemma.

(9.1.11) Lemma. *There exists a polynomial time algorithm that, for any graph $G = (V, E)$ and for any vector $y \in \mathbb{Q}^V$, either*

(a) *asserts that $y \in$ CSTAB(G), or*

(b) *finds an inequality from* (9.1.1), (9.1.2), *or* (9.1.4) *violated by y.*

Proof. The inequalities in (9.1.1) and (9.1.2) are easily checked by substitution. So we may assume that $y \geq 0$ and that, for each edge $uv \in E$, $y_u + y_v \leq 1$.

Define, for each edge $e = uv \in E$, $z_e := 1 - y_u - y_v$. So $z_e \geq 0$. Then (9.1.4) is equivalent to the following set of inequalities :

$$\text{(9.1.12)} \qquad \sum_{e \in C} z_e \geq 1 \quad \text{for each odd circuit } C.$$

If we view z_e as the "length" of edge e, then (9.1.12) says that the length of a shortest odd circuit is at least 1. But a shortest odd circuit can be found in polynomial time (see (8.3.6) and the remarks thereafter). This proves the lemma. □

Since for t-perfect graphs CSTAB(G) = STAB(G) holds, we can conclude:

(9.1.13) Corollary. *A maximum weight stable set in a t-perfect graph can be found in polynomial time.* □

Theorem (9.1.10) implies that the property of t-perfectness is in co-$\mathcal{NP}$. In fact, to show that G is not t-perfect, it suffices to exhibit a vertex of CSTAB(G) which is nonintegral, and since we can optimize over CSTAB(G) in polynomial time, we can prove that the exhibited vector is indeed a vertex of CSTAB(G) in polynomial time by Corollary (6.5.10). We do not know whether the problem of checking t-perfectness is in $\mathcal{NP}$ or in $\mathcal{P}$.

*9.2 Clique Constraints and Perfect Graphs

Instead of the odd circuit constraints, it is also natural to consider the following system of so-called **clique constraints**:

$$x(Q) \leq 1 \quad \text{for all cliques } Q \subseteq V. \tag{9.2.1}$$

Note that (9.2.1) contains (9.1.2) as a special case, and also contains the triangle constraints from (9.1.4), but in general, (9.1.4) and (9.2.1) do not imply each other. The graphs G for which the constraints (9.1.1) and (9.2.1) suffice to describe STAB(G) are called **perfect**.

While polyhedrally this is a natural way to arrive at the notion of perfect graphs, there are several equivalent definitions, some of them in terms of more elementary graph theory. Perfect graphs were introduced by BERGE (1961) as common generalizations of several nice classes of graphs (see below for these special cases).

To motivate these elementary definitions of perfect graphs, observe that each graph G satisfies the following inequalities for the clique number $\omega(G)$, the coloring number $\chi(G)$, the stability number $\alpha(G)$ and the clique covering number $\bar{\chi}(G)$:

$$\begin{aligned} \omega(G) &\leq \chi(G), \\ \alpha(G) &\leq \bar{\chi}(G). \end{aligned} \tag{9.2.2}$$

The pentagon C_5 shows that strict inequality can occur in both inequalities above. Now BERGE (1961,1962) called a graph G **perfect** if

$$\omega(G') = \chi(G') \tag{9.2.3}$$

holds for each induced subgraph G' of G.

We will give a list of known classes of perfect graphs later, but observe that, e. g., not only bipartite graphs (trivially), line graphs of bipartite graphs (by König's edge-coloring theorem (7.4.3)) and comparability graphs (trivially) are perfect, but so are their complements (by König's edge covering theorem (8.2.4), König's matching theorem (8.2.3) and Dilworth's theorem (8.3.19), respectively). These results made BERGE (1961,1962) conjecture that the complement of a perfect graph is perfect again. This was proved by LOVÁSZ (1972). Note that this result generalizes the theorems of König and Dilworth mentioned above.

To illustrate the ideas relating combinatorial and polyhedral properties of perfect graphs, we give the proof of the following theorem, which is a combination of results of FULKERSON (1970) and LOVÁSZ (1972).

Let $w: V \to \mathbb{Z}_+$ be any weighting of the nodes of the graph G, and let $\chi(G,w)$ denote the weighted chromatic number of G, i. e., the minimum number k of stable sets $S_1, S_2, \ldots, S_k$ such that each $i \in V$ is contained in w_i of these sets. Note that for $w = \mathbb{1}$ this number is just the chromatic number of G, for arbitrary w, it could be defined as the chromatic number of the graph obtained from G by replacing each $i \in V$ by a complete subgraph of w_i nodes.

(9.2.4) Theorem. *For any graph $G = (V,E)$, the following are equivalent.*

(i) $\chi(G') = \omega(G')$ *for each induced subgraph G' of G.*
(ii) $\chi(G,w) = \omega(G,w)$ *for each weighting* $w: V \to \mathbb{Z}_+$.
(iii) STAB(G) *is determined by the constraints* (9.1.1) *and* (9.2.1).
(iv) *The complement $\overline{G}$ of G satisfies* (i).
(v) *The complement $\overline{G}$ of G satisfies* (ii).
(vi) *The complement $\overline{G}$ of G satisfies* (iii) .

Proof. (i) $\Rightarrow$ (ii). We use induction on $w(V) = \sum_{v \in V} w(v)$. If $w \leq \mathbb{1}$ then (ii) specializes to just (i), so we may assume that there is a node $i_0 \in V$ such that $w(i_0) > 1$. Consider the weights

$$w'(i) := \begin{cases} w(i_0) - 1 & \text{if } i = i_0, \\ w(i) & \text{if } i \neq i_0. \end{cases}$$

Then by the induction hypothesis, $\chi(G,w') = \omega(G,w')$, that is, there exists a family $S_1, S_2, \ldots, S_N$ of stable sets such that each node i is contained in $w'(i)$ of them and $N = \omega(G,w')$. Since $w'(i_0) = w(i_0) - 1 \geq 1$, there is a stable set S_j containing the node i_0, say $i_0 \in S_1$.

Now consider the weighting

$$w'' := w - \chi^{S_1}.$$

Let Q be a clique so that $w''(Q) = \omega(G,w'')$. If $S_1 \cap Q = \emptyset$ then

$$\omega(G,w'') = w''(Q) = \sum_{i \in Q} w''(i) \leq \sum_{i \in Q} w'(i) = \sum_{j=1}^{N} |S_j \cap Q| \leq \sum_{j=2}^{N} 1 = N - 1$$
$$= \omega(G,w') - 1 \leq \omega(G,w) - 1 .$$

If $S_1 \cap Q \neq \emptyset$ then

$$\omega(G, w'') = w''(Q) \leq w(Q) - 1 \leq \omega(G, w) - 1.$$

So it follows that

$$\omega(G, w'') \leq \omega(G, w) - 1.$$

By the induction hypothesis, there is a family $S_1', S_2', \ldots, S_{\omega(G,w'')}$ of stable sets in G such that each node i is contained in $w''(i)$ of them. Adding S_1 to this family, we see that $\chi(G, w) \leq \omega(G, w)$. Since the reverse inequality holds trivially, (ii) is proved.

(ii) $\Rightarrow$ (iii). Let $x \in \mathbb{Q}^V$ be any vector satisfying (9.1.1) and (9.2.1). We show that $x \in \mathrm{STAB}(G)$. Let q be the least common denominator of the entries in x. Then $qx \in \mathbb{Z}_+^V$ and (9.2.1) says that $\omega(G, qx) \leq q$. Hence by (ii), $\chi(G, qx) \leq q$, i. e., there exists a family $S_1, S_2, \ldots, S_q$ of stable sets such that each $i \in V$ is contained in exactly qx_i of them. In other words

$$qx = \sum_{j=1}^{q} \chi^{S_j}, \quad \text{i. e.,} \quad x = \tfrac{1}{q} \sum_{j=1}^{q} \chi^{S_j},$$

which shows that $x \in \mathrm{STAB}(G)$.

(iii) $\Rightarrow$ (iv). If $\mathrm{STAB}(G)$ is determined by (9.1.1) and (9.2.1) then the same is true for every induced subgraph G' of G. So we only have to show that G can be partitioned into $\alpha(G)$ complete subgraphs. We use induction on $|V|$. Let F be the face of $\mathrm{STAB}(G)$ spanned by all stable sets of size $\alpha(G)$; obviously, there is a facet of the form $x(Q) \leq 1$ containing F, where Q is a clique. But this means that for each maximum stable set S, $|S \cap Q| = \chi^S(Q) = 1$. Hence $\alpha(G - Q) < \alpha(G)$. By the induction hypothesis, $G - Q$ can be partitioned into $\alpha(G - Q)$ complete subgraphs. Adding Q to this system, we obtain a partition of G into $\alpha(G)$ complete subgraphs.

The implications (iv) $\Rightarrow$ (v) $\Rightarrow$ (vi) $\Rightarrow$ (i) follow by interchanging the roles of G and $\overline{G}$. □

It follows from these characterizations that, for perfect graphs, the clique, stable set, coloring, and clique covering problems, as well as their weighted versions, belong to the class $\mathcal{NP} \cap \text{co-}\mathcal{NP}$.

In fact, as we shall see later, in the perfect graph case these problems belong to the class $\mathcal{P}$ of polynomially solvable problems. So this implies the polynomial time solvability of these four problems for several classes of graphs which were shown to be perfect.

First we shall list a number of classes of perfect graphs discovered until now. Clearly, with each graph automatically the class of complementary graphs is associated, which are perfect again. At each of the classes we shall give references to polynomial algorithms developed for those graphs for the four combinatorial optimization problems mentioned above (in a combinatorial, non-ellipsoidal way).

One should keep in mind that for perfect graphs determining $\omega(G)$ is the same as determining $\chi(G)$. However, finding an explicit coloring of size $\chi(G)$ and finding an explicit maximum clique may be more complex (similarly, for $\alpha(G)$ and $\bar{\chi}(G)$). Moreover, the clique problem (coloring problem) for a class of graphs is equivalent to the stable set problem (clique covering problem) for the class of complementary graphs.

Beside the four combinatorial optimization problems mentioned, there is the **recognition problem** for a class $\mathscr{C}$ of graphs: given a graph G, does G belong to $\mathscr{C}$? Clearly, one can speak of this problem being in $\mathscr{NP}$, in co-$\mathscr{NP}$, in $\mathscr{P}$, or $\mathscr{NP}$-complete. With each of the classes below we shall also discuss what is known on the complexity of the corresponding recognition problem. Obviously, the recognition problem for a class of graphs is equivalent to the recognition problem for the class of complements.

The status of the recognition problem of the class of all perfect graphs is unknown. It is well-known that this problem belongs to co-$\mathscr{NP}$. That is, one can prove in polynomial time that a given graph is not perfect. This would also follow directly from the strong perfect graph conjecture, posed by BERGE (1962), which is still unsolved.

(9.2.5) Strong Perfect Graph Conjecture. *A graph is perfect if and only if it does not contain an odd circuit of length at least five, or its complement, as an induced subgraph.* □

The content of this conjecture is that the minimal (under taking induced subgraphs) imperfect graphs (these graphs are also called critically imperfect graphs) are the odd circuits of length at least five and their complements.

We now give a list of classes of perfect graphs. Note that each class is closed under taking induced subgraphs. In these notes, by perfectness we mean that (9.2.3) is satisfied. Since a number of perfectness results appeared before Theorem (9.2.4) was established, we mention the results for classes of graphs and their complements separately.

(9.2.6) Classes of perfect graphs.

I. Bipartite graphs, and their complements. Bipartite graphs are trivially perfect. The perfectness of their complements follows from König's edge covering theorem (8.2.4). For bipartite graphs, the weighted clique and coloring problems are easily polynomially solvable, while the weighted stable set and clique covering problems were shown to be polynomially solvable by KUHN (1955) and FORD and FULKERSON (1956) – cf. Section 8.2. The recognition problem for bipartite graphs is easily solvable in polynomial time.

II. Line graphs of bipartite graphs, and their complements. The perfectness of line graphs of bipartite graphs follows from König's edge-coloring theorem (7.4.3), and that of their complements from König's Matching Theorem (8.2.3). For line graphs of bipartite graphs, the weighted clique problem is trivial. A polynomial algorithm for the unweighted coloring problem follows from König's proof in KÖNIG (1916). The weighted case follows from the proof of Satz 15 in Kapitel

XI given in KÖNIG (1936). Polynomial algorithms for the weighted stable set and clique covering problems were given by KUHN (1955) and FORD and FULKERSON (1956) – cf. Section 8.2. VAN ROOIJ and WILF (1965) showed that line graphs can be recognized and that their "ancestors" can be reconstructed in polynomial time. Hence, the recognition problem for line graphs of bipartite graphs is polynomially solvable.

III. Interval graphs, and their complements. A graph is an **interval graph** if its nodes can be represented by intervals on the real line such that two nodes are adjacent if and only if the corresponding intervals have a nonempty intersection. The perfectness of interval graphs follows from Dilworth's theorem (8.3.19). The perfectness of their complements was observed by Gallai. The polynomial solvability of the weighted clique, stable set, coloring, and clique covering problems is not difficult. The recognition problem can be solved in polynomial time (FULKERSON and GROSS (1965)), even in linear time – see BOOTH and LUEKER (1976) (see LEKKERKERKER and BOLAND (1962) and GILMORE and HOFFMAN (1964) for good characterizations). A graph is an interval graph if and only if it is triangulated and its complement is a comparability graph – see IV and V below.

IV. Comparability graphs, and their complements. A graph is a **comparability graph** if its edges can be oriented to obtain a transitive acyclic digraph $D = (V, A)$, i. e., a digraph satisfying: *if* $(u, v) \in A$ *and* $(v, w) \in A$ *then* $(u, w) \in A$. The perfectness of comparability graphs is easy, while that of their complements is equivalent to Dilworth's theorem (8.3.19). For comparability graphs polynomial algorithms for the weighted clique and coloring problems are trivial. Polynomial algorithms for the weighted stable set and clique covering problems can be easily derived from min-cost flow algorithms. Such an algorithm can also be derived from any maximum flow algorithm by the construction of FORD and FULKERSON (1962). The recognition problem for comparability graphs is polynomially solvable by the method of GALLAI (1967) (membership in $\mathcal{NP}$ is trivial, while membership in co-$\mathcal{NP}$ was shown by GHOUILA-HOURI (1962) and GILMORE and HOFFMAN (1964)).

Note that comparability graphs include bipartite graphs and complements of interval graphs. Another subclass is formed by the **permutation graphs**, which can be defined as those comparability graphs whose complement is again a comparability graph – see EVEN, PNUELI and LEMPEL (1972).

V. Triangulated graphs, and their complements. A graph is a **triangulated** (or chordal) **graph** if it does not contain a circuit of length at least four as induced subgraph. BERGE (1960) proved perfectness of triangulated graphs, and HAJNAL and SURÁNYI (1958) proved perfectness of their complements. DIRAC (1961) showed that a triangulated graph always contains a node all of whose neighbors form a clique. This yields that an undirected graph is triangulated if and only if its edges can be oriented to obtain an acyclic directed graph $D = (V, A)$ satisfying: *if* $(u, v) \in A$ *and* $(u, w) \in A$ *then* $(v, w) \in A$ *or* $(w, v) \in A$. Dirac's theorem also gives that a graph is triangulated if and only if it is the intersection graph of a collection of subtrees of a tree. The polynomial solvability of the weighted clique, stable set, coloring, and clique covering problems was shown by GAVRIL (1972) and FRANK (1976) (here again Dirac's result is used). Also the recognition

problem can be shown to be polynomially solvable with Dirac's result. For linear time algorithms – see LUEKER (1974), ROSE and TARJAN (1975), and ROSE, TARJAN and LUEKER (1976). Triangulated graphs include interval graphs. They also include **split graphs**, which are graphs whose node set is the union of a clique and a stable set. These graphs can be characterized by the property that they are triangulated and their complements are triangulated.

VI. Parity graphs, and their complements. A graph is a **parity graph** if each odd circuit of length at least five has two crossing chords. The perfectness of this class of graphs was shown by E. Olaru – see SACHS (1970). The polynomial solvability of the weighted clique, stable set, coloring, and clique covering problems was shown by BURLET and UHRY (1982), who also proved the polynomial time solvability of the recognition problem for parity graphs. Clearly, parity graphs include bipartite graphs. They also include **line perfect** graphs (which are graphs G for which the line graph $L(G)$ is perfect), since TROTTER (1977) showed that a graph is line perfect if and only if it does not contain an odd circuit of length larger than three.

VII. Gallai graphs, and their complements. A graph is a **Gallai graph** (or *i*-triangulated graph) if each odd circuit of length at least five has two noncrossing chords. The perfectness of this class of graphs was shown by GALLAI (1962). BURLET and FONLUPT (1984) gave combinatorial algorithms that solve the unweighted versions of the four basic problems in polynomial time, and they also showed that Gallai graphs can be recognized in polynomial time. The recognition problem was also solved by WHITESIDES (1984). Gallai graphs include bipartite graphs, interval graphs, line perfect graphs, and triangulated graphs.

VIII. Meyniel graphs, and their complements. A graph is a **Meyniel graph** if each odd circuit of length at least five has two chords. The perfectness of this class of graphs was shown by MEYNIEL (1976). Again BURLET and FONLUPT (1984) found polynomial time combinatorial algorithms for the unweighted versions of all problems in question. Meyniel graphs include bipartite graphs, interval graphs, triangulated graphs, parity graphs, and Gallai graphs.

IX. Perfectly orderable graphs, and their complements. A graph is a **perfectly orderable** graph if its edges can be oriented to obtain an acyclic directed graph (V, A) with no induced subgraph isomorphic to $(\{t, u, v, w\}, \{(t, u), (u, v), (w, v)\})$. The perfectness of perfectly orderable graphs was shown by CHVÁTAL (1984). The recognition problem for perfectly orderable graphs trivially is in $\mathcal{NP}$, but no polynomial time algorithm for recognizing these graphs is known. Once an orientation with the above property is found, a maximum clique and a minimum coloring can be obtained by a greedy algorithm – see CHVÁTAL (1984). Combinatorial polynomial time algorithms for the problems in question are not known. Perfectly orderable graphs include bipartite graphs, interval graphs, complements of interval graphs, comparability graphs, triangulated graphs, and complements of triangulated graphs. CHVÁTAL, HOANG, MAHADEV and DE WERRA (1985) showed that a number of further classes of graphs are perfectly orderable.

X. Unimodular graphs, and their complements. An undirected graph is **unimodular** if the matrix whose rows are the incidence vectors of its maximal cliques is

totally unimodular. The perfectness of unimodular graphs follows from results of BERGE and LAS VERGNAS (1970). The perfectness of complements of unimodular graphs follows from the results of HOFFMAN and KRUSKAL (1956) on totally unimodular matrices. W. H. Cunningham (private communication) observed that the recognition problem for unimodular graphs can be solved in polynomial time using the algorithm of SEYMOUR (1980a) to recognize totally unimodular matrices. The stable set and clique cover problems for unimodular graphs can be written as explicit linear programs with totally unimodular matrix and therefore they are solvable by any linear programming algorithm. MAURRAS, TRUEMPER and AKGÜL (1981) designed a special algorithm for such linear programs while Bland and Edmonds (unpublished) remarked that Seymour's decomposition of totally unimodular matrices yields a combinatorial algorithm for such LP's. The clique problem is trivial, since, by a result of HELLER (1957), there are at most $\binom{n}{2}$ maximal cliques. Also the coloring problem can be reduced to linear programs over totally unimodular matrices. Unimodular graphs include bipartite graphs and their line graphs, interval graphs, and the class of graphs which do not contain an odd circuit of length at least five, the complement of such a circuit, or a $K_4 - e$ as an induced subgraph.

XI. Parthasarathy-Ravindra graphs, and their complements. A graph is a **Parthasarathy-Ravindra graph** if it does not contain a claw, an odd circuit of length at least five, or the complement of such an odd circuit as an induced subgraph. Perfectness of these graphs was proved by PARTHASARATHY and RAVINDRA (1976). SBIHI (1978, 1980) and MINTY (1980) showed that the stable set problem can be solved in polynomial time for claw-free (not necessarily perfect) graphs (this class of graphs contains the line graphs, and thus the matching problem is included). Minty's algorithm extends to the weighted case. For Parthasarathy-Ravindra graphs polynomial time algorithms for the cardinality coloring problem resp. the weighted clique and weighted clique covering problems were given by HSU (1981) resp. HSU and NEMHAUSER (1981, 1982). Parthasarathy-Ravindra graphs can be recognized in polynomial time by a method of V. Chvátal and N. Sbihi (personal communication). This class of graphs includes line graphs of bipartite graphs and complements of bipartite graphs.

XII. Strongly perfect graphs, and their complements. A graph is **strongly perfect** if every induced subgraph contains a stable set of nodes that meets all its (inclusionwise) maximal cliques. Perfectness of these graphs was observed by BERGE and DUCHET (1984) who also proved that the recognition problem for strongly perfect graphs is in co-$\mathcal{NP}$. A polynomial time algorithm for recognizing these graphs is not known. Moreover, combinatorial polynomial time algorithms for the four optimization problems in question have not been found yet. The class of strongly perfect graphs contains Meyniel graphs, comparability graphs, and perfectly orderable graphs.

XIII. Weakly triangulated graphs. A graph is **weakly triangulated** if it does neither contain a circuit of length at least five nor the complement of a circuit of length at least five as an induced subgraph. HAYWARD (1985) proved that weakly triangulated graphs are perfect. The recognition problem for weakly triangulated graphs is trivially in co-$\mathcal{NP}$. No polynomial time algorithm for the recognition

problem, and no combinatorial polynomial time algorithm for any of the four optimization problems for weakly triangulated graphs is known. Triangulated graphs and their complements are weakly triangulated.

XIV. Quasi parity graphs, and their complements. A graph is a **quasi parity** graph if every induced subgraph G' that is not a clique has two nodes that are not connected by a chordless path of odd length in G'. Perfectness of quasi parity graphs was shown by MEYNIEL (1985). Berge (personal communication) observed that this class of graphs can be enlarged by the graphs for which every induced subgraph G' with at least two nodes has the property that either G' or the complement of G' has two nodes that are not connected by a chordless path of odd length. The recognition problem for this class (resp. these two classes) of graphs is in co-$\mathcal{NP}$ but is not known to be in $\mathcal{P}$. No combinatorial polynomial time algorithm for the four optimization problems is known. Quasi parity graphs include Meyniel graphs, and hence parity graphs and Gallai graphs. □

More information about perfect graphs and their properties can be found in GOLUMBIC (1980), LOVÁSZ (1983a), and the collection of papers BERGE and CHVÁTAL (1984).

Now we continue our study of the inequality systems (9.1.1) and (9.2.1). Let $G = (V, E)$ be any graph. We set

$$\text{(9.2.7)} \qquad \text{QSTAB}(G) := \{ x \in \mathbb{R}^V \mid x \text{ satisfies (9.1.1) and (9.2.1)} \},$$

and call it the **clique-constrained stable set polytope**. (In the literature QSTAB (G) is often called the "fractional stable set polytope".) The following immediate observation relates stable set properties of a graph G and its complement $\overline{G}$:

(9.2.8) Proposition. *The antiblocker of* STAB(G) *is* QSTAB$(\overline{G})$, *and the antiblocker of* QSTAB(G) *is* STAB$(\overline{G})$. □

By analogy with the preceding section, one may expect that the optimization problem over QSTAB(G) is polynomially solvable. This expectation is even more justified since QSTAB(G) has nice and "easily recognizable" facets. However GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981) proved the following :

(9.2.9) Theorem. *The optimization problem for* QSTAB(G) *is* $\mathcal{NP}$*-hard.*

Although we usually do not prove $\mathcal{NP}$-hardness results in this book, we sketch the proof of Theorem (9.2.9) because it makes use of the ellipsoid method.

Proof. (Sketch). The optimization problem for QSTAB(G) is polynomially equivalent to the optimization problem for its antiblocker (see Exercise (6.5.18)), which is just the polytope STAB$(\overline{G})$. So this problem is equivalent to the stable set problem for $\overline{G}$, which is $\mathcal{NP}$-hard for general graphs. □

In the next section, however, we introduce an infinite class of valid inequalities for STAB(G) which includes the clique constraints (9.2.1) and for which the separation problem can be solved in polynomial time.

Antiblockers of Hypergraphs

Before doing this let us come back – just for a side remark – to an issue raised in Section 8.1. We have seen there how the notion of blocking hypergraphs and its relation to blocking polyhedra provides a common framework for quite a number of combinatorial optimization problems. We now introduce the analogous notion of antiblocking hypergraphs and relate it to antiblocking polyhedra. It will turn out that this is equivalent to the study of perfect graphs. These results are due to FULKERSON (1971, 1972).

Given a hypergraph H, its **antiblocker** $\mathrm{ABL}(H)$ is the set of all inclusionwise maximal subsets of $\cup H$ that intersect each edge of H in at most one element. So $\mathrm{ABL}(H)$ is a clutter. Define a graph $G(H)$ on the node set $\cup H$ by connecting two nodes if and only if they are contained in an edge of H. Then $\mathrm{ABL}(H)$ is the collection of maximal stable sets of $G(H)$. In contrast to (8.1.1), which states that $\mathrm{BL}(\mathrm{BL}(H)) = H$ holds for each clutter H, there are clutters H for which $\mathrm{ABL}(\mathrm{ABL}(H)) \neq H$; e. g., take $H = \{\{1,2\},\{2,3\},\{1,3\}\}$. In fact, $\mathrm{ABL}(\mathrm{ABL}(H)) = H$ if and only if H is the clutter of all maximal stable sets of a graph. Such an H is called a **conformal clutter**. In this case, $\mathrm{ABL}(H)$ is just the clutter of maximal cliques of the graph, i. e., the clutter of maximal stable sets of the complementary graph. Note that this complementary graph is just $G(H)$.

In analogy to blocking theory we consider the antidominant in $\mathbb{R}_+^{\cup H}$ – see Section 0.1 – of incidence vectors of a clutter H and denote it by $\mathrm{admt}(H)$, i. e.,

$$\mathrm{admt}(H) := \left(\mathrm{conv}\{\chi^E \in \mathbb{R}^{\cup H} \mid E \in H\} - \mathbb{R}_+^{\cup H}\right) \cap \mathbb{R}_+^{\cup H} .$$

If H is conformal then this is just the stable set polytope of the complement $\overline{G(H)}$ of $G(H)$. Again valid inequalities for $\mathrm{admt}(H)$ are

$$\text{(9.2.10)} \qquad \begin{aligned} &\text{(a)} \quad x_v \geq 0 \quad \text{for all } v \in \cup H \ , \\ &\text{(b)} \quad x(F) \leq 1 \quad \text{for all } F \in \mathrm{ABL}(H) \ . \end{aligned}$$

The question when these inequalities suffice to describe $\mathrm{admt}(H)$ is completely answered by the following theorem which can easily be derived from the previous results on perfect graphs.

(9.2.11) Theorem. *For a clutter H the following are equivalent:*
(i) (9.2.10) *suffices to describe* $\mathrm{admt}(H)$*, i. e.,* (9.2.10) *is TPI.*
(ii) (9.2.10) *is TDI.*
(iii) *H is conformal and $G(H)$ is perfect.*
(iv) *The following system is TPI: $x_v \geq 0$ for all $v \in \cup H$, $x(F) \leq 1$ for all $F \in H$.*
(v) *The system in* (iv) *is TDI.*

□

So the analogue of Lehman's theorem (8.1.5) holds even with dual integrality. The above shows that instead of antiblocking hypergraphs it suffices to study stable sets in graphs.

*9.3 Orthonormal Representations

The approach described in this section is based on LOVÁSZ (1979). The extension to the weighted case and a discussion of its algorithmic implications can be found in GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1981, 1984b, 1986).

Let $G = (V, E)$ be a graph. An **orthonormal representation of** G is a sequence $(u_i \mid i \in V)$ of $|V|$ vectors $u_i \in \mathbb{R}^N$, where N is some positive integer, such that $\|u_i\| = 1$ for all $i \in V$ and $u_i^T u_j = 0$ for all pairs i,j of nonadjacent vertices. Trivially, every graph has an orthonormal representation (just take all the vectors u_i mutually orthogonal in $\mathbb{R}^V$). Figure 9.3 shows a less trivial orthonormal representation of the pentagon C_5 in $\mathbb{R}^3$. It is constructed as follows. Consider an umbrella with five ribs of unit length (representing the nodes of C_5) and open it in such a way that nonadjacent ribs are orthogonal. Clearly, this can be achieved in $\mathbb{R}^3$ and gives an orthonormal representation of the pentagon. The central handle (of unit length) is also shown.

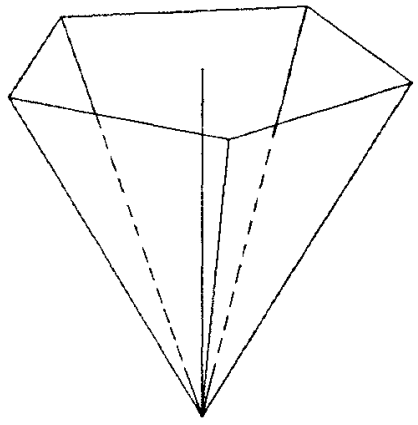


Figure 9.3

Let $(u_i \mid i \in V)$, $u_i \in \mathbb{R}^N$, be any orthonormal representation of G and let $c \in \mathbb{R}^N$ with $\|c\| = 1$. Then for any stable set $S \subseteq V$, the vectors u_i, $i \in S$, are mutually orthogonal and hence,

$$\sum_{i \in S} (c^T u_i)^2 \leq 1 \,. \tag{9.3.1}$$

Since $\sum_{i \in V} (c^T u_i)^2 \chi_i^S = \sum_{i \in S} (c^T u_i)^2$, we see that the inequality

$$\sum_{i \in V} (c^T u_i)^2 x_i \leq 1 \tag{9.3.2}$$

holds for the incidence vector $\chi^S \in \mathbb{R}^V$ of any stable S set of nodes of G. Thus, (9.3.2) is a valid inequality for STAB(G) for any orthonormal representation $(u_i \mid i \in V)$ of G, where $u_i \in \mathbb{R}^N$, and any unit vector $c \in \mathbb{R}^N$. We shall call (9.3.2) the **orthonormal representation constraints** for STAB(G).

If Q is any clique of G, we can define an orthonormal representation $(u_i \mid i \in V)$ as follows. Let $\{u_i \mid i \in V \setminus Q\} \cup \{c\}$ be mutually orthogonal unit vectors and set $u_j = c$ for $j \in Q$. Then the constraint (9.3.2) is just the clique constraint (9.2.1)

determined by Q. The orthonormal representation of C_5 depicted in Figure 9.3, with c pointing along its axis of symmetry, yields the inequality $\sum_{i\in C_5} \frac{1}{\sqrt{5}} x_i \leq 1$ or $\sum_{i\in C_5} x_i \leq \sqrt{5}$, which is not as strong as the odd circuit constraint but is not implied by the clique constraints. For any graph $G = (V, E)$, set

$$\text{(9.3.3)} \qquad \mathrm{TH}(G) := \{\, x \in \mathbb{R}^V \mid x \geq 0 \text{ and } x \text{ satisfies all orthonormal representation constraints (9.3.2)} \,\}.$$

(TH comes from the function "ϑ" to be studied below.) $\mathrm{TH}(G)$ is the intersection of infinitely many halfspaces, so $\mathrm{TH}(G)$ is a convex set. From the remarks above, it follows that

$$\text{(9.3.4)} \qquad \mathrm{STAB}(G) \subseteq \mathrm{TH}(G) \subseteq \mathrm{QSTAB}(G).$$

In general, $\mathrm{TH}(G)$ is not a polytope. In fact, we will prove later that $\mathrm{TH}(G)$ is a polytope if and only if G is perfect. The most important property of $\mathrm{TH}(G)$ – for our purposes – is that one can optimize any linear objective function over $\mathrm{TH}(G)$ in polynomial time. The algorithm to achieve this depends on a number of somewhat involved formulas for the maximum value of such an objective function.

Given a graph $G = (V, E)$ and weights $w \in \mathbb{R}_+^V$, we are going to study the value

$$\text{(9.3.5)} \qquad \vartheta(G, w) := \max\{\, w^T x \mid x \in \mathrm{TH}(G) \,\}.$$

Let us (temporarily) introduce a number of further values (which eventually will all turn out to be equal to $\vartheta(G, w)$).

Where $(u_i \mid i \in V)$, $u_i \in \mathbb{R}^N$, ranges over all orthonormal representations of G and $c \in \mathbb{R}^N$ over all vectors of unit length, let

$$\text{(9.3.6)} \qquad \vartheta_1(G, w) := \min_{\{c,(u_i)\}} \max_{i\in V} \frac{w_i}{(c^T u_i)^2}.$$

The quotient in (9.3.6) has to be interpreted as follows. If $w_i = 0$ then we take $w_i/(c^T u_i)^2 = 0$, even if $c^T u_i = 0$. If $w_i > 0$ but $c^T u_i = 0$ then we take $w_i/(c^T u_i)^2 = +\infty$. For notational convenience we introduce a number of further sets associated with the graph $G = (V, E)$:

$$\text{(9.3.7)} \qquad \begin{aligned} \mathcal{S} &:= \{\, A \in \mathbb{R}^{V\times V} \mid A \text{ symmetric} \}, \\ \mathcal{M} &:= \{\, B = (b_{ij}) \in \mathcal{S} \mid b_{ij} = 0 \text{ for all } i,j \text{ adjacent in } G \}, \\ \mathcal{M}^\perp &:= \{\, A = (a_{ij}) \in \mathcal{S} \mid a_{ii} = 0 \text{ for all } i \in V \text{ and} \\ &\qquad\qquad a_{ij} = 0 \text{ for all } i,j \text{ nonadjacent in } G \}, \\ \mathcal{D} &:= \{\, A \in \mathcal{S} \mid A \text{ positive semidefinite} \,\}. \end{aligned}$$

Clearly, $\mathcal{M}$ is a linear subspace of the space $\mathcal{S}$ of symmetric $|V|\times|V|$-matrices, and $\mathcal{M}^\perp$ is the orthogonal complement of $\mathcal{M}$ in $\mathcal{S}$. For $w \in \mathbb{R}_+^V$, let moreover

$$\text{(9.3.8)} \qquad \begin{aligned} \overline{w} &:= (\sqrt{w_i} \mid i \in V) \in \mathbb{R}^V, \\ W &:= \overline{w}\,\overline{w}^T = (\sqrt{w_i w_j} \mid i,j \in V) \in \mathcal{S}. \end{aligned}$$

Define

$$\vartheta_2(G,w) := \min\{\Lambda(A+W) \mid A \in \mathscr{M}^\perp\}, \tag{9.3.9}$$

where $\Lambda(D)$ denotes the largest eigenvalue of D, and

$$\vartheta_3(G,w) := \max\{\overline{w}^T B\overline{w} \mid B \in \mathscr{D} \cap \mathscr{M} \text{ and } \operatorname{tr}(B) = 1\}. \tag{9.3.10}$$

Finally, let $(v_i \mid i \in V)$ with $v_i \in \mathbb{R}^N$ range over all orthonormal representations of the complementary graph $\overline{G}$, and $d \in \mathbb{R}^N$ over all vectors of Euclidean norm 1; define

$$\vartheta_4(G,w) := \max_{\{d,(v_i)\}} \sum_{i \in V} (d^T v_i)^2 w_i \,. \tag{9.3.11}$$

We now show the following result.

(9.3.12) Theorem. *For every graph $G = (V,E)$ and every $w \in \mathbb{R}_+^V$,*

$$\vartheta(G,w) = \vartheta_1(G,w) = \vartheta_2(G,w) = \vartheta_3(G,w) = \vartheta_4(G,w) \,.$$

Proof. First we remark that the theorem trivially holds if $w = 0$. So we may assume $w \neq 0$. The proof will consist of showing $\vartheta \leq \vartheta_1 \leq \vartheta_2 \leq \vartheta_3 \leq \vartheta_4 \leq \vartheta$. First we show

$$\vartheta(G,w) \leq \vartheta_1(G,w). \tag{9.3.13}$$

Choose a vector $x \in \mathrm{TH}(G)$ that maximizes the linear objective function $w^T x$. Let $(u_i \mid i \in V)$ with $u_i \in \mathbb{R}^N$ be any orthonormal representation of G and $c \in \mathbb{R}^N$ any vector with $\|c\| = 1$. Then by (9.3.5) and (9.3.2)

$$\begin{aligned} \vartheta(G,w) = w^T x &= \sum_{i \in V} w_i x_i \\ &\leq \left(\max_i \frac{w_i}{(c^T u_i)^2}\right) \sum_{i \in V} (c^T u_i)^2 x_i \\ &\leq \max_i \frac{w_i}{(c^T u_i)^2}. \end{aligned}$$

Here we have used the convention that if $c^T u_i = 0$, then $w_i/(c^T u_i)^2 = +\infty$ if $w_i > 0$, while $w_i/(c^T u_i)^2 = 0$ if $w_i = 0$. By (9.3.6), this proves (9.3.13).

Now we show

$$\vartheta_1(G,w) \leq \vartheta_2(G,w). \tag{9.3.14}$$

Choose a matrix $A \in \mathscr{M}^\perp$, and set $t := \Lambda(A+W)$. (Note that $t > 0$ since $\operatorname{tr}(A+W) = \operatorname{tr}(W) > 0$.) Then the matrix $tI - A - W$ is positive semidefinite. And so by (0.1.4) we can write $tI - A - W = X^T X$ with some matrix $X \in \mathbb{R}^{V \times V}$. Let

$x_i \in \mathbb{R}^V$ denote the column of X corresponding to $i \in V$. Then our definitions imply

$$x_i^T x_i = t - w_i \quad \text{for all } i \in V \tag{9.3.15}$$

and

$$x_i^T x_j = -\sqrt{w_i w_j} \quad \text{for all } i,j \text{ nonadjacent in } G. \tag{9.3.16}$$

Let $c \in \mathbb{R}^V$ be a vector with $\|c\| = 1$ and orthogonal to all x_i, $i \in V$ (such a vector exists since X is singular), and consider the vectors $u_i := \sqrt{w_i/t}\, c + t^{-1/2} x_i$. Then for each $i \in V$, we obtain from (9.3.15)

$$u_i^T u_i = \frac{w_i}{t} c^T c + \frac{1}{t} x_i^T x_i = 1$$

and from (9.3.16) for any two nonadjacent nodes i,j,

$$u_i^T u_j = \frac{\sqrt{w_i w_j}}{t} c^T c + \frac{1}{t} x_i^T x_j = 0\,.$$

Hence $(u_i \mid i \in V)$ is an orthonormal representation of G, and so by (9.3.6) and the definition of u_i

$$\vartheta_1(G,w) \leq \max_{i \in V} \frac{w_i}{(c^T u_i)^2} = \max_{i \in V} \frac{w_i}{w_i/t} = t = \Lambda(A + W).$$

Since this holds for each $A \in \mathcal{M}^\perp$, by (9.3.9) this proves (9.3.14).

Next we show

$$\vartheta_2(G,w) \leq \vartheta_3(G,w)\,. \tag{9.3.17}$$

This relation is the heart of the proof; it provides the good characterization of $\vartheta(G,w)$ by going from "min" to "max". By the definition of $\vartheta_3 := \vartheta_3(G,w)$ in (9.3.10) the inequality

$$\overline{w}^T B \overline{w} \leq \vartheta_3 \cdot \operatorname{tr}(B)$$

is valid for all matrices $B \in \mathcal{D} \cap \mathcal{M}$. But this can be viewed as a linear inequality for B in the space $\mathcal{S}$, namely the inequality can be written as

$$(W - \vartheta_3 I) \bullet B \leq 0,$$

where "$\bullet$" denotes the Euclidean inner product in $\mathbb{R}^{V \times V}$, i. e., $A \bullet B = \operatorname{tr}(A^T B)$. Note that $\mathcal{D}$ and $\mathcal{M}$ are cones. So $\mathcal{D} \cap \mathcal{M}$ is a cone, and recall that the cone polar to this cone in the linear space $\mathcal{S}$ is $(\mathcal{D} \cap \mathcal{M})^\circ := \{A \in \mathcal{S} \mid A \bullet B \leq 0 \text{ for all } B \in \mathcal{D} \cap \mathcal{M}\}$. Moreover, we know that $(\mathcal{D} \cap \mathcal{M})^\circ = \mathcal{D}^\circ + \mathcal{M}^\circ$; hence the inequality above gives

$$W - \vartheta_3 I \in \mathcal{D}^\circ + \mathcal{M}^\circ\,.$$

It is easy to see that $\mathscr{D}^\circ = -\mathscr{D}$ and $\mathscr{M}^\circ = \mathscr{M}^\perp$, and so we can write

$$W - \vartheta_3 I = -D - A,$$

where $D \in \mathscr{D}$ and $-A \in \mathscr{M}^\perp$. So $\vartheta_3 I - (A + W)$ is positive semidefinite, and hence ϑ_3 is not smaller than the largest eigenvalue of $A + W$. Thus by definition (9.3.9), we have

$$\vartheta_3 \geq \Lambda(A + W) \geq \vartheta_2(G, w),$$

which proves (9.3.17).

We now show

$$\vartheta_3(G, w) \leq \vartheta_4(G, w)\,. \tag{9.3.18}$$

Choose $B \in \mathscr{D} \cap \mathscr{M}$ such that $\mathrm{tr}(B) = 1$ and $\overline{w}^T B \overline{w} = \vartheta_3(G, w) =: \vartheta_3$. Since B is positive semidefinite, we can write it in the form $B = Y^T Y$ with $Y \in \mathbb{R}^{V \times V}$ – see (0.1.4). Let $e_i \in \mathbb{R}^V$ be the incidence vector of the singleton $\{i\}$, $i \in V$. Then $y_i := Y e_i$ is the i-th column of Y. Let $P := \{i \in V \mid y_i \neq 0\}$ and set $v_i := \frac{1}{\|y_i\|} y_i$ for all $i \in P$. Moreover, choose an orthonormal basis of $(\mathrm{lin}\{v_i \mid i \in P\})^\perp$ and let for each $i \in V \setminus P$ one of the elements of this basis represent i; call this vector v_i. Since $y_i^T y_j = b_{ij} = 0$ for all $i, j \in V$ that are adjacent, we see that $(v_i \mid i \in V)$ is an orthonormal representation of $\overline{G}$. Furthermore, $(Y\overline{w})^T Y \overline{w} = \overline{w}^T Y^T Y \overline{w} = \overline{w}^T B \overline{w} = \vartheta_3$ and hence $d := \frac{1}{\sqrt{\vartheta_3}} Y \overline{w}$ is a vector of unit length. Moreover, for $i \in P$, we have

$$d^T v_i = \frac{1}{\sqrt{\vartheta_3}\,\|y_i\|} \overline{w}^T Y^T Y e_i = \frac{1}{\sqrt{\vartheta_3}\,\|y_i\|} \overline{w}^T B e_i,$$

and so $\|y_i\| d^T v_i = \frac{1}{\sqrt{\vartheta_3}} \overline{w}^T B e_i$ holds for all $i \in V$. Hence

$$\sum_{i \in V} \|y_i\| \sqrt{w_i}\, d^T v_i = \frac{1}{\sqrt{\vartheta_3}} \sum_{i \in V} \overline{w}^T B e_i \sqrt{w_i} = \frac{1}{\sqrt{\vartheta_3}} \overline{w}^T B \overline{w} = \sqrt{\vartheta_3}\,.$$

Thus by the Cauchy-Schwarz inequality (0.1.26) and by (9.3.11)

$$\begin{aligned}
\vartheta_3 &= \Big(\sum_{i \in V} \|y_i\| \sqrt{w_i}\, d^T v_i\Big)^2 \\
&\leq \Big(\sum_{i \in V} \|y_i\|^2\Big)\Big(\sum_{i \in V} w_i (d^T v_i)^2\Big) \\
&= (\mathrm{tr}(B))\Big(\sum_{i \in V} w_i (d^T v_i)^2\Big) \\
&= \sum_{i \in V} w_i (d^T v_i)^2 \\
&\leq \vartheta_4(G, w)\,.
\end{aligned}$$

This proves (9.3.18).

Finally, we prove

(9.3.19) $$\vartheta_4(G,w) \leq \vartheta(G,w) .$$

Choose an orthonormal representation $(v_i \mid i \in V)$, $v_i \in \mathbb{R}^N$, of $\overline{G}$ and a vector $d \in \mathbb{R}^N$, $\|d\| = 1$, such that the maximum in the definition of $\vartheta_4(G,w)$ is achieved – see (9.3.11). We claim that the vector $((d^T v_i)^2 \mid i \in V)^T$ belongs to TH(G). To see this, let $(u_i \mid i \in V), u_i \in \mathbb{R}^N$, be any orthonormal representation of G and $c \in \mathbb{R}^N$ be any vector with unit length. We consider the matrices $u_i v_i^T \in \mathbb{R}^{N \times N}$. Note that these matrices are mutually orthogonal with respect to the inner product $\bullet$ in $\mathbb{R}^{N \times N}$ and have unit length, that is

$$(u_i v_i^T) \bullet (u_j v_j^T) = (u_i^T u_j)(v_i^T v_j) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

Similarly, the matrix cd^T has unit length with respect to this inner product. Hence

$$1 = (cd^T) \bullet (cd^T) \geq \sum_{i \in V} ((cd^T) \bullet (u_i v_i^T))^2 = \sum_{i \in V} (c^T u_i)^2 (d^T v_i)^2.$$

This shows that the vector $((d^T v_i)^2 \mid i \in V)^T$ is in TH(G), and so

$$\vartheta_4(G,w) = \sum_{i \in V} w_i (d^T v_i)^2 \leq \vartheta(G,w)$$

by the definition of $\vartheta(G,w)$ in (9.3.5).

The inequalities (9.3.13), (9.3.14), (9.3.17), (9.3.18), and (9.3.19) immediately imply Theorem (9.3.12). □

(9.3.20) Remarks. (a) Since we have equality in all estimates (9.3.13), (9.3.14), (9.3.17), (9.3.18), and (9.3.19), it follows that the inequalities in their proofs must be tight, and this yields a number of further structural properties of optimal orthogonal representations and optimal matrices. In particular, equality in (9.3.14) implies that for every graph G and every weighting $w : V \to \mathbb{R}_+$ there exists an orthonormal representation $(u_i \mid i \in V)$, $u_i \in \mathbb{R}^N$, of G and a vector $c \in \mathbb{R}^N$, $\|c\| = 1$, such that $\vartheta(G,w)(c^T u_i)^2 = w_i$ for all $i \in V$.

(b) It is easy to see from the previous considerations – details will follow in Section 9.4 – that we have also obtained a lower bound on the weighted chromatic number of a graph, namely the following inequality holds

$$\omega(G,w) \leq \vartheta(\overline{G},w) \leq \chi(G,w) .$$

In particular, we have

$$\omega(G) \leq \vartheta(\overline{G},\mathbf{1}) \leq \chi(G) .$$

□

The next lemma formulates a further property of orthonormal representations which will be needed in the sequel.

(9.3.21) Lemma. *Let* $(v_i \mid i \in V)$, $v_i \in \mathbb{R}^N$, *be an orthonormal representation of* $\overline{G}$ *and* $d \in \mathbb{R}^N$ *with* $\|d\| = 1$. *Let* $w : V \to \mathbb{R}_+$ *be any weighting of the nodes of* G, *and assume that*

$$\sum_{i \in V} (d^T v_i)^2 w_i = \vartheta(G, w).$$

Then

$$\sum_{i \in V} w_i (d^T v_i) v_i = \vartheta(G, w) d.$$

Proof. By (9.3.11) and (9.3.12) we have

$$\sum_{i \in V} w_i (y^T v_i)^2 \leq \vartheta(G, w)$$

for all unit vectors $y \in \mathbb{R}^N$, and so by assumption, the left hand side is maximized by $y = d$. But the left hand side can be written as a quadratic form in y as follows:

$$\sum_{i \in V} w_i (y^T v_i)^2 = y^T \Big(\sum_{i \in V} w_i v_i v_i^T \Big) y \,,$$

and so its maximum over all vectors y of unit length is attained when y is an eigenvector of $\sum_{i \in V} w_i v_i v_i^T$, and the maximum value is the corresponding eigenvalue. Hence d is an eigenvector of this matrix associated with the eigenvalue $\vartheta(G, w)$. This implies

$$\sum_{i \in V} w_i (d^T v_i) v_i = \Big(\sum_{i \in V} w_i v_i v_i^T \Big) d = \vartheta(G, w) d \,.$$

□

Let us mention some corollaries of Theorem (9.3.12). We will first show that there are a number of further ways to describe the convex set $\mathrm{TH}(G)$ defined in (9.3.3).

(9.3.22) Corollary. *Each of the following expressions defines* $\mathrm{TH}(G)$:

(a) $\mathrm{TH}(G) = \{ x \in \mathbb{R}_+^V \mid \vartheta(\overline{G}, x) \leq 1 \}$;

(b) $\mathrm{TH}(G) = \{ x \in \mathbb{R}_+^V \mid x^T y \leq \vartheta(G, y)$ *for all* $y \in \mathbb{R}_+^V \}$;

(c) $\mathrm{TH}(G) = \{ ((d^T v_i)^2 \mid i \in V) \mid$ *where* $(v_i \mid i \in V)$, $v_i \in \mathbb{R}^N$, *is an orthonormal representation of* $\overline{G}$ *and* $d \in \mathbb{R}^N$ *satisfies* $\|d\| = 1 \}$.

Proof. (a) follows if, using that $\vartheta = \vartheta_4$ by Theorem (9.3.12), we substitute

$$\vartheta(\overline{G}, x) = \max \sum_{i \in V} (c^T u_i)^2 x_i \,,$$

where $(u_i \mid i \in V)$, $u_i \in \mathbb{R}^N$, is an orthonormal representation of G and $c \in \mathbb{R}^N$, $\|c\| = 1$.

(b) Each $x \in \mathrm{TH}(G)$ satisfies $x^T y \leq \vartheta(G, y)$ by the definition of $\vartheta(G, y)$ in (9.3.5). On the other hand, setting for each orthonormal representation $(u_i \mid i \in V)$ and each vector c with $\|c\| = 1$, $y_i := (c^T u_i)^2$ and $y := (y_i \mid i \in V)$, we conclude from (9.3.6) and (9.3.12) that $\vartheta(G, y) \leq 1$. And thus the inequalities (9.3.2) are implied by the inequalities $x^T y \leq \vartheta(G, y)$ for all $y \in \mathbb{R}_+^V$.

(c) In the proof of inequality (9.3.19) we have already shown that $((d^T v_i)^2 \mid i \in V) \in \mathrm{TH}(G)$ for each orthonormal representation (v_i) of $\bar{G}$ and each unit vector d. Conversely, let $x \in \mathrm{TH}(G)$. Then by (a) $t := \vartheta(\bar{G}, x) \leq 1$. By Remark (9.3.20) (a), $\bar{G}$ has an orthonormal representation $(v_i \mid i \in V)$, $v_i \in \mathbb{R}^N$, and there is a vector $d \in \mathbb{R}^N$, $\|d\| = 1$ such that $t(d^T v_i)^2 = x_i$ for all $i \in V$. Let d_1 be a vector orthogonal to d, and all v_i, $i \in V$ (in any linear space containing $\mathbb{R}^N$) with $\|d_1\| = 1$, and set $d_0 := \sqrt{t}\, d + \sqrt{1-t}\, d_1$. Then $\|d_0\| = 1$ and $(d_0^T v_i)^2 = t(d^T v_i)^2 = x_i$ for all $i \in V$. This proves (c). □

Combining definition (9.3.3) of $\mathrm{TH}(G)$ with (9.3.22) (c) we get as an immediate consequence:

(9.3.23) Corollary. *The antiblocker of* $\mathrm{TH}(G)$ *is equal to* $\mathrm{TH}(\bar{G})$. □

We have already mentioned that $\mathrm{TH}(G)$ is not a polytope in general. We will now characterize its facets.

(9.3.24) Theorem. *If an inequality defines a facet of* $\mathrm{TH}(G)$, *then it is a positive multiple of one of the nonnegativity constraints* (9.1.1) *or of one of the clique constraints* (9.2.1).

Proof. Suppose that $a^T x \leq \alpha$ defines a facet of $\mathrm{TH}(G)$, and let z be a point in the relative interior of $F := \{x \in \mathrm{TH}(G) \mid a^T x = \alpha\}$. Then either $z_i = 0$ for some i, or $\vartheta(\bar{G}, z) = 1$ by (9.3.22) (a). In the first case $a^T x \leq \alpha$ is trivially equivalent to $x_i \geq 0$. So suppose that $\vartheta(\bar{G}, z) = 1$. Since $\vartheta = \vartheta_4$ by Theorem (9.3.12), there exists an orthonormal representation (u_i) of G and a unit vector c such that

$$\sum_{i \in V} z_i (c^T u_i)^2 = 1 .$$

Since by definition (9.3.3) the orthonormal representation constraint

$$\sum_{i \in V} x_i (c^T u_i)^2 \leq 1 \tag{9.3.25}$$

is satisfied by all $x \in \mathrm{TH}(G)$, it follows that (9.3.25) must be equivalent with $a^T x \leq \alpha$, and hence we may assume that $\alpha = 1$ and $a_i = (c^T u_i)^2$. We also see that

$$\sum_{i \in V} x_i (c^T u_i)^2 = 1 = \vartheta(\bar{G}, x) \tag{9.3.26}$$

for all $x \in F$, and thus by Lemma (9.3.21)

$$\sum_{i \in V} x_i (c^T u_i) u_i = c$$

holds for all $x \in F$, in other words,

$$\sum_{i \in V} x_i (c^T u_i)(u_i)_j = c_j$$

for all $j \in V$. Since F is $(n-1)$-dimensional, these equations must follow from the equation (9.3.26), and hence $c_j (c^T u_i)^2 = (c^T u_i)(u_i)_j$ for all $i,j \in V$. Hence for any $i \in V$ such that $c^T u_i \neq 0$ we have $(c^T u_i)c = u_i$. Since $\|u_i\| = \|c\| = 1$, this yields that $u_i = \pm c$. Clearly, we may assume that $u_i = c$.

So we see that for each i, either $c^T u_i = 0$ or $u_i = c$. Set $Q := \{ i \in V \mid u_i = c \}$. Then Q is a complete subgraph, since for any two nodes $i,j \in Q$, $u_i^T u_j = c^T c = 1 \neq 0$. So (9.3.25) is just the clique inequality

$$\sum_{i \in Q} x_i \leq 1 ,$$

which proves the theorem. □

(9.3.27) Corollary. TH(G) *is a polytope if and only if* G *is a perfect graph.*

Proof. If G is perfect then by definition STAB(G) = QSTAB(G). Thus by (9.3.4) STAB(G) = TH(G), which implies that TH(G) is a polytope.

Conversely, suppose that TH(G) is a polytope. Then TH(G) is the solution set of all inequalities determining a facet of TH(G). By Theorem (9.3.24), all these inequalities are either nonnegativity constraints (9.1.1) or clique constraints (9.2.1), and so TH(G) = QSTAB(G). By Corollary (9.3.23) TH($\overline{G}$) = abl(TH(G)), and hence TH($\overline{G}$) is also a polytope. Using the same arguments as above we conclude that TH($\overline{G}$) = QSTAB($\overline{G}$). But now STAB(G) = abl(QSTAB($\overline{G}$)) = abl(TH($\overline{G}$)) = TH(G) = QSTAB(G), and thus G is a perfect graph. □

Two immediate corollaries are the following:

(9.3.28) Corollary. TH(G) = STAB(G) *if and only if* G *is perfect.* □

(9.3.29) Corollary. TH(G) = QSTAB(G) *if and only if* G *is perfect.* □

The theorems and corollaries above show that in spite of its complicated definition, TH(G) does have nice and useful properties. For our purposes the following property of TH(G) is the most important.

(9.3.30) Theorem. *The weak optimization problem for* TH(G) *is solvable in polynomial time.*

It is necessary to restrict our attention to weak optimization, because TH(G) is not a polyhedron in general and so the results of Chapter 6 do not apply. To see an example, let G be the pentagon C_5 and let $x_0 \in \mathbb{R}^5$ be the vector maximizing the linear objective function $\mathbb{1} = (1,1,1,1,1)^T$ over TH(C_5). By symmetry, we may assume that $x_0 = (t,t,t,t,t)^T$ for some $t > 0$. So

$$\mathbb{1}^T x \leq \mathbb{1}^T x_0 = 5t$$

is valid for all $x \in \mathrm{TH}(G)$ and so the vector $\frac{1}{5t}\mathbb{1}$ belongs to the antiblocker of TH(C_5). But C_5 is self-complementary and hence $\frac{1}{5t}\mathbb{1} \in \mathrm{TH}(C_5)$ by Corollary (9.3.23). Thus $\mathbb{1}^T\left(\frac{1}{5t}\mathbb{1}\right) \leq 5t$, whence $t \geq \frac{1}{\sqrt{5}}$. As remarked before, the orthonormal representation constraint derived from the orthonormal representation of C_5 shown in Figure 9.3 reads $\sum_{i \in C_5} x_i \leq \sqrt{5}$. This implies $t = \frac{1}{\sqrt{5}}$. So neither the maximizing vector nor the maximum value are rational.

Another preliminary remark : it would not do much good to apply the optimization-separation equivalence at this point. In fact, the separation problem for TH(G) is equivalent to the violation problem for its antiblocker; but this is just TH($\overline{G}$), and so all we can achieve by this is to reduce the optimization (or violation) problem for TH(G) to the same problem for the complementary graph.

Proof of (9.3.30). Since TH(G) is closed and convex, and it is trivial to find inscribed and circumscribed balls, it suffices to solve the weak validity problem.

So let $c^T x \leq \gamma$ be any inequality, $c \in \mathbb{Q}^V$, $\gamma \in \mathbb{Q}$. If $c_i \leq 0$ for some $i \in V$ then clearly it suffices to check whether, dropping the i-th coordinate, the remaining inequality is valid for TH($G - i$). So we may assume that $c > 0$.

By the definition of ϑ, our task is done if we can compute $\vartheta(G,c)$ in polynomial time (in the sense of computing real numbers in polynomial time). For this, we can use formula (9.3.10):

$$\vartheta(G,c) = \vartheta_3(G,c) = \max\{\bar{c}^T B \bar{c} \mid B \in \mathscr{D} \cap \mathscr{M} \text{ and } \mathrm{tr}(B) = 1\},$$

where $\mathscr{D}$ and $\mathscr{M}$ are the sets defined in (9.3.7). To see that this maximum is indeed computable in polynomial time, we apply Yudin and Nemirovskii's theorem (4.3.2). For this, consider the affine subspace

$$\mathscr{M}_1 := \{B \in \mathscr{M} \mid \mathrm{tr}(B) = 1\}$$

and the set

$$\mathscr{K} := \mathscr{M}_1 \cap \mathscr{D}\,.$$

So we have $\vartheta(G,c) = \max\{\bar{c}^T B \bar{c} \mid B \in \mathscr{K}\}$.

Claim 1. $\mathscr{K}$ is convex.

This follows since both $\mathscr{M}_1$ and $\mathscr{D}$ are convex.

Claim 2. $\mathscr{K} \subseteq S(0,n)$.

Indeed, let $B = (b_{ij}) \in \mathscr{K}$. Since B is positive semidefinite, $b_{ii} \geq 0$. Since $\operatorname{tr} B = 1$, we also have $b_{ii} \leq 1$. Using that B is positive semidefinite again, we have that $b_{ii}b_{jj} - b_{ij}^2 \geq 0$, and hence $|b_{ij}| \leq \sqrt{b_{ii}b_{jj}} \leq 1$. Hence $\|B\|_{\max} \leq n$, and so by (0.1.24) $B \in S(0, n)$.

Claim 3. $S\left(\frac{1}{n}I, \frac{1}{n}\right) \cap \mathscr{M}_1 \subseteq \mathscr{K}$.

For, let $B \in \mathscr{M}_1$ be any symmetric matrix in $\mathbb{R}^{V \times V}$ such that $\left\|\frac{1}{n}I - B\right\|_2 \leq \frac{1}{n}$. Then by (0.1.25) the largest eigenvalue of $\frac{1}{n}I - B$ is at most $\frac{1}{n}$. So the smallest eigenvalue of B is at least 0. Thus by (0.1.4) $B \in \mathscr{D}$, and so $B \in \mathscr{D} \cap \mathscr{M}_1 = \mathscr{K}$.

The first three claims show that $\mathscr{K}$ is a centered, well-bounded, convex body in $\mathscr{M}_1$. In what follows, we shall work in the space $\mathscr{M}_1$. (To be precise, we project $\mathscr{M}_1$ (and $\mathscr{K}$) on the space $\mathbb{R}^S$ where $S = \{(i,j) \mid ij \notin E,\ i \leq j,\ (i,j) \neq (n,n)\}$.)

Claim 4. The weak membership problem for $\mathscr{K}$ is solvable in polynomial time.

All we have to do is to check whether a given symmetric matrix B is positive semidefinite. There are various characterizations of positive semidefiniteness – see (0.1.4) – which can be used for the design of an efficient proof of this property.

For instance, polynomial time algorithms can be obtained from Gaussian elimination and Cholesky decomposition. In Gaussian elimination we allow pivots on the main diagonal only; if the rank of the matrix is found and only positive pivots have been carried out, then the matrix is positive semidefinite. Cholesky decomposition can be used in a similar way. A further method is to compute the smallest eigenvalue λ_n; if λ_n is nonnegative then the matrix is positive semidefinite. This algorithm may be fast in practice but it is not so easy to implement it in polynomial time.

Let us describe in detail the algorithm based on Gaussian elimination – see Section 1.4. To check positive semidefiniteness we use criterion (0.1.4) (iv). So let $B = (b_{ij})$ be a symmetric rational $n \times n$-matrix. If $b_{11} < 0$ then B is not positive semidefinite. If $b_{11} = 0$ but $b_{1i} \neq 0$ for some $2 \leq i \leq n$ then B is not positive semidefinite (in fact, its principal submatrix $\begin{pmatrix} b_{11} & b_{1i} \\ b_{i1} & b_{ii} \end{pmatrix}$ has determinant $-b_{1i}^2 < 0$). If $b_{11} = 0$ and also $b_{1i} = 0$ for all i, then we can drop the first row and column and restrict our attention to the remaining $(n-1) \times (n-1)$-matrix. If $b_{11} > 0$ then pivot on b_{11}, i. e., consider the matrix $B' = (b'_{ij})_{i,j=2}^n$ where $b'_{ij} := b_{ij} - \frac{b_{i1}}{b_{11}} b_{1j}$. Then B' is positive semidefinite if and only if B is, and so again we have reduced our task to an $(n-1) \times (n-1)$ problem.

It has to be checked, however, that the entries of the smaller matrices obtained this way do not grow too large, i. e., their encoding lengths remain bounded by the encoding length of B. This is easy to see, however, since these entries are subdeterminants of the original matrix B – see the proof of Theorem (1.4.8). □

(9.3.31) Remark. One could also use formula (9.3.9)

$$\vartheta(G, w) = \min\{\Lambda(A + W) \mid A \in \mathscr{M}^{\perp}\}$$

to obtain $\vartheta(G, w)$. Namely, $\Lambda(A + W)$ is a convex function of A, and A ranges through a linear space. So we obtain an unconstrained convex function minimization problem. Using standard linear algebra, one can design an evaluation oracle

for $\Lambda(A+W)$ and also derive a bound R on the Frobenius norm of the minimizing matrix. Then Theorem (4.3.13) can be applied to compute $\vartheta(G,w)$. □

Corollary (9.3.27), Theorem (9.3.30), and Theorem (6.3.2) immediately give :

(9.3.32) Corollary. *The weighted stable set problem for perfect graphs is solvable in polynomial time.* □

Trivially, we also obtain the following:

(9.3.33) Corollary. *The weighted clique problem for perfect graphs is solvable in polynomial time.* □

*9.4 Coloring Perfect Graphs

We shall now describe how the results of Section 9.3 can be used to solve further optimization problems for perfect graphs in polynomial time. Recall the following two problems.

(9.4.1) Minimum Weighted Coloring Problem. *Given a graph $G=(V,E)$ with node weights $w : V \to \mathbb{Z}_+$, find stable sets $S_1, \ldots, S_k \subseteq V$ and nonnegative integers $\lambda_1, \ldots, \lambda_k$ such that for each node $v \in V$ the sum $\sum_{i,v\in S_i} \lambda_i$ is not smaller than $w(v)$ and such that $\lambda_1 + \ldots + \lambda_k$ is as small as possible.*

(9.4.2) Minimum Weighted Clique Covering Problem. *Given a graph $G=(V,E)$ with node weights $w : V \to \mathbb{Z}_+$, find cliques $Q_1, \ldots, Q_k \subseteq V$ and nonnegative integers $\lambda_1, \ldots, \lambda_k$ such that for each node $v \in V$ the sum $\sum_{i,v\in Q_i} \lambda_i$ is not smaller than $w(v)$ and such that $\lambda_1 + \ldots + \lambda_k$ is as small as possible.*

As before, we denote the optimum value of (9.4.1) by $\chi(G,w)$ and the optimum value of (9.4.2) by $\bar{\chi}(G,w)$. Clearly, the coloring problem (9.4.1) for G is nothing but the clique covering problem (9.4.2) for the complementary graph $\bar{G}$, and vice versa. Both problems are $\mathcal{NP}$-hard for general graphs. We will now show that they are solvable in polynomial time for perfect graphs. It suffices to discuss problem (9.4.2).

(9.4.3) Theorem. *There is an algorithm that solves the minimum weighted clique covering problem for perfect graphs in polynomial time.*

Proof. An integer programming formulation of (9.4.2) can be given as follows.

$$\text{(9.4.4)} \qquad \min \sum_{Q\subseteq V \text{ clique}} \lambda_Q$$

$$\text{(i)} \qquad \sum_{Q \text{ clique},\, Q\ni v} \lambda_Q \geq w(v) \qquad \text{for all } v \in V,$$

$$\text{(ii)} \qquad \lambda_Q \geq 0 \qquad \text{for all cliques } Q \subseteq V,$$

$$\text{(iii)} \qquad \lambda_Q \text{ integral} \qquad \text{for all cliques } Q \subseteq V.$$

The linear program (9.4.4) without the integrality stipulation (iii) is the dual of the linear program

$$\max w^T x\,, \qquad x \in \mathrm{QSTAB}(G)\,, \tag{9.4.5}$$

where QSTAB(G) is the polytope defined by the nonnegativity and clique constraints – see (9.2.7). (By Theorem (9.2.9), problem (9.4.5) is $\mathcal{NP}$-hard in general, hence even the LP-relaxation of (9.4.4) is $\mathcal{NP}$-hard.) By (9.3.28) and (9.3.29) we have QSTAB(G) = STAB(G) = TH(G) if and only if G is perfect, and thus by (9.3.32), an optimum solution of (9.4.5) can be found in polynomial time if G is a perfect graph.

Therefore, our general results of Chapter 6 imply that we can find an optimum solution of the linear program dual to (9.4.5) in polynomial time, and thus an optimum solution of (9.4.4) without the integrality condition (iii) can be computed in polynomial time, in case G is a perfect graph. But by Theorem (9.2.4), if G is a perfect graph, the integrality condition (iii) of (9.4.4) is superfluous, i. e., the linear program (9.4.4) (i), (ii) always has an optimum solution which is integral. Our method, however, for finding an optimum dual solution is only guaranteed to find a basic optimum solution, and since (9.4.4) without (iii) may have nonintegral basic optimum solutions (in addition to the integral ones) we are not sure that the general techniques solve (9.4.4). But in this special case it is possible to construct an integral optimum solution from a fractional one. This goes as follows.

By finding a basic optimal solution of the LP-dual of (9.4.5) we obtain a basic optimum solution of (9.4.4) without (iii). It is a trivial matter to modify this solution such that it is optimal for

$$\min \sum_{Q \subseteq V \text{ clique}} \lambda_Q \tag{9.4.6}$$

$$\text{(i)} \qquad \sum_{Q \text{ clique},\, Q \ni v} \lambda_Q = w(v) \qquad \text{for all } v \in V,$$

$$\text{(ii)} \qquad \lambda_Q \geq 0 \qquad \text{for all cliques } Q \subseteq V.$$

Let Q_1 be any clique such that $\lambda_{Q_1} > 0$ in this optimum solution. Because of condition (i) of (9.4.6) we have $w(q) > 0$ for all $q \in Q_1$. Moreover by complementary slackness, Q_1 has a nonempty intersection with all stable sets $S \subseteq V$ satisfying $w(S) = \alpha(G, w)$.

We define $w'(v) := w(v)$ for all $v \in V \setminus Q_1$, and using the algorithm of Corollary (9.3.32), we compute $\alpha(G - Q_1, w')$. We set

$$\mu_{Q_1} := \min\{\alpha(G, w) - \alpha(G - Q_1, w'), \min\{w(q) \mid q \in Q_1\}\}\,.$$

Clearly, μ_{Q_1} is a positive integer. Consider the new weighting

$$w_1 := w - \mu_{Q_1} \chi^{Q_1}\,. \tag{9.4.7}$$

By our choice of μ_{Q_1} we have $w_1 \geq 0$, and our construction yields

$$\alpha(G, w_1) = \alpha(G, w) - \mu_{Q_1}.$$

To see that this equation holds, observe first that $\alpha(G, w_1) \geq \alpha(G, w) - \mu_{Q_1}$, since there exists a stable set S with $\alpha(G, w) = w(S) \leq w_1(S) + \mu_{Q_1} \leq \alpha(G, w_1) + \mu_{Q_1}$. To show the converse inequality, let S be any stable set satisfying $w_1(S) = \alpha(G, w_1)$. If $S \cap Q_1 = \emptyset$ then $\alpha(G, w_1) = w_1(S) = w'(S) \leq \alpha(G - Q_1, w') \leq \alpha(G, w) - \mu_{Q_1}$. If $S \cap Q_1 \neq \emptyset$ then $\alpha(G, w_1) = w_1(S) \leq w(S) - \mu_{Q_1} \leq \alpha(G, w) - \mu_{Q_1}$.

If $w_1 \neq 0$ we repeat this procedure, with w replaced by w_1.

This algorithm ends up with a list of cliques $Q_1, \ldots, Q_k$ and of positive integers $\mu_{Q_1}, \ldots, \mu_{Q_k}$ such that

$$\mu_{Q_1}\chi^{Q_1} + \ldots + \mu_{Q_k}\chi^{Q_k} = w$$
$$\mu_{Q_1} + \ldots + \mu_{Q_k} = \alpha(G, w).$$

Thus we have found an integral solution of (9.4.6) and hence a solution of (9.4.4). Clearly, each step of the iteration described above can be executed in polynomial time. To finish the running time analysis of our algorithm we prove the following:

Claim. The vectors $\chi^{Q_1}, \ldots, \chi^{Q_k}$ are linearly independent.

We show this by proving that for each i, $1 \leq i \leq k$, there exists a vector orthogonal to $\chi^{Q_{i+1}}, \ldots, \chi^{Q_k}$ but not to χ^{Q_i}. Let us denote the vector w_1 defined in (9.4.7) in the i-th iteration of the algorithm by w_i, $i = 1, \ldots, k$. By the definition of μ_{Q_i} there are two cases. Either there is a node $q \in Q_i$ such that $w_i(q) = 0$ or there exists a stable set $S \subseteq V(G) \setminus Q_i$ such that $w_i(S) = \alpha(G, w_i)$. In the first case, $w_j(q) = 0$ for all $i \leq j \leq k-1$ and hence $q \notin Q_{j+1}$. So $\chi^{\{q\}}$ is orthogonal to $\chi^{Q_{i+1}}, \ldots, \chi^{Q_k}$ but not to χ^{Q_i}. In the second case observe that – by construction – S is a maximum weight stable set of G not only with respect to the weighting w_i but also with respect to $w_{i+1}, \ldots, w_{k-1}$. Hence by complementary slackness $|S \cap Q_{j+1}| = 1$ for $j = i, \ldots, k-1$. If S_0 is a maximum weight stable set of G with respect to the initial weighting w then $|S_0 \cap Q_j| = 1$ for all j, and hence $\chi^{S_0} - \chi^S$ is a vector orthogonal to $\chi^{Q_{i+1}}, \ldots, \chi^{Q_k}$ but not to χ^{Q_i}. This proves the claim.

It follows from our claim that $k \leq n$, and hence the algorithm terminates in polynomial time. Moreover, the claim shows that $\mu_{Q_1}, \ldots, \mu_{Q_k}$ yields a basic optimum dual solution of (9.4.5). □

Since the coloring problem (9.4.1) for a graph G is the clique covering problem (9.4.2) for the complementary graph $\overline{G}$, and since a graph is perfect if and only if its complement is perfect we obtain :

(9.4.8) Corollary. *There is an algorithm that solves the minimum weighted coloring problem for perfect graphs in polynomial time.* □

*9.5 More Algorithmic Results on Stable Sets

In this last section we survey some further classes of graphs for which the stable set problem can be solved in polynomial time.

Given any class $\mathscr{L}$ of valid inequalities for STAB(G), we can consider the polyhedron

$$\mathscr{L}\,\mathrm{STAB}(G) := \{ x \in \mathbb{R}_+^V \mid x \text{ satisfies } \mathscr{L} \}\,, \tag{9.5.1}$$

and ask ourselves the question whether the optimization problem for $\mathscr{L}\,\mathrm{STAB}(G)$ can be solved in polynomial time. By the general results of Chapters 4 and 6, this is equivalent to checking whether or not a given vector $y \in \mathbb{R}_+^V$ belongs to $\mathscr{L}\,\mathrm{STAB}(G)$, i. e., whether or not y satisfies all the inequalities in $\mathscr{L}$. If this problem can be solved in polynomial time, then the stable set problem can be solved in polynomial time for all graphs G such that $\mathscr{L}\,\mathrm{STAB}(G) = \mathrm{STAB}(G)$. Let us call a graph G **$\mathscr{L}$-perfect** if $\mathscr{L}\,\mathrm{STAB}(G) = \mathrm{STAB}(G)$.

The significance of such a result depends on whether or not there are any interesting examples of $\mathscr{L}$-perfect graphs. In Sections 9.1, 9.2, and 9.3 we have studied two classes $\mathscr{L}$ of valid inequalities for which such examples do exist. Here we shall mention some further classes for which the membership problem for $\mathscr{L}\,\mathrm{STAB}(G)$ can be solved in polynomial time, even though we do not know really interesting classes of $\mathscr{L}$-perfect graphs.

As a first remark, note that if the membership problem is solvable for $\mathscr{L}_1\,\mathrm{STAB}(G)$ and $\mathscr{L}_2\,\mathrm{STAB}(G)$ in polynomial time then it is also solvable for $\mathscr{L}_1\,\mathrm{STAB}(G) \cap \mathscr{L}_2\,\mathrm{STAB}(G) = (\mathscr{L}_1 \cup \mathscr{L}_2)\,\mathrm{STAB}(G)$. So we may take the nonnegativity constraints (9.1.1) and the odd circuit constraints (9.1.4) together with the orthonormal representation constraints (9.3.2) and obtain a convex set containing STAB(G) for which the optimization problem can be solved in polynomial time. The corresponding "perfect" graphs are defined by

$$\mathrm{TH}(G) \cap \mathrm{CSTAB}(G) = \mathrm{STAB}(G)\,. \tag{9.5.2}$$

It follows from Theorem (9.3.24) that this condition is equivalent to the following:

$$\mathrm{QSTAB}(G) \cap \mathrm{CSTAB}(G) = \mathrm{STAB}(G)\,. \tag{9.5.3}$$

Graphs satisfying (9.5.3) are called **h-perfect**; these are those graphs for which the nonnegativity, clique, and odd circuit constraints suffice to describe STAB(G). Since every h-perfect graph also satisfies (9.5.2), we obtain the following.

(9.5.4) Corollary. *The stable set problem for h-perfect graphs can be solved in polynomial time.* □

We do not know any interesting classes of h-perfect graphs that are neither perfect, nor t-perfect, nor combinations of these. But, of course, there are such graphs. For instance, the graph in Figure 9.4 is h-perfect but neither perfect nor t-perfect.

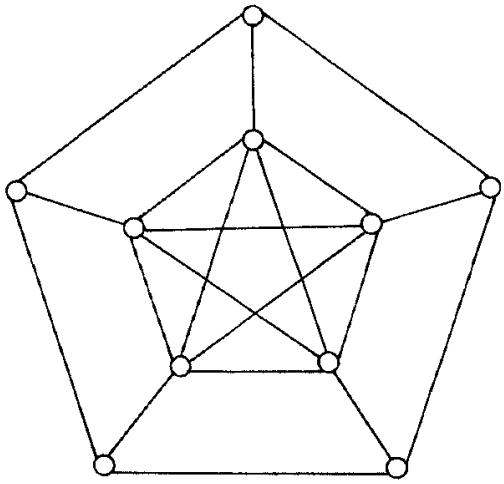


Figure 9.4

The **wheel constraints** for STAB(G) are determined by the **odd wheels** in G, i. e., induced subgraphs $G[W]$ of G containing a node w (the **center**) such that w is adjadent to all other nodes of $G[W]$ and $G[W]-w$ is an odd circuit. Figure 9.5 shows a wheel, where $G[W]-w$ is a 7-circuit.

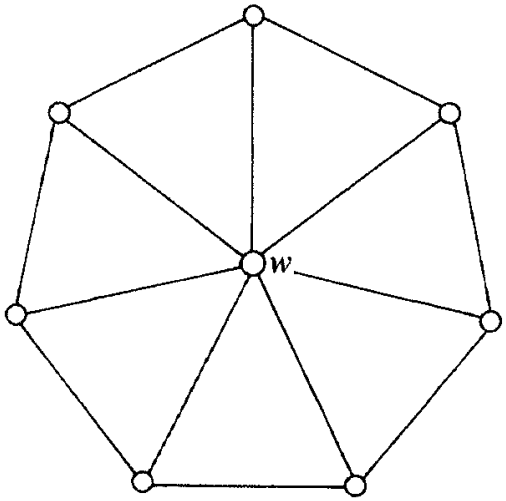


Figure 9.5

Given such a subgraph $G[W]$, where $|W|$ is even, we can write up the following **odd wheel constraint**:

$$x(W \setminus \{w\}) + \tfrac{|W|-2}{2} x_w \leq \tfrac{|W|-2}{2}. \tag{9.5.5}$$

Note that (9.5.5) is stronger than the odd circuit constraint determined by $G[W]-w$. Let $\mathscr{W}$ denote the class of constraints (9.1.1), (9.1.2), (9.1.4), (9.5.5). Then we have

(9.5.6) Theorem. *The strong separation (and the strong optimization) problem for* $\mathscr{W}\,\mathrm{STAB}(G)$ *can be solved in polynomial time.*

Proof. Let $y \in \mathbb{Q}^V$. We know that we can check in polynomial time whether y satisfies the nonnegativity constraints (9.1.1), the edge constraints (9.1.2), and the odd circuit constraints (9.1.4). Suppose that we find that it does. Then define, for each node $w \in V$, a weighting of the edges spanned by the set $\Gamma(w)$ of neighbors of w, by

$$l_w(uv) := 1 - y_u - y_v - y_w \qquad \text{for all } uv \in E \text{ with } u, v \in \Gamma(w).$$

Since y satisfies the odd circuit constraints, we know that $l_w(uv) \geq 0$. So for each node $w \in V$, we can find, by the algorithm solving problem (8.3.6), an odd circuit of $(\Gamma(w), E(\Gamma(w)))$ with smallest l_w-length. Suppose that we find this way an odd circuit $(u_0, \ldots, u_{2k})$ with l_w-length less than $1 - y(w)$. This means

$$\sum_{i=0}^{2k} (1 - y(u_i) - y(u_{i+1}) - y(w)) < 1 - y(w)$$

and so

$$\sum_{i=0}^{2k} y(u_i) + ky(w) > k.$$

Hence, y violates an odd wheel constraint. Conversely, if for every $w \in V$ every odd circuit in $\Gamma(w)$ has l_w-length at least $1 - y(w)$, then clearly $y \in \mathscr{W}\,\mathrm{STAB}(G)$. □

It follows that the stable set problem can be solved in polynomial time for $\mathscr{W}$-perfect graphs. Unfortunately, we are not aware of any interesting examples of such graphs.

There are further classes of inequalities known that are valid for STAB(G) or define facets – see for instance Padberg (1973), Nemhauser and Trotter (1974), or – for a survey – Balas and Padberg (1975). For most of these classes we do not know whether or not the associated separation problem can be solve in polynomial time. An example of such a class is the class of **odd antihole inequalities**. These are inequalities of the form $x(W) \leq 2$ where the subgraph of G induced by W is the complement of a chordless odd circuit of length at least five.

The papers Nemhauser and Trotter (1974) and Padberg (1975) describe general methods – so-called lifting techniques – with which, for each induced subgraph G' of G, valid (or facet defining) inequalities for STAB(G') can be turned into valid (or facet defining) inequalities for STAB(G). (These methods apply to integral polyhedra in general.) In fact, the odd wheel constraints can be obtained from the odd circuit constraints with this lifting technique, and we are able to modify the trick described in the proof of Theorem (9.5.6) to solve the separation problem for further classes of lifted inequalities. Similarly, we can solve the separation problem for the separation problem for $\mathscr{L}\,\mathrm{STAB}(G)$ if $\mathscr{L}$ consists of

constraints derived from odd subdivisions of K_4, or, more generally, from odd subdivisions of wheels. But a general theory of such classes of inequalities is not known. It is also an open question whether the more involved linear algebraic methods of Section 9.3 can be pushed any further.

We conclude with some classes of graphs for which the stable set problem is polynomially solvable, although not directly by polyhedral methods.

Since the stable set problem for the line graph of a graph is trivially equivalent to the matching problem for the original graph, it is solvable in polynomial time (see Section 7.3). MINTY (1980) and SBIHI (1978, 1980) extended this result to the stable set problem of claw-free graphs, i. e., graphs not containing a 3-star as an induced subgraph. (It is easy to see that every line graph is claw-free, but not vice versa.) Minty also solved the weighted version in polynomial time. However, in spite of considerable efforts, no decent system of inequalities describing STAB(G) for claw-free graphs is known – cf. GILES and TROTTER (1981).

Let us remark that for some interesting classes of graphs, the number of all maximal stable sets is polynomial in $|V|$, and all such sets can be actually listed in polynomial time :

complements of triangulated graphs,
complements of bipartite graphs, and
complements of line-graphs

are some examples. Now, if $G = (V, E)$ is a graph whose nodes can be partitioned into two classes V_1 and V_2 such that both $G[V_1]$ and $G[V_2]$ belong to one of the classes above, then the stable set problem for G is easily solved in polynomial time, by solving it for all bipartite graphs $G[S_1 \cup S_2]$, where S_i is a maximal stable set in $G[V_i]$.

One of the fastest branch and bound methods for the cardinality version of the stable set (resp. clique) problem in arbitrary graphs known to date is based on observations of this type. The algorithm is due to BALAS and YU (1984). It iteratively generates maximal triangulated induced subgraphs or complements of these and heavily exploits the fact that maximum stable sets in such graphs can be computed in time linear in $|V|+|E|$. The algorithm terminates in polynomial time for instance for complements of triangulated graphs, but also for some classes of graphs that include some imperfect graphs. BALAS, CHVÁTAL and NEŠETŘIL (1985) have modified and extended these ideas to obtain algorithms that solve the weighted stable set problem for several further classes of graphs in polynomial time. The common feature of these classes of graphs is that every stable set of any of these graphs is contained in some member of a polynomial-sized collection of induced subgraphs that are bipartite.

HSU, IKURA and NEMHAUSER (1981) showed that for each fixed k the weighted stable set problem can be solved in polynomial time by enumerative methods if the longest odd cycle in the graph is bounded by k.

It follows from the results of Chapter 6 that, for each of the classes of graphs mentioned above, the facets of STAB(G) can in a sense be described : we can decide in polynomial time whether or not a given inequality is valid for STAB(G), and also whether or not it determines a facet of STAB(G). Moreover, we can

represent every valid inequality as a nonnegative combination of facet defining inequalities of STAB(G). We can also find, for any vector not in STAB(G), a facet separating it from STAB(G) in polynomial time. But of course, it would be interesting to find a direct description of STAB(G) by linear inequalities and then use this to solve the stable set problem. For none of the three classes of graphs mentioned above, such a description of STAB(G) is known.

*Chapter 10

Submodular Functions

The concept of a **submodular function** in discrete optimization appears to be in several respects analogous to that of a convex function in continuous optimization. In many combinatorial theorems and problems, submodularity is involved, in one form or another, and submodularity often plays an essential role in a proof or an algorithm. Moreover, analogous to the fast methods for convex function minimization, it turns out that submodular functions can also be minimized fast, viz. in polynomial time. However, the only method known for this is, as yet, the ellipsoid method.

Submodular functions were studied first in relation to matroids (RADO (1942), INGLETON (1959)) and lattices (DILWORTH (1944)). Its importance for optimization was revealed by RADO (1957) and EDMONDS (1970), who proved several fundamental theorems and introduced the corresponding geometric concept of **polymatroid**. (The terminology is not uniform and alternative names have been proposed for submodular functions and polymatroids in the literature.)

In this chapter we give a survey on submodular functions and polymatroids, and we discuss methods to handle associated algorithmic problems with the ellipsoid method.

Let us remark that all polynomial time algorithms for "primal" problems presented in this chapter can be turned into strongly polynomial ones using the general theory developed in Section 6.6. Some difficulties arise in connection with minimizing a submodular function, but these can be overcome as will be explained. As usual, difficulties occur with "dual" problems. In some cases we will use ad hoc methods (based on making collections of sets "cross-free"), to turn the algorithms into strongly polynomial ones.

*10.1 Submodular Functions and Polymatroids

Let E be a finite set. A function $f : 2^E \to \mathbb{R}$ is called **submodular** on 2^E if

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T) \quad \text{for all } S, T \subseteq E. \tag{10.1.1}$$

The main question studied in this chapter is how to find the minimum of such a function. Let us describe some examples of submodular functions and illustrate that a variety of combinatorial optimization problems are special cases of submodular function minimization.

(10.1.2) Examples.

I. Capacity of cuts. Let $D = (V, A)$, be a directed graph, and let $c : A \to \mathbb{Q}_+$ be a "capacity" function. Choose $r, s \in V$, and let $E := V \setminus \{r, s\}$. Define, for $S \subseteq E$,

$$f(S) := c(\delta^+(S \cup \{r\})),$$

i. e., let $f(S)$ be the capacity of the cut of D determined by $S \cup \{r\}$. Then f is submodular on 2^E, as one easily checks. Minimizing f means finding a cut of minimum capacity separating r and s – see (8.3.8).

II. Number of neighbors. Let $G = (V, E)$ be a bipartite graph, with color classes S and T. Let $\Gamma(U)$ denote the set of neighbors of nodes in U. Then the function $|\Gamma(U)|$ is submodular on 2^S, as can be easily seen. Hence so is the function

$$f(U) := |S \setminus U| + |\Gamma(U)|.$$

Moreover, the minimum value of f is equal to the node covering number $\tau(G)$ which, by König's matching theorem (8.2.3), is equal to the matching number $\nu(G)$.

III. Supply of neighbors. Let $G = (V, E)$ be a bipartite graph with color classes S and T. Let $b : T \to \mathbb{Q}_+$ describe the "supply" available at node $t \in T$ – cf. (8.2.11). Define, for $U \subseteq S$,

$$f(U) := b(\Gamma(U)).$$

Then f is a submodular function on 2^S. The value $f(U)$ may be interpreted as the maximum amount of goods the set U can receive over the given edges of G. Let $d : S \to \mathbb{Q}_+$ describe the demand at node $s \in S$. Recall that d is called satisfiable if there exists a function $y : E \to \mathbb{Q}_+$ such that

$$\sum_{e \ni s} y(e) = d(s) \quad \text{for all } s \in S,$$
$$\sum_{e \ni t} y(e) \leq b(t) \quad \text{for all } t \in T.$$

By Theorem (8.2.15), a vector d is satisfiable if and only if

$$d(U) \leq f(U) \quad \text{for all } U \subseteq S.$$

These inequalities can be tested by minimizing the submodular function

$$\tilde{f}(U) := f(U) - d(U) \quad \text{over all } U \subseteq S.$$

IV. Rank function of a matroid. Let $M = (E, \mathcal{I})$ be a matroid, with rank function r. Then r is a submodular function – see (7.5.10). Recall that the convex hull $\mathrm{IND}(M)$ of the incidence vectors of the independent sets in M is given by the inequalities (7.5.13) and (7.5.14):

$$x_e \geq 0 \quad \text{for all } e \in E,$$
$$x(F) \leq r(F) \quad \text{for all } F \subseteq E.$$

Given a vector $y \in \mathbb{Q}^E$ we want to check whether y is in IND(M). Checking the nonnegativity constraints is trivial. To check the "rank inequalities", note that the function

$$\tilde{r}(F) := r(F) - y(F)$$

is submodular. We want to know whether the minimum value of $\tilde{r}$ is nonnegative. If we can find the minimum value of $\tilde{r}$ we are done.

V. Mixed rank function of two matroids. Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be matroids with rank functions r_1 and r_2, respectively. Since both r_1 and r_2 are submodular, also the function $f : 2^E \to \mathbb{Z}_+$ defined by

$$f(S) := r_1(S) + r_2(E \setminus S) \quad \text{for all } S \subseteq E$$

is submodular. A theorem of EDMONDS (1970) – see Corollary (7.5.17) – states that the minimum value of f is equal to the maximum cardinality of a set in $\mathcal{I}_1 \cap \mathcal{I}_2$. □

Two polyhedra can be associated with a submodular function $f : 2^E \to \mathbb{R}$ in a natural way:

$$P_f := \{x \in \mathbb{R}^E \mid x(F) \leq f(F) \text{ for all } F \subseteq E,\ x \geq 0\}, \tag{10.1.3}$$

$$EP_f := \{x \in \mathbb{R}^E \mid x(F) \leq f(F) \text{ for all } F \subseteq E\}. \tag{10.1.4}$$

The polytope (10.1.3) is called the **polymatroid** P_f associated with f, while the polyhedron (10.1.4) we call the **extended polymatroid** EP_f associated with f. Note that $EP_f \neq \emptyset$ iff $f(\emptyset) \geq 0$, whereas $P_f \neq \emptyset$ iff $f \geq 0$.

Optimizing linear objective functions over extended polymatroids is an easy task. It can be done by a natural extension of the greedy algorithm (7.5.11) which can be formalized as follows. Let $X \subseteq \mathbb{R}^n$ and $c \in \mathbb{Q}^n_+$. Let us order the coordinates of c so that $c_1 \geq c_2 \geq \ldots \geq c_n$. Find successively

$$\begin{cases} x_1^* := \max\{x_1 \mid x \in X\}, \\ x_2^* := \max\{x_2 \mid x \in X, x_1 = x_1^*\}, \\ \quad \vdots \\ x_n^* := \max\{x_n \mid x \in X, x_1 = x_1^*, \ldots, x_{n-1} = x_{n-1}^*\}. \end{cases} \tag{10.1.5}$$

So $x^* = (x_1^*, \ldots, x_n^*)^T$ is in X, if it exists. We say that the **greedy algorithm works** for c if for every nonincreasing ordering of the components of c the maxima in (10.1.5) exist and the vector x^* thus obtained maximizes the linear objective function $c^T x$ over X. The following result is due to EDMONDS (1970).

(10.1.6) Theorem. *Let $X \neq \emptyset$ be a down-monotone polyhedron in $\mathbb{R}^n$. Then the greedy algorithm works for every $c \in \mathbb{Q}^n_+$ if and only if X is an extended polymatroid.* □

An equivalent version of this theorem characterizes polymatroids among down-monotone sets in $\mathbb{R}^n_+$ by the greedy algorithm.

Let $X = EP_f$ be an extended polymatroid associated with the submodular function $f : 2^E \to \mathbb{R}$. Then the maxima in (10.1.5) can easily be computed by the following formulas. Let $c \in \mathbb{Q}^E_+$, and let us order the elements of $E = \{e_1, \ldots, e_n\}$ so that $c(e_1) \geq \ldots \geq c(e_n)$. Then

$$\text{(10.1.7)} \qquad \begin{cases} x^*(e_1) := f(\{e_1\}), \\ x^*(e_2) := f(\{e_1, e_2\}) - f(\{e_1\}), \\ \qquad \vdots \\ x^*(e_n) := f(\{e_1, \ldots, e_{n-1}, e_n\}) - f(\{e_1, \ldots, e_{n-1}\}) \end{cases}$$

defines a vertex x^* of EP_f maximizing $c^T x$ over EP_f. (This can be proved using the submodularity of f.) The maximum value is given by

$$\text{(10.1.8)} \quad \hat{f}(c) := c(e_1) f(\{e_1\}) + \sum_{i=2}^{n} c(e_i)(f(\{e_1, \ldots, e_i\}) - f(\{e_1, \ldots, e_{i-1}\}))$$

$$= \sum_{i=1}^{n-1} (c(e_i) - c(e_{i+1})) f(\{e_1, \ldots, e_i\}) + c(e_n) f(\{e_1, \ldots, e_n\}).$$

Another way to state this formula is the following. First note that every nonzero vector $c \in \mathbb{R}^E_+$ can be written uniquely as

$$\text{(10.1.9)} \qquad c = \lambda_1 \chi^{T_1} + \ldots + \lambda_k \chi^{T_k}$$

such that $\lambda_i > 0$ and $\emptyset \neq T_1 \subset T_2 \subset \ldots \subset T_k \subseteq E$. Then

$$\text{(10.1.10)} \qquad \hat{f}(c) = \lambda_1 f(T_1) + \ldots + \lambda_k f(T_k).$$

This shows in particular that

$$\text{(10.1.11)} \qquad \hat{f}(\chi^S) = f(S) \quad \text{for all } S \subseteq E, S \neq \emptyset.$$

This formula proves that the extended polymatroid EP_f uniquely determines f, except for $f(\emptyset)$.

The situation is somewhat more complicated for polymatroids, since in that case the formulas (10.1.7) only apply if f is monotone (i. e., $S \subseteq T \subseteq E \Rightarrow f(S) \leq f(T)$).

It is easy to monotonize any submodular function f by defining

$$\text{(10.1.12)} \qquad f_{\text{mon}}(S) := \min\{f(T) \mid S \subseteq T \subseteq E\} \quad \text{for all } S \subseteq E.$$

Then f_{mon} is again submodular, and we obviously have

$$P_{f_{\text{mon}}} = P_f.$$

So every polymatroid can be defined by a monotone submodular function. If $P_f \neq \emptyset$ then this monotone submodular function is uniquely determined by P_f, except on $\emptyset$, by the formula

$$f_{\text{mon}}(S) := \max\{x(S) \mid x \in P_f\} \text{ for } \emptyset \neq S \subseteq E.$$

By formula (10.1.10) the function $\hat{f}$ may be viewed as an extension of f to $\mathbb{R}_+^E$. In fact, (10.1.10) defines an extension of any set function $f : 2^E \to \mathbb{R}$ to a function $\hat{f} : \mathbb{R}_+^E \to \mathbb{R}$. If f is submodular then $\hat{f}$ is convex since, by definition (10.1.8), it is the value function of a linear program. It is not difficult to show conversely that if $\hat{f}$ is convex then f is submodular.

It follows from formula (10.1.7) that if f is an integer valued submodular set function then all vertices of EP_f are integral. It is easy to derive from this that all vertices of P_f are integral. EDMONDS (1970) proved the following much deeper result.

(10.1.13) Theorem. *If f and g are integer valued submodular functions on the same ground set then all vertices of $P_f \cap P_g$ as well as all vertices of $EP_f \cap EP_g$ are integral.* □

This theorem contains a large number of integrality results in polyhedral combinatorics. In particular it generalizes the matroid intersection theorem (7.5.16).

*10.2 Algorithms for Polymatroids and Submodular Functions

Now we come to the algorithmic aspects of the structures introduced in Section 10.1. Since all problems we encounter will concern submodular functions, we start with the discussion of how such a function $f : 2^E \to \mathbb{R}$ is given. If f were given by an explicit table of all subsets S of E together with the values $f(S)$, the question of minimizing the function in polynomial time would become trivial. Similarly to the case of matroids – see Section 7.5 – we assume that the function f is given by an oracle returning $f(S)$ for each query $S \subseteq E$. Note that for each of the examples (10.1.2), realizations of the oracles can be designed easily whose running time is polynomial in the "natural" encoding lengths of the structures involved. In the examples involving matroids we assume that the matroids are given by an independence oracle or equivalently by a rank oracle. It follows from our General Assumption (1.2.1) on oracles that with the oracle we know an upper bound β on the encoding lengths of its outputs. Therefore, we can define the encoding length $\langle f \rangle$ of a submodular function f on 2^E as follows:

$$\langle f \rangle := |E| + \beta.$$

Let us start with problems concerning polymatroids.

(10.2.1) Optimization over Extended Polymatroids. *Given a submodular function $f : 2^E \to \mathbb{Q}$ (by an oracle) and a vector $c \in \mathbb{Q}_+^E$, find a vector $x \in EP_f$ maximizing $c^T x$.*

This problem is clearly solvable in oracle-polynomial time by the greedy algorithm (10.1.7). While this problem is trivial, it is somewhat surprising that the following one can only be solved in oracle-polynomial time by the ellipsoid method – as yet.

(10.2.2) Optimization over Polymatroids. *Given a submodular function $f : 2^E \to \mathbb{Q}$ and a vector $c \in \mathbb{Q}^E$, find a vector $x \in P_f$ maximizing $c^T x$.*

We may restrict our attention to the case when $c \geq 0$ since the variables x_e with $c_e < 0$ can be set to zero. By our remark after Theorem (10.1.6) the greedy algorithm (10.1.5) can still be used to find this optimum. However, formulas (10.1.7) only yield the greedy solution if f is monotone. If f is not monotone, then it is not known (without the ellipsoid method) how to perform the first iteration of the greedy algorithm (10.1.5), namely to maximize x_1 over P_f. Even the nonemptiness problem for P_f, i. e., deciding whether or not $f \geq 0$, is not known to be solvable by a combinatorial polynomial time algorithm. Equivalently, there is no fast combinatorial way known to compute the monotonization f_{mon} (10.1.12) from f. On the other hand, as

$$P_f = EP_f \cap \mathbb{R}_+^E \tag{10.2.3}$$

and as we can optimize both over EP_f and over $\mathbb{R}_+^E$ in oracle-polynomial time, we can also optimize over P_f in oracle-polynomial time by Exercise (6.5.18).

Consider now the integral version of problem (10.2.1).

(10.2.4) Integral Optimization over Extended Polymatroids. *Given a submodular function $f : 2^E \to \mathbb{Q}$ and a vector $c \in \mathbb{Q}^E$, find an integral vector $z \in EP_f$ maximizing $c^T z$ over the integral vectors in EP_f.*

If f is integral valued, then Problem (10.2.4) is trivial as each vertex of EP_f is integral. In general, however, (10.2.4) is not solvable in oracle-polynomial time, even if we restrict ourselves to monotone half-integral valued functions f such that $f(S) \leq |S|$ for all $S \subseteq E$. Indeed, this special case is equivalent to the matroid matching (or "matroid parity") problem – see JENSEN and KORTE (1981), LOVÁSZ (1980).

(10.2.5) Separation from Extended Polymatroids. *Given a submodular function $f : 2^E \to \mathbb{Q}$ and a vector $y \in \mathbb{Q}^E$, decide whether $y \in EP_f$, and if y is not, find a set $S \subseteq E$ such that $y(S) > f(S)$.*

By the equivalence of optimization and separation, this problem can be solved in oracle-polynomial time since the optimization problem for extended polymatroids can be solved in oracle-polynomial time. There is no combinatorial polynomial time algorithm for solving problem (10.2.5) known.

By the intersection formula (10.2.3), we can also solve the separation problem for polymatroids P_f in oracle-polynomial time. Again, there is no combinatorial

polynomial time algorithm for this problem known, but for the special case where the submodular function f is the rank function of a matroid, CUNNINGHAM (1984) has designed a combinatorial separation algorithm for P_f.

(10.2.6) Optimization over Intersections of Extended Polymatroids. *Given two submodular functions $f, g : 2^E \to \mathbb{Q}$ and a vector $c \in \mathbb{Q}^E$, find a vertex x of $EP_f \cap EP_g$ maximizing $c^T x$.*

Since we can optimize over EP_f and EP_g in oracle-polynomial time we can also solve (10.2.6) in oracle-polynomial time by Exercise (6.5.18). Edmonds' theorem (10.1.13) implies that if f and g are integral valued then the optimum vertex of $EP_f \cap EP_g$ is an integral vector. Optimization over the intersection of three or more extended polymatroids can also be carried out in oracle-polynomial time. However, in this case the optimum vertex may not be integral even if the submodular functions involved are integer valued. In fact, to find an integral vector that maximizes a linear objective function over the integral vectors in the intersection of three matroid polytopes is $\mathcal{NP}$-hard. Similarly as in the previous chapters it is also of interest to look for optimum dual or optimum integral dual solutions to linear programs over P_f, EP_f, and intersections of such polyhedra. We will treat these questions in the next section in a more general setting. Now we come to what is perhaps the most important algorithmic problem on submodular functions.

(10.2.7) Submodular Function Minimization. *Given a submodular function $f : 2^E \to \mathbb{Q}$, find a set $S \subseteq E$ minimizing f.*

Our preparations allow three ways of attack on this problem. First note that the separation of the zero vector from EP_f is equivalent to deciding whether $f \geq 0$ and finding a set $S \subseteq E$ with $f(S) < 0$, if any such set exists. Since we can decide similarly whether the submodular function f minus a constant is nonnegative, we can find by binary search the minimum value μ of f (using the fact that this value μ has encoding length at most β) and also a minimizing set S.

The second approach uses the extension function $\hat{f}$ defined in (10.1.8) – see also (10.1.10). By replacing f by $f - f(\emptyset)$ we may assume that $f(\emptyset) = 0$. Then the following identity holds:

$$\min\{\hat{f}(x) \mid 0 \leq x \leq 1\} = \min\{f(S) \mid S \subseteq E\}. \tag{10.2.8}$$

Moreover, if x^* minimizes $\hat{f}(x)$, and if we write $x^* = \lambda_1 \chi^{T_1} + \ldots + \lambda_k \chi^{T_k}$ as in (10.1.9), then each of the sets T_i minimizes f. Namely, as $f(S) = \hat{f}(\chi^S)$ we have $\min\{\hat{f}(x) \mid 0 \leq x \leq 1\} \leq \min\{f(S) \mid S \subseteq E\} =: \mu$. On the other hand

$$\hat{f}(x^*) = \lambda_1 f(\chi^{T_1}) + \ldots + \lambda_k f(\chi^{T_k}) \geq (\lambda_1 + \ldots + \lambda_k)\mu \geq \mu,$$

since $\|x^*\|_\infty \leq 1$ and $\mu \leq 0$. So it suffices to minimize $\hat{f}$ over the unit cube. But $\hat{f}$ is polyhedral and computable in oracle-polynomial time, so this can be done in oracle-polynomial time by Theorem (6.5.19).

Note that none of the two algorithms presented above is strongly polynomial: The first uses binary search over the range of f, while the second calls the algorithm of Theorem (6.5.19) which cannot be turned into a strongly polynomial one.

However, we will now describe a third method to minimize a submodular function f which is strongly polynomial. Let $F \subseteq \mathbb{R}^E$ denote the set of vectors in the unit hypercube minimizing $\hat{f}$. By the remark above, F is a nonempty polytope with $\{0,1\}$-vertices. So the vertex-complexity of F is at most $2|E|$. Now we design a separation algorithm for F in the sense of Remark (6.4.11).

Let a vector $y \in \mathbb{Q}^E$ be given. If y has a component that is larger than 1 or smaller than 0, then (strong) separation is trivial. If $0 \leq y \leq 1$, then we order the elements of $E = \{e_1, \ldots, e_n\}$ such that $y(e_1) \geq y(e_2) \geq \ldots \geq y(e_n)$ and compute the vector x^* given by formula (10.1.7). So x^* maximizes $y^T x$ over the extended polymatroid EP_f. Moreover, x^* can be viewed as a subgradient of $\hat{f}$ at y. For, by the definition of $\hat{f}$ and x^* we have $y^T x^* = \hat{f}(y)$, $x^T x^* \leq \hat{f}(x)$ for all x in the unit hypercube, and thus for all these vectors x

$$\hat{f}(x) \geq \hat{f}(y) + (x^*)^T(x - y).$$

In particular, if $x \in F$ then $\hat{f}(y) \geq \hat{f}(x)$ and so

$$(x^*)^T x \leq (x^*)^T y,$$

and if $y \notin F$ then $\hat{f}(y) > \hat{f}(x)$ and so $(x^*)^T x < (x^*)^T y$. This proves that the vector x^* has the properties required in Remark (6.4.11). And thus we can solve the strong nonemptiness problem for F in oracle-polynomial time. By Theorem (6.6.5), this algorithm can be turned into a strongly polynomial one. In fact, a more careful analysis shows that it is already strongly polynomial.

There is no combinatorial algorithm known that minimizes a submodular function in polynomial time; although for the case where f is an integral submodular function, CUNNINGHAM (1985) has given a combinatorial algorithm that is polynomial in $|E| + 2^\beta$ (a so-called pseudo-polynomial algorithm). Cunningham's algorithm, for instance, finds a minimum of the submodular function defined in Example (10.1.2) V. (matroid intersection) in polynomial time.

At first sight the problem of maximizing a submodular function would seem analogous. However, this problem is much more difficult; it contains $\mathcal{NP}$-complete special cases like the max-cut problem, and it is easy to show that it cannot be solved in oracle-polynomial time. For a discussion of some approximation schemes as well as some special cases when it can be solved in polynomial time, we refer to LOVÁSZ (1983b) and NEMHAUSER and WOLSEY (1981).

Packing Bases of a Matroid

We finish this section by studying the problem of packing bases of a matroid, which applies, e. g., to the problems (8.4.26) and (8.4.27) on packing spanning trees in an undirected graph. Let $M = (E, \mathcal{I})$ be a matroid, with rank function r. Consider the polyhedron P in $\mathbb{R}^E$ determined by the following linear inequalities:

$$\begin{aligned} x(B) &\geq 1 \quad \text{for all bases } B \text{ of } M, \\ x_e &\geq 0 \quad \text{for all } e \in E. \end{aligned} \tag{10.2.9}$$

So P is equal to the blocking polyhedron of the convex hull of the incidence vector of bases of M. This convex hull itself is a face of the matroid polytope IND(M) of M defined in Section 7.5.

Hence bl(P) is equal to the dominant of the incidence vectors of bases of M, which is determined by the following linear inequalities:

$$x(S) \geq r(E) - r(E \setminus S) \text{ for all } S \subseteq E.$$

(It is not difficult to derive this from the fact that the extended polymatroid EP_f has integral vertices only, where f is the submodular function defined by $f(S) := r(E \setminus S) - r(E)$.)

Hence P is equal to the dominant of the vectors

$$\frac{1}{r(E) - r(E \setminus S)} \chi^S$$

for $S \subseteq E$ with $r(E \setminus S) < r(E)$. In particular, for $c \in \mathbb{Z}_+^E$, the minimum value of $c^T x$ over P is equal to

$$\min\left\{ \frac{1}{r(E) - r(E \setminus S)} c(S) \mid S \subseteq E, r\ E(\setminus S) < r(E) \right\}. \tag{10.2.10}$$

By linear programming duality this minimum value is equal to

$$\begin{aligned} \max \quad & \sum_{B \text{ basis}} \lambda_B \\ \text{s. t.} \quad & \sum_{\substack{B \text{ basis} \\ B \ni e}} \lambda_B \leq c_e \quad \text{for all } e \in E, \\ & \lambda_B \geq 0 \quad \text{for all bases } B \text{ of } M. \end{aligned} \tag{10.2.11}$$

The maximum value of (10.2.11) can be considered as the maximum size of a fractional packing of bases, subject to c.

It was shown by EDMONDS (1965c) that, if we add to (10.2.11) the condition

$$\lambda_B \text{ integer for all bases } B \text{ of } M, \tag{10.2.12}$$

then the maximum value of (10.2.11) might go down, but by less than 1. That is, the maximum value of the integer linear program (10.2.11), (10.2.12) is equal to the lower integer part of the maximum value of the linear program (10.2.11). Since the LP-maximum is equal to (10.2.10), it is not difficult to see that this statement is equivalent to the following theorem of EDMONDS (1965c).

(10.2.13) Theorem. *The maximum number of pairwise disjoint bases in $M = (E, \mathcal{I})$ is equal to*

$$\min\left\{ \left\lfloor \frac{|S|}{r(E) - r(E \setminus S)} \right\rfloor \mid S \subseteq E, r(E \setminus S) < r(E) \right\}.$$

□

Although this theorem seems to cover the case $c = \mathbb{1}$ only, the case of arbitrary $c \in \mathbb{Z}_+^E$ can be derived by the following "multiplication". Replace each $e \in E$ by c_e "copies" $e_1, \ldots, e_{c_e}$ and, for $\tilde{E} := \bigcup_{e \in E} \{e_1, \ldots, e_{c_e}\}$, define a collection $\tilde{\mathcal{F}}$ of subsets of $\tilde{E}$ by:

$$S \in \tilde{\mathcal{F}} \iff S \subseteq \tilde{E} \text{ and, for all } e \in E,\ |S \cap \{e_1, \ldots, e_{c_e}\}| \leq 1$$
$$\text{and } \{e \in E \mid S \cap \{e_1, \ldots, e_{c_e}\} \neq \emptyset\} \in \mathcal{F}.$$

Then $(\tilde{E}, \tilde{\mathcal{F}})$ is a matroid, and a maximum packing of bases in $(\tilde{E}, \tilde{\mathcal{F}})$ corresponds to an optimum solution of the integer LP (10.2.11) with (10.2.12).

EDMONDS (1965c) also gave an oracle-polynomial time algorithm for finding a maximum packing of bases as required in (10.2.13). As a special case this gives a polynomial time algorithm for the spanning tree packing problem (8.4.26).

We cannot obtain an oracle-polynomial time algorithm for the capacitated case (i. e., the integer LP (10.2.11) with (10.2.12)) from the uncapacitated case by the above "multiplication". However, we can proceed as follows. First solve the linear program (10.2.11), with the method of Theorem (6.5.14). This can be done in oracle-polynomial time, since the separation problem for polyhedra P determined by (10.2.9) is solvable in oracle-polynomial time (by the greedy algorithm). This gives bases $B_1, \ldots, B_n$ and rationals $\lambda_{B_1}, \ldots, \lambda_{B_n} > 0$, so that $n \leq |E|$, forming an optimum solution for the LP (10.2.11) (with $\lambda_B := 0$ if $B \notin \{B_1, \ldots, B_n\}$). Hence, setting

$$\lambda'_{B_i} := \lambda_{B_i} - \lfloor \lambda_{B_i} \rfloor \quad \text{for } i = 1, \ldots, n,$$

$$c'_e := \left\lceil \sum_{\substack{i=1 \\ e \ni B_i}}^{n} \lambda'_{B_i} \right\rceil \quad \text{for all } e \in E,$$

the λ'_{B_i} form an optimum solution for the linear program (10.2.11), with c replaced by c'. Now $\sum_{e \in E} c'_e \leq |E| \cdot n \leq |E|^2$. We can multiply each $e \in E$ by c'_e copies $e_1, \ldots, e_{c'_e}$, as we explained above, to obtain the matroid $(\tilde{E}, \tilde{\mathcal{F}})$. We can find a maximum packing of bases in $(\tilde{E}, \tilde{\mathcal{F}})$ with Edmonds' algorithm. This gives an optimum solution μ_B for the integer LP (10.2.11) with (10.2.12), where c is replaced by c'. Then $\lfloor \lambda_B \rfloor + \mu_B$ form an optimum solution for the integer LP (10.2.11) with (10.2.12), for the original objective function c.

As a special case, we have a polynomial time algorithm for the capacitated spanning tree packing problem (8.4.27).

*10.3 Submodular Functions on Lattice, Intersecting, and Crossing Families

We now extend the results of Sections 10.1 and 10.2 to submodular functions of a more general type. Let E be a finite set. A nonempty collection $\mathcal{C}$ of subsets of E is called a **lattice family** or a **ring family** on E if

(10.3.1) $$S, T \in \mathcal{C} \implies S \cap T \in \mathcal{C} \text{ and } S \cup T \in \mathcal{C}.$$

An important class of lattice families comes from posets. An **ideal** of a partially ordered set $(P, \leq)$ is a subset I of P such that if $x \in I$ and $y \leq x$ then $y \in I$. The ideals of a poset form a lattice family. Conversely, every lattice family can be derived from a poset by duplicating elements and adding dummy elements. More precisely, every lattice family $\mathscr{C}$ on a finite set E can be described by a digraph as follows. Let

$$(10.3.2) \qquad R_{\mathscr{C}} := \bigcap_{T \in \mathscr{C}} T, \quad U_{\mathscr{C}} := \bigcup_{T \in \mathscr{C}} T,$$
$$A_{\mathscr{C}} := \{(i,j) \in E \times E \mid \text{for each } T \in \mathscr{C}, \text{ if } i \in T \text{ then } j \in T\}.$$

So $(E, A_{\mathscr{C}})$ is a digraph.

(10.3.3) Proposition. *The lattice family $\mathscr{C}$ is determined by E, $R_{\mathscr{C}}$, $U_{\mathscr{C}}$, and $A_{\mathscr{C}}$. In fact, $T \in \mathscr{C}$ if and only if*

$$(10.3.4) \qquad R_{\mathscr{C}} \subseteq T \subseteq U_{\mathscr{C}} \text{ and there is no } (i,j) \in A_{\mathscr{C}} \text{ with } i \in T, j \notin T.$$

Proof. Every $T \in \mathscr{C}$ satisfies (10.3.4) by the definitions in (10.3.2). Conversely, let T satisfy (10.3.4). The definition of $A_{\mathscr{C}}$ implies that for each pair $i \in T$, $j \notin T$ there exists a set T_{ij} in $\mathscr{C}$ with $i \in T_{ij}$ and $j \notin T_{ij}$. If $T = \emptyset$ then $R_{\mathscr{C}} = \emptyset$, and hence $T \in \mathscr{C}$ as $R_{\mathscr{C}} = \bigcap_{S \in \mathscr{C}} S$ belongs to $\mathscr{C}$ (since $\mathscr{C}$ is a lattice family). Similarly, if $T = E$, then $T \in \mathscr{C}$. So we may assume that $T \neq \emptyset$ and $T \neq E$. Now for each i in T we know that $T_i := \bigcap_{j \notin T} T_{ij} \in \mathscr{C}$. Note that $i \in T_i \subseteq T$. Therefore, $T = \bigcup_{i \in T} T_i$ belongs to $\mathscr{C}$. □

The structures defined in (10.3.2) yield an encoding of a lattice family $\mathscr{C}$ whose space requirements are polynomial in $|E|$ only. From this encoding it is easy to decide whether a set belongs to $\mathscr{C}$ or not. From now on we assume that every lattice family is given this way. For our purposes we may also assume without loss of generality that $R_{\mathscr{C}} = \emptyset$ and $U_{\mathscr{C}} = E$, so $\mathscr{C}$ is just given by the digraph $(E, A_{\mathscr{C}})$.

A lattice family $\mathscr{C}$ gives rise to a closure operator on $\mathscr{C}$ defined for every $X \subseteq \cup\mathscr{C}$:

$$(10.3.5) \qquad \overline{X} := \cap\{S \in \mathscr{C} \mid X \subseteq S\}.$$

So $\overline{X}$ is the smallest set in $\mathscr{C}$ containing X as a subset. Note that $\overline{X}$ can easily be determined from the digraph $(E, A_{\mathscr{C}})$: $\overline{X}$ is the union of X with the set of all nodes $j \in E$ that can be reached from some node $i \in X$ by an arc.

A function $f : \mathscr{C} \to \mathbb{R}$ is called **submodular** (on the lattice family $\mathscr{C}$) if

$$(10.3.6) \qquad f(S \cap T) + f(S \cup T) \leq f(S) + f(T) \text{ for all } S, T \in \mathscr{C}.$$

Every submodular function defined on a lattice family $\mathscr{C}$ on E can be extended to a submodular function $\bar{f}$ on 2^E by the formula

$$(10.3.7) \qquad \bar{f}(X) := f(\overline{X}) + K|\overline{X} \setminus X| \quad \text{for all } X \subseteq E$$

where K is a sufficiently large constant (e. g., $K := 2^\beta$ suffices, where β is an upper bound on $\langle f(X) \rangle$ for all $X \subseteq E$). It is easy to see that (10.3.7) defines a submodular function on 2^E.

(10.3.8) Minimization of a Submodular Function on a Lattice Family. *Given a lattice family $\mathscr{C}$ on E by the encoding* (10.3.2) *and a submodular function f on $\mathscr{C}$ by an oracle, find a set $T \in \mathscr{C}$ such that $f(T)$ is as small as possible.*

Using the previous discussion, this problem is easily reduced to the problem of minimizing a submodular function defined on all subsets. It suffices to observe that the function $\bar{f}$ defined in (10.3.7) has the same minimizing sets and the same minimum value as f and that the additional computations (calculating $\overline{X}$ and $\bar{f}(X)$) can be carried out in time oracle-polynomial in $\langle f \rangle$ and $|A_{\mathscr{C}}|$. So (10.3.8) can be solved in oracle-polynomial time.

This result can be extended to more general structures. A family $\mathscr{C}$ of subsets of E is called:

(10.3.9) an **intersecting family** if

$$S, T \in \mathscr{C}, S \cap T \neq \emptyset \implies S \cap T \in \mathscr{C}, S \cup T \in \mathscr{C},$$

a **crossing family** if

$$S, T \in \mathscr{C}, S \cap T \neq \emptyset, S \cup T \neq E \implies S \cap T \in \mathscr{C}, S \cup T \in \mathscr{C}.$$

So each lattice family is an intersecting family, and each intersecting family is a crossing family. If we have a crossing family $\mathscr{C}$, then those members of $\mathscr{C}$ missing a given element $e \in E$ form an intersecting family. If we have an intersecting family $\mathscr{C}$ then those members of $\mathscr{C}$ containing a given element $e \in E$ form a lattice family. The complements of the members of a crossing family form another crossing family.

A trivial example of an intersecting family is the family of nonempty subsets of a set E. The family of all nonempty proper subsets of E forms a crossing family. If G is any graph then those node sets $\emptyset \neq W \neq V(G)$ for which $\delta(W)$ is a minimum cardinality cut form a crossing family. If D is any digraph and $r \in V(D)$, then those node sets $\emptyset \neq W \subseteq V(D) \setminus \{r\}$ for which $\delta^-(W)$ is a minimum cardinality r-cut form an intersecting family.

As is the case for lattice families, also crossing families have a compact specification: For any $\mathscr{C} \subseteq 2^E$, and any pair $i, j \in E$ set:

(10.3.10) $$\mathscr{C}_{ij} := \{T \in \mathscr{C} \mid i \in T, j \notin T\}.$$

It is easy to see that $\mathscr{C}$ is a crossing family if and only if $\mathscr{C}_{ij}$ is a lattice family for each pair $i, j \in E$. By specifying each $\mathscr{C}_{ij}$ by $R_{\mathscr{C}_{ij}}$, $U_{\mathscr{C}_{ij}}$, $A_{\mathscr{C}_{ij}}$ (like in (10.3.2)), by Proposition (10.3.3), we can describe $\mathscr{C}$ in space bounded by a polynomial in $|E|$. (Note that we should tell separately whether $\emptyset$ belongs to $\mathscr{C}$ and whether E belongs to $\mathscr{C}$, as they do not belong to any $\mathscr{C}_{ij}$.)

Let $f : \mathscr{C} \to \mathbb{R}$. If $\mathscr{C}$ is an intersecting family, then f is called **submodular on intersecting pairs** if

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T)$$

for all S, T in $\mathscr{C}$ with $S \cap T \neq \emptyset$. If $\mathscr{C}$ is a crossing family, then f is called **submodular on crossing pairs** if this inequality holds for all S, T in $\mathscr{C}$ with $S \cap T \neq \emptyset$, $S \cup T \neq E$.

(10.3.11) Minimization of a Submodular Function on a Crossing Family. *Given a crossing family $\mathscr{C}$ (by $R_{\mathscr{C}_{ij}}$, $U_{\mathscr{C}_{ij}}$, $A_{\mathscr{C}_{ij}}$ for all $i,j \in E$, and by $\mathscr{C} \cap \{\emptyset, E\}$), and a function $f : \mathscr{C} \to \mathbb{Q}$ submodular on crossing pairs (by an oracle), find a set $T \in \mathscr{C}$ such that $f(T)$ is as small as possible.*

The problem of minimizing a submodular function on a lattice family is of course a special case of (10.3.11). We now show that (10.3.11) can be solved in polynomial time. In fact, it is easily reduced to the case of lattice families. For each pair $i,j \in E$ we set $f_{ij} := f \mid \mathscr{C}_{ij}$. For each pair i,j, the function f_{ij} is a submodular function on the lattice family $\mathscr{C}_{ij}$. So we can find a set T_{ij} in $\mathscr{C}_{ij}$ minimizing f_{ij} in oracle-polynomial time. Now, over all pairs i,j, we determine T_{ij} with $f(T_{ij})$ minimal. We compare this value with $f(\emptyset)$, (if $\emptyset \in \mathscr{C}$) and $f(E)$ (if $E \in \mathscr{C}$). This yields the minimum of f.

Just as we have extended the notion of a submodular function on 2^E to more general set systems we will now generalize polymatroids and results about these in a similar way. Let $\mathscr{C} \subseteq 2^E$ be an intersecting family and $f : \mathscr{C} \to \mathbb{Q}$ submodular on intersecting pairs of $\mathscr{C}$. We define the **polymatroid** $P_f(\mathscr{C})$ and the **extended polymatroid** $EP_f(\mathscr{C})$ **corresponding to** f as follows:

$$\begin{aligned} (10.3.12) \qquad P_f(\mathscr{C}) &:= \{x \in \mathbb{R}^E \mid x(T) \leq f(T) \text{ for all } T \in \mathscr{C},\ x \geq 0\}, \\ EP_f(\mathscr{C}) &:= \{x \in \mathbb{R}^E \mid x(T) \leq f(T) \text{ for all } T \in \mathscr{C}\}. \end{aligned}$$

Now there are the following fundamental integrality results, due to EDMONDS (1970) and EDMONDS and GILES (1977), generalizing Theorem (10.1.13). Let $\mathscr{C}_1$ and $\mathscr{C}_2$ be intersecting families, and let $f_1 : \mathscr{C}_1 \to \mathbb{Q}$ and $f_2 : \mathscr{C}_2 \to \mathbb{Q}$ be submodular on intersecting pairs. Let $l, u \in (\mathbb{R} \cup \{\pm\infty\})^E$. Consider the system of linear inequalities for vectors $x \in \mathbb{R}^E$:

$$\begin{aligned} (10.3.13) \qquad &\text{(i)} && l(e) \leq x(e) \leq u(e) && \text{for all } e \in E, \\ &\text{(ii)} && x(T) \leq f_1(T) && \text{for all } T \in \mathscr{C}_1, \\ &\text{(iii)} && x(T) \leq f_2(T) && \text{for all } T \in \mathscr{C}_2. \end{aligned}$$

(10.3.14) Polymatroid Intersection Theorem. *Assume that for a given objective function $c^T x$, $c \in \mathbb{Q}^E$, the maximization problem over* (10.3.13) *has an optimum solution. If l, u, f_1, and f_2 in* (10.3.13) *are integral (allowing infinity for l and u) then this maximization problem has an integral optimum solution. If c is integral then the dual problem has an integral optimum solution.* □

Note that the polyhedra $P_f(\mathscr{C})$ and $EP_f(\mathscr{C})$ – see (10.3.12) – can also be defined for a submodular function $f : \mathscr{C} \to \mathbb{Q}$, where $\mathscr{C}$ is a crossing family. But it turns out that even for the case of a single integral submodular function $f : \mathscr{C} \to \mathbb{Z}$, $\mathscr{C}$ a crossing family, the polyhedra $P_f(\mathscr{C})$ and $EP_f(\mathscr{C})$ may have nonintegral vertices. Consider for instance the crossing family $\mathscr{C}$ consisting of all 2-element subsets of a 3-element set and the function f assigning the value 1 to each of these sets. Then the vector $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T$ is a vertex of both $P_f(\mathscr{C})$ and $EP_f(\mathscr{C})$. This shows in particular that Theorem (10.3.14) does not extend to crossing families.

The first part of Theorem (10.3.14) states that $EP_{f_1}(\mathscr{C}_1) \cap EP_{f_2}(\mathscr{C}_2) \cap B$ has integral vertices, for each "integral box" $B = \{x \mid l \leq x \leq u\}$. The second part states that the system (10.3.13) is totally dual integral for all l, u, in other words (10.3.13) (ii), (iii) is "box-TDI".

As each intersecting family is a crossing family, and as we can minimize submodular functions on crossing families (see the remarks after (10.3.11)), we can solve the separation problem for $EP_{f_1}(\mathscr{C}_1)$ (and a given vector $y \in \mathbb{Q}^E$) in oracle-polynomial time, since it amounts to minimizing the function $\tilde{f}_1 : \mathscr{C}_1 \to \mathbb{Q}$, where $\tilde{f}_1(T) := f_1(T) - y(T)$ for all $T \in \mathscr{C}_1$, which is submodular on intersecting pairs. This observation also applies to $EP_{f_2}(\mathscr{C}_2)$. The strong separation problem for the box B is trivial; we just check the upper and lower bounds. Hence we can solve the strong separation problem for $EP_{f_1}(\mathscr{C}_1) \cap EP_{f_2}(\mathscr{C}_2) \cap B$. Therefore, by Exercise (6.5.18), also the strong optimization problem for $EP_{f_1}(\mathscr{C}_1) \cap EP_{f_2}(\mathscr{C}_2) \cap B$ is polynomially solvable. That is, any linear program over (10.3.13) is solvable in oracle-polynomial time.

If f_1, f_2, l, u are integer-valued and the solution set of (10.3.13) has a vertex, then by finding an optimum vertex solution we will automatically get an integral optimum solution. Since we have allowed infinite components for l and u, it may happen that the solution set of (10.3.13) has no vertices. Then we can find an integer optimum solution as follows. Let x^* be any optimum solution obtained by maximizing $c^T x$ over (10.3.13). Then x^* is also an optimum solution after intersecting the feasible region with the box

$$B' := \{x \mid \lfloor x^*(e) \rfloor \leq x(e) \leq \lceil x^*(e) \rceil \text{ for all } e \in E\}.$$

For the new linear program we can find an optimum solution x^* that is a vertex of $EP_{f_1}(\mathscr{C}_1) \cap EP_{f_2}(\mathscr{C}_2) \cap B'$ (by Exercise (6.5.18)). Clearly, x^* is again an optimum solution of the original linear program. By Theorem (10.3.14), this vertex is integral.

It is less direct to find an integral dual optimum solution, if the objective vector c is integral. Finding this is based not only on the ellipsoid method, but also on a variant of an "uncrossing" technique. With the ellipsoid method we can find in polynomial time an optimum solution for the dual problem:

$$\begin{aligned} (10.3.15) \qquad \min & \sum_{e \in E} (u(e)z(e) - l(e)w(e)) + \sum_{T \in \mathscr{C}_1} f_1(T) y_T^1 + \sum_{T \in \mathscr{C}_2} f_2(T) y_T^2, \\ & z - w + \sum_{T \in \mathscr{C}_1} \chi^T y_T^1 + \sum_{T \in \mathscr{C}_2} \chi^T y_T^2 = c, \\ & z, w, y^1, y^2 \geq 0, \end{aligned}$$

such that at most $|E|$ of the components of y^1 and y^2 are strictly positive.

We will now modify the vector y^1 so that the corresponding collection $\mathscr{F}_1 := \{T \in \mathscr{C}_1 \mid y_T^1 > 0\}$ is **laminar**, i. e., if $S, T \in \mathscr{F}_1$ and $S \cap T \neq \emptyset$ then $T \subseteq S$ or $S \subseteq T$. Define, for the current vector y^1,

$$d := \sum_{T \in \mathscr{F}_1} \chi^T y_T^1,$$

and for each $e \in E$:

$$R_e := \cap\{T \in \mathscr{F}_1 \mid e \in T\}.$$

Now we easily find sets $T_1, \ldots, T_k$ such that $\emptyset \neq T_1 \subset T_2 \subset \ldots \subset T_k$ and

$$d = \lambda_1 \chi^{T_1} + \lambda_2 \chi^{T_2} + \ldots + \lambda_k \chi^{T_k}$$

for certain $\lambda_1, \ldots, \lambda_k > 0$. For each $i = 1, \ldots, k$, let $C_{i1}, \ldots, C_{it_i}$ be the components of the hypergraph

$$\{R_e \mid e \in T_i\}.$$

(Recall that the components of a hypergraph H are the equivalence classes with respect to the following equivalence relation: for two nodes u, v of H we set $u \sim v :\Longleftrightarrow \exists$ edges $E_1, \ldots, E_k$ of H with $E_i \cap E_{i+1} \neq \emptyset$, $i = 1, \ldots, k-1$ and $u \in E_1$, $v \in E_k$.)

Note that for each $e \in T_i$, we have $R_e \subseteq T_i$. Namely, suppose $f \notin T_i$. Then $d_f < d_e$, and hence $e \in T$, $f \notin T$ for some $T \in \mathscr{F}_1$. Therefore, $f \notin R_e$.

Now we claim:

(10.3.16) (i) $\{C_{ij} \mid i = 1, \ldots, k;\ j = 1, \ldots, t_i\}$ is a laminar subcollection of $\mathscr{C}_1$;
(ii) Setting $\tilde{y}^1_T := \lambda_i$ if $T = C_{ij}$ for some $i = 1, \ldots, k;\ j = 1, \ldots, t_i$,
$\tilde{y}^1_T := 0$ for the other $T \in \mathscr{C}_1$
gives another optimum solution.

The fact that the collection in (i) is laminar follows easily from the way of constructing the C_{ij}. Also the fact that the $\tilde{y}^1_T$ give again a feasible solution follows easily, as:

$$\sum_{i=1}^{k} \sum_{j=1}^{t_i} \lambda_i \chi^{C_{ij}} = \sum_{i=1}^{k} \lambda_i \sum_{j=1}^{t_i} \chi^{C_{ij}} = \sum_{i=1}^{k} \lambda_i \chi^{T_i} = d.$$

Next observe that each C_{ij} arises from sets in $\mathscr{F}_1$ by making intersections and unions of intersecting pairs of sets. Hence each C_{ij} belongs to $\mathscr{C}_1$. Moreover, if x^* is an optimum primal solution, then $x^*(C_{ij}) = f_1(C_{ij})$ for each C_{ij}, as $\mathscr{F}_1 \subseteq \{T \in \mathscr{C}_1 \mid x^*(T) = f_1(T)\}$, and as the latter set is closed under taking intersections and unions of intersecting pairs. Therefore, by complementary slackness, the $\tilde{y}^1_T$ give again an optimum solution. So we may assume that $\mathscr{F}_1$ is laminar.

Similarly we can transform y^2_T so that the collection $\mathscr{F}_2 := \{T \in \mathscr{C}_2 \mid y^2_T > 0\}$ is laminar. Thus, the minimum value of (10.3.15) is equal to the minimum value of:

$$\text{(10.3.17)} \quad \min \sum_{e \in E} (u(e)z(e) - l(e)w(e)) + \sum_{T \in \mathscr{F}_1} f_1(T) y^1_T + \sum_{T \in \mathscr{F}_2} f_2(T) y^2_T,$$

$$z - w + \sum_{T \in \mathscr{F}_1} \chi^T y^1_T + \sum_{T \in \mathscr{F}_2} \chi^T y^2_T = c,$$

$$z, w, y^1, y^2 \geq 0.$$

As both $\mathcal{F}_1$ and $\mathcal{F}_2$ are laminar, the constraint matrix in (10.3.17) is totally unimodular. Therefore we can find an integral optimum solution for (10.3.17) in polynomial time – see Section 7.1. Extending this solution appropriately with 0's, we obtain an integral optimum solution for the original minimization problem (10.3.15). This argument also yields a proof of the Edmonds-Giles theorem (10.3.14).

So we have the following conclusion.

(10.3.18) Theorem. *(a) For any integer-valued functions f_1, f_2, l, u (allowing infinity for l and u) and for any rational vector c, the integer linear program* $\max\{c^T x \mid x$ *satisfies* (10.3.13) *and x integral*$\}$ *can be solved in oracle-polynomial time.*
(b) For any rational-valued functions f_1, f_2, l, u (allowing infinity for l and u) and for any integral vector c, the dual of the linear program $\max\{c^T x \mid x$ *satisfies* (10.3.13)$\}$ *can be solved in integers in oracle-polynomial time.* □

We describe two consequences of this result. Let $\mathcal{C} \subseteq 2^E$ be an intersecting family, and let $f : \mathcal{C} \to \mathbb{Q}$ be submodular on intersecting pairs. Define $\tilde{\mathcal{C}}$ and $\tilde{f} : \tilde{\mathcal{C}} \to \mathbb{Q}$ by

(10.3.19) $\tilde{\mathcal{C}} := \{T \subseteq E \mid T$ can be partitioned into sets $T_1, \ldots, T_t \in \mathcal{C}\}$,

$$\tilde{f}(T) := \min\{f(T_1) + \ldots + f(T_t) \mid t \geq 0; T_1, \ldots, T_t \in \mathcal{C} \setminus \{\emptyset\} \text{ form a partition of } T\}.$$

As a degenerate case, this definition gives that $\emptyset \in \tilde{\mathcal{C}}$ and $\tilde{f}(\emptyset) = 0$. It can be shown that $\tilde{\mathcal{C}}$ is a lattice family, and that $\tilde{f}$ is submodular (on all pairs of $\tilde{\mathcal{C}}$). In fact, $\tilde{\mathcal{C}}$ is the smallest lattice family containing $\mathcal{C}$, and $\tilde{f}$ is the unique maximal submodular function g on $\tilde{\mathcal{C}}$ such that $g(\emptyset) = 0$ and $g(T) \leq f(T)$ for all T in $\mathcal{C} \setminus \{\emptyset\}$. (That is, if h is submodular on $\tilde{\mathcal{C}}$, $h(\emptyset) = 0$ and $h(T) \leq f(T)$ for all $T \in \mathcal{C} \setminus \{\emptyset\}$, then $h(T) \leq \tilde{f}(T)$ for all T in $\tilde{\mathcal{C}}$.) Note that

$$R_{\tilde{\mathcal{C}}} = \emptyset, \quad U_{\tilde{\mathcal{C}}} = \bigcup_{i,j \in E} U_{\mathcal{C}_{ij}}, \quad A_{\tilde{\mathcal{C}}} = \{(i,j) \in E \times E \mid \mathcal{C}_{ij} = \emptyset\}.$$

So the compact encoding for the lattice family $\tilde{\mathcal{C}}$ can be derived easily from that for the intersecting family $\mathcal{C}$. Now we have:

(10.3.20) Corollary. *For any intersecting family $\mathcal{C} \subseteq 2^E$ (given by the representation scheme described before), any function $f : \mathcal{C} \to \mathbb{Q}$ submodular on intersecting pairs (given by an oracle), and any given set $T \in \tilde{\mathcal{C}}$, one can determine the value $\tilde{f}(T)$ defined in* (10.3.19) *in oracle-polynomial time.*

Proof. Clearly $\mathcal{C} \setminus \{\emptyset\}$ is an intersecting family, and so by Theorem (10.3.18) we can find in oracle-polynomial time an integral optimum solution for

$$\min \sum_{S \in \mathcal{C} \setminus \{\emptyset\}} f(S) y_S$$

$$\sum_{S \in \mathcal{C} \setminus \{\emptyset\}} \chi^S y_S = \chi^T,$$

$$y_S \geq 0 \quad \text{for all } S \in \mathcal{C} \setminus \{\emptyset\}.$$

It follows that $y_S = 0$ or 1 for all S, and that if $T_1, \ldots, T_k$ are the sets S with $y_S = 1$ then these sets partition T and $\tilde{f}(T) = f(T_1) + \ldots + f(T_k)$ holds. □

A special case of (10.3.19) is the following. Let $\mathscr{C} \subseteq 2^E$ be an intersecting family. Let $f : \mathscr{C} \to \mathbb{Q}$ be submodular on intersecting pairs. Let $g : \mathscr{C} \to \mathbb{Q}$ be defined by

$$g(T) := f(T) - 1 \quad \text{for all } T \in \mathscr{C}.$$

Trivially, g is again submodular on intersecting pairs. The function $\tilde{g}$, derived from g as defined in (10.3.19), is called the **Dilworth truncation** of f. In particular, if $\mathscr{C} = 2^E$, then $\tilde{g}$ is the maximal submodular function $h : 2^E \to \mathbb{Q}$ such that $h(\emptyset) = 0$ and $h(T) \leq f(T) - 1$ for all $T \neq \emptyset$.

(10.3.21) Corollary. *For any intersecting family $\mathscr{C} \subseteq 2^E$ (given by the representation scheme described before), any function $f : \mathscr{C} \to \mathbb{Q}$ submodular on intersecting pairs (given by an oracle), and any given set T in $\mathscr{C}$, one can determine $h(T)$ in oracle-polynomial time, where h denotes the Dilworth truncation of f.* □

The following construction of EDMONDS and GILES (1977) brings together flows (discussed in Sections 7.1 and 8.3) and submodular functions. Let $D = (V, A)$ be a directed graph. Recall that for $W \subseteq V$, $\delta^+(W)$ ($\delta^-(W)$) denotes the set of arcs in A leaving W (entering W). Let $\mathscr{C}$ be a crossing family of subsets of V, and let $f : \mathscr{C} \to \mathbb{Q}$ be submodular on crossing pairs. Let $l, u, c \in (\mathbb{Q} \cup \{\pm\infty\})^A$, and consider the linear program

$$\text{(10.3.22)} \qquad \max \sum_{a \in A} c(a)x(a)$$

$$\text{(i)} \quad \sum_{a \in \delta^+(W)} x(a) - \sum_{a \in \delta^-(W)} x(a) \leq f(W) \quad \text{for all } W \in \mathscr{C},$$

$$\text{(ii)} \quad l(a) \leq x(a) \leq u(a) \quad \text{for all } a \in A.$$

A solution of this linear program is called a **submodular circulation**. If $\mathscr{C} = \{\{v\} \mid v \in V\}$ and $f \equiv 0$, then we get just the circulations of a digraph – see (8.3.27). The following was shown by Edmonds and Giles:

(10.3.23) Theorem. *Suppose that program* (10.3.22) *has an optimum solution. If f, l, and u are integral, then* (10.3.22) *has an integral optimum solution. If c is integral, then the dual of* (10.3.22) *has an integral optimum solution.* □

The first assertion here means that each minimal face of the feasible region of (10.3.22) contains an integral point. So, if this feasible region has a vertex, all vertices are integral. The second assertion says that the inequality system (10.3.22) is box-TDI.

Edmonds and Giles' result contains several other results as special cases. It contains the Polymatroid Intersection Theorem (10.3.14), by taking

$$V := \{h_e \mid e \in E\} \cup \{t_e \mid e \in E\}, \tag{10.3.24}$$
$$A := \{(t_e, h_e) \mid e \in E\},$$

where, for each $e \in E$, the h_e and t_e are new nodes. Moreover, $\mathscr{C}$ is the collection of all subsets W of V such that

$$\begin{array}{lll} \text{(i)} & W = \{t_e \mid e \in T\} & \text{for some } T \text{ in } \mathscr{C}_1, \text{ or} \\ \text{(ii)} & W = V \setminus \{h_e \mid e \in T\} & \text{for some } T \text{ in } \mathscr{C}_2, \end{array} \tag{10.3.25}$$

and $f : \mathscr{C} \to \mathbb{Q}$ is defined by

$$\begin{aligned} f(W) &:= f_1(T) \quad \text{for } W \text{ as in (10.3.25) (i)}, \\ f(W) &:= f_2(T) \quad \text{for } W \text{ as in (10.3.25) (ii)}. \end{aligned} \tag{10.3.26}$$

Then $\mathscr{C}$ is a crossing family and f is submodular on crossing pairs. By identifying every arc $(t_e, h_e) \in A$ with its corresponding element $e \in E$ we see that the inequalities in (10.3.22) are equivalent to those in (10.3.13). Thus, Theorem (10.3.14) is a special case of Theorem (10.3.23). In particular, the matroid intersection theorem (7.5.16) is a special case.

As remarked before, another special case of (10.3.22) is the minimum-cost circulation problem (8.3.27), and hence so is the min-cost flow problem.

A further special case is the Lucchesi-Younger theorem (8.4.32). This is the case where, for a given digraph $D = (V, A)$,

$$\begin{aligned} \mathscr{C} &= \{W \subseteq V \mid \delta_A^-(W) = \emptyset,\ \emptyset \neq W \neq V\}, \\ f(W) &= -1, && \text{for all } W \in \mathscr{C}, \\ c(a) &= +1,\ l(a) = -\infty,\ u(a) = 0 && \text{for all } a \in A. \end{aligned} \tag{10.3.27}$$

For these data, Theorem (10.3.23) is equivalent to the Lucchesi-Younger theorem: the maximum number of pairwise disjoint dicuts in a digraph is equal to the minimum number of arcs intersecting all dicuts (i. e., the minimum cardinality of a dijoin).

(10.3.28) Theorem. *The submodular circulation problem* (10.3.22) *can be solved in oracle-polynomial time. If f, l, u are integral, one can find an integral optimum solution in oracle-polynomial time. If c is integral, one can find an integral optimum dual solution in oracle-polynomial time.*

Proof. Again, we can solve (10.3.22) in oracle-polynomial time, as the separation problem for the feasible region can be solved in oracle-polynomial time: checking (10.3.22) (ii) is easy, while checking (10.3.22) (i) means checking whether, for a given $y \in \mathbb{Q}^A$, the submodular function $g : \mathscr{C} \to \mathbb{Q}$ on the crossing family $\mathscr{C}$ defined by

$$g(W) := f(W) - \sum_{a \in \delta^+(W)} y(a) + \sum_{a \in \delta^-(W)} y(a) \quad \text{for all } W \in \mathscr{C} \tag{10.3.29}$$

is nonnegative. This can be done by the method described after (10.3.11).

Suppose now that f, l, u are integral. Let x^* be a (possibly fractional) optimum solution. Next, find a vertex as optimum solution of problem (10.3.22) with $l(a)$ replaced by $\lfloor x^*(a) \rfloor$ and $u(a)$ replaced by $\lceil x^*(a) \rceil$. By the Edmonds-Giles theorem, this vertex is integral, and one easily checks that it also is an optimum solution of the original problem (10.3.22).

To prove the last assertion of the theorem, we use the familiar uncrossing technique. It will, however, be quite tricky to guarantee the polynomiality of the procedure. Let us start with the following lemma. For any collection $\mathcal{F}$ of subsets of a set V, let $\mathrm{cr}(\mathcal{F})$ denote the smallest crossing family on V containing $\mathcal{F}$.

(10.3.30) Lemma. *There exists a polynomial time algorithm for the following problem: given a finite set V, subsets $U_1, \ldots, U_t$ of V, rationals $\lambda_1, \ldots, \lambda_t > 0$, and a subset T of V such that there exists a rational μ with*

$$T = \{v \in V \mid (\lambda_1 \chi^{U_1} + \ldots + \lambda_t \chi^{U_t})_v \geq \mu\}; \tag{10.3.31}$$

find sets $V_1, \ldots, V_p$ in $\mathrm{cr}(\{U_1, \ldots, U_t\})$ so that

$$\chi^{V_1} + \ldots + \chi^{V_p} = \chi^T + \nu \mathbb{1} \tag{10.3.32}$$

for some integer ν.

Proof. We describe the algorithm. Set $\mathcal{C} := \mathrm{cr}(\{U_1, \ldots, U_t\})$. For each $v \in T$, let

$$R_v := \cap\{U_i \mid v \in U_i\}.$$

From (10.3.31) we know $v \in R_v \subseteq T$. Let $D_1, \ldots, D_s$ be the components of the hypergraph $\{R_v \mid v \in T\}$. So the sets $D_1, \ldots, D_s$ partition T.

For each $j = 1, \ldots, s$ and for each $u \notin D_j$, let

$$E_{ju} := \cup\{U_i \mid D_j \cap U_i \neq \emptyset, u \notin U_i\}.$$

One easily checks that E_{ju} belongs to $\mathcal{C}$, and that $D_j \subseteq E_{ju}$. For $j = 1, \ldots, s$, let

$$C_{j1}, \ldots, C_{jt_j} \tag{10.3.33}$$

be the components of the hypergraph $\{V \setminus E_{ju} \mid u \notin D_j\}$. Then (10.3.33) partitions $V \setminus D_j$, and each $V \setminus C_{jl}$ belongs to $\mathcal{C}$. Moreover,

$$\chi^{V \setminus C_{j1}} + \ldots + \chi^{V \setminus C_{jt_j}} = \chi^{D_j} + (t_j - 1)\mathbb{1}.$$

Hence

$$\sum_{j=1}^{s} (\chi^{V \setminus C_{j1}} + \ldots + \chi^{V \setminus C_{jt_j}}) = \chi^T + \nu \mathbb{1}$$

for some integer ν. So we have found sets and a number as required. □

We continue with the proof of Theorem (10.3.28). With the ellipsoid method we can find in oracle-polynomial time an optimum solution z, w, y for the dual problem of (10.3.22):

$$\begin{aligned}(10.3.34)\quad &\min \sum_{a\in A}(u(a)z(a)-l(a)w(a))+\sum_{W\in\mathscr{C}} f(W)y_W,\\ &z(a)-w(a)+\sum_{\substack{W\in\mathscr{C}\\ a\in\delta^+(W)}} y_W-\sum_{\substack{W\in\mathscr{C}\\ a\in\delta^-(W)}} y_W=c(a) \text{ for all } a\in A,\\ &z,w,y\geq 0,\end{aligned}$$

such that at most $|A|$ of the components of y are strictly positive.

We will now replace the vector y by a vector $\tilde{y}$ so that $z, w, \tilde{y}$ again forms an optimum solution for (10.3.34), and so that the collection $\tilde{\mathscr{F}} := \{W \subseteq V \mid \tilde{y}_W > 0\}$ is **cross-free**, i. e., if $U, W \in \tilde{\mathscr{F}}$ then $W \cap U = \emptyset$ or $W \cup U = V$ or $W \subseteq U$ or $U \subseteq W$.

Let $\mathscr{F} := \{W \mid y_W > 0\}$. We first determine sets $T_1, \ldots, T_m$ and $\lambda_1, \ldots, \lambda_m > 0$ with

$$(10.3.35)\qquad \sum_{W\in\mathscr{F}} y_W\chi^W = \lambda_1\chi^{T_1}+\lambda_2\chi^{T_2}+\ldots+\lambda_m\chi^{T_m}$$

so that $T_1 \supset T_2 \supset \ldots \supset T_m$. By applying Lemma (10.3.30) to $T_i = \{v \in V \mid (\sum_{W\in\mathscr{F}} y_W\chi^W)_v \geq \lambda_1+\ldots+\lambda_i\}$, we can find for each $i = 1, \ldots, m$, sets $V_1^i, \ldots, V_{p_i}^i$ in $\mathrm{cr}(\mathscr{F})$ so that

$$\chi^{V_1^i}+\ldots+\chi^{V_{p_i}^i} = \chi^{T_i}+\nu_i\mathbb{1},$$

for some ν_i. Hence, if we write $V_1^1, \ldots, V_{p_1}^1, V_1^2, \ldots, V_1^m, \ldots, V_{p_m}^m$ as $W_1, \ldots, W_p$ (note that this sequence can contain repeated sets), then

$$(10.3.36)\qquad \chi^{W_1}+\ldots+\chi^{W_p} = \chi^{T_1}+\ldots+\chi^{T_m}+(\nu_1+\ldots+\nu_m)\mathbb{1}.$$

Now if there are i, j so that W_i and W_j **cross** (i. e., $W_i \cap W_j \neq \emptyset$ and $W_i \cup W_j \neq V$ and $W_i \not\subseteq W_j$ and $W_j \not\subseteq W_i$), we can replace W_i and W_j by $W_i \cap W_j$ and $W_i \cup W_j$. Then (10.3.36) is maintained, $\sum_{i=1}^p |W_i|$ does not change, and $\sum_{i=1}^p |W_i|^2$ increases.

Thus, after a polynomial number of applications of this uncrossing technique we find cross-free $\tilde{W}_1, \ldots, \tilde{W}_p \in \mathrm{cr}(\mathscr{F})$ satisfying (10.3.36). Since $\mathrm{cr}(\{\tilde{W}_1, \ldots, \tilde{W}_p\}) = \{\tilde{W}_1, \ldots, \tilde{W}_p\}$, by applying Lemma (10.3.30) to $T_i = \{v \in V \mid (\chi^{\tilde{W}_1}+\ldots+\chi^{\tilde{W}_p})_v \geq i+(\nu_1+\ldots+\nu_m)\}$ we can find, for each $i = 1, \ldots, m$, sets $U_1^i, \ldots, U_{q_i}^i$ in $\{\tilde{W}_1, \ldots, \tilde{W}_p\}$ such that

$$\chi^{U_1^i}+\ldots+\chi^{U_{q_i}^i} = \chi^{T_i}+\kappa_i\mathbb{1}$$

for some κ_i. Hence by (10.3.35),

$$\sum_{i=1}^m \lambda_i\left(\chi^{U_1^i}+\ldots+\chi^{U_{q_i}^i}\right) = \sum_{W\in\mathscr{F}} y_W\chi^W+(\lambda_1\kappa_1+\ldots+\lambda_m\kappa_m)\mathbb{1}.$$

Therefore,

$$\sum_{\substack{W\in\mathcal{F}\\ a\in\delta^+(W)}} y_W - \sum_{\substack{W\in\mathcal{F}\\ a\in\delta^-(W)}} y_W = \sum_{\substack{i,j=1\\ a\in\delta^+(U_j')}}^{m,q_i} \lambda_i - \sum_{\substack{i,j=1\\ a\in\delta^-(U_j')}}^{m,q_i} \lambda_i$$

for each $a \in A$. Hence, by defining

$$\tilde{y}_W := \sum_{\substack{i,j=1\\ U_j'=W}}^{m,q_i} \lambda_i$$

for $W \in \mathcal{C}$, we obtain a feasible solution $z, w, \tilde{y}$ for (10.3.34) so that $\{W \mid \tilde{y}_W > 0\}$ is cross-free (as it is a subcollection of $\{\tilde{W}_1, \ldots, \tilde{W}_p\}$). Moreover, as each $\tilde{W}_i$ belongs to $\mathrm{cr}(\mathcal{F})$, we know that $z, w, \tilde{y}$ again satisfies the complementary slackness conditions (0.1.50):

$$\sum_{a\in\delta^+(U)} x^*(a) - \sum_{a\in\delta^-(U)} x^*(a) = f(U) \tag{10.3.37}$$

for all U with $\tilde{y}_U > 0$ and any primal optimum solution x^* (as (10.3.37) holds for each U in $\mathcal{F}$, and as (10.3.37) is maintained under intersecting and joining crossing pairs of sets).

Defining $\tilde{\mathcal{F}} := \{W \mid \tilde{y}_W > 0\}$, it follows that the minimum value of (10.3.34) is equal to the minimum value of:

$$\begin{aligned} &\min \sum_{a\in A}(u(a)z(a) - l(a)w(a)) + \sum_{W\in\tilde{\mathcal{F}}} f(W)y_W, \\ &z(a) - w(a) + \sum_{\substack{W\in\tilde{\mathcal{F}}\\ a\in\delta^+(W)}} y_W - \sum_{\substack{W\in\tilde{\mathcal{F}}\\ a\in\delta^-(W)}} y_W = c(a) \text{ for all } a \in A \\ &z, w, y \geq 0. \end{aligned} \tag{10.3.38}$$

As $\tilde{\mathcal{F}}$ is cross-free, the constraint matrix in (10.3.38) is totally unimodular. Therefore we can find an integral optimum solution for (10.3.38) in oracle-polynomial time. Extending this solution appropriately with 0's, we obtain an integral optimum solution for the original minimization problem (10.3.34). □

There are several further theorems with a content related to the Edmonds-Giles theorem (e. g., results on "kernel systems" of FRANK (1979), "polymatroid network flows" of HASSIN (1978) and LAWLER and MARTEL (1982), "generalized polymatroids" of FRANK (1984)). These can be treated algorithmically along similar lines as in Theorem (10.3.28) – see SCHRIJVER (1984) for a survey.

*10.4 Odd Submodular Function Minimization and Extensions

We showed in Section 7.3 that the separation problem for the perfect matching polytope is equivalent to

(10.4.1) $$\min\{c(\delta(W)) \mid W \subseteq V, |W| \text{ odd}\},$$

for some undirected graph $G = (V, E)$ with $|V|$ even and some function $c : E \to \mathbb{Q}_+$, that is, to finding a minimum-capacity odd cut. This problem is, as we saw, indeed polynomially solvable. Since the minimum cut problem is a special case of the submodular function minimization problem – cf. (10.1.2) I. – the question arises whether we can generalize this "odd variant" to arbitrary submodular functions. So the question is: can we solve, for any given submodular function $f : 2^E \to \mathbb{Q}$, the problem

(10.4.2) $$\min\{f(S) \mid S \subseteq E, |S| \text{ odd}\}$$

in polynomial time? In this section we shall see that this is indeed the case.

More generally, consider the following. Let E be a finite set, and let $\mathcal{C}$ be a lattice family on E. Let $\mathcal{G}$ be a subcollection of $\mathcal{C}$ satisfying the following condition for all $S, T \subseteq E$:

(10.4.3) If three of the sets S, T, $S \cap T$, $S \cup T$ belong to $\mathcal{C} \setminus \mathcal{G}$, then also the fourth belongs to $\mathcal{C} \setminus \mathcal{G}$.

Note that for any lattice family $\mathcal{C}$, the collection $\mathcal{G} := \{S \in \mathcal{C} \mid |S| \text{ odd}\}$ satisfies (10.4.3). More generally, for any $a, b \in \mathbb{Z}$, the collection $\mathcal{G} := \{S \in \mathcal{C} \mid |S| \not\equiv a \pmod b\}$ satisfies (10.4.3). Another example is obtained if $\mathcal{G}$ is any subcollection of $\mathcal{C}$ such that $\mathcal{C} \setminus \mathcal{G}$ is an antichain (i. e., no two sets in $\mathcal{C} \setminus \mathcal{G}$ are contained in each other).

Let us consider another, similar condition:

(10.4.4) If $S \in \mathcal{G}$ and $T \in \mathcal{C} \setminus \mathcal{G}$ then $S \cap T \in \mathcal{G}$ or $S \cup T \in \mathcal{G}$.

The relation between (10.4.3) and (10.4.4) is the following.

(10.4.5) Lemma. (a) (10.4.3) *implies* (10.4.4).
(b) *If* $\mathcal{C} = 2^E$ *and* $\emptyset, E \notin \mathcal{G}$ *then* (10.4.4) *implies* (10.4.3).

Proof. (a) Suppose that $S \in \mathcal{G}$ and $T \in \mathcal{C} \setminus \mathcal{G}$. Then, since $\mathcal{C}$ is a lattice family, i. e., $\mathcal{C}$ is closed under union and intersection, we have that $S \cup T$, $S \cap T \in \mathcal{C}$. If both $S \cup T$, $S \cap T \in \mathcal{C} \setminus \mathcal{G}$ then exactly three of the sets S, T, $S \cup T$, and $S \cap T$ belong to $\mathcal{C} \setminus \mathcal{G}$, which contradicts (10.4.3). So at least one of $S \cup T$ and $S \cap T$ must belong to $\mathcal{G}$.

(b) Suppose three of the sets S, T, $S \cup T$, and $S \cap T$ belong to $\mathcal{C} \setminus \mathcal{G}$; we show that so does the fourth. There are essentially two cases to consider.

Case 1. $S \cup T$, $S \cap T$, and S belong to $\mathcal{C} \setminus \mathcal{G}$. Then $T \in \mathcal{G}$ would immediately contradict (10.4.4).

Case 2. S, T, and (say) $S \cap T$ belong to $\mathcal{C} \setminus \mathcal{G}$. Consider $T' := T \cup (E \setminus S)$. Since $S \in \mathcal{C} \setminus \mathcal{G}$, $S \cap T' = S \cap T \in \mathcal{C} \setminus \mathcal{G}$ and $S \cup T' = E \in \mathcal{C} \setminus \mathcal{G}$, we have by (10.4.4) that $T' \in \mathcal{C} \setminus \mathcal{G}$. But also $T' \cap (S \cup T) = T \in \mathcal{C} \setminus \mathcal{G}$ and $T' \cup (S \cup T) = E \in \mathcal{C} \setminus \mathcal{G}$, and hence again by (10.4.4), $S \cup T \in \mathcal{C} \setminus \mathcal{G}$. □

Now we will prove that if $\mathcal{G}$ satisfies (10.4.3), then we can minimize in polynomial time any function f, submodular on $\mathcal{C}$, over sets in $\mathcal{G}$. We assume – as before – that the lattice family $\mathcal{C} \subseteq 2^E$ is given by the structure (10.3.2), that f is given by an oracle and that we know an upper bound β on the encoding lengths of the outputs of this oracle. Moreover, $\mathcal{G}$ is also given by a membership oracle: we can give any set S to the oracle, and it answers whether S belongs to $\mathcal{G}$ or not. Let us mention as a side remark that, for any oracle-algorithm for the minimization of a submodular function over set-systems $\mathcal{G}$ satisfying (10.4.4), there are instances forcing the algorithm to call the oracle for $\mathcal{G}$ an exponential number of times – see GRÖTSCHEL, LOVÁSZ and SCHRIJVER (1984c).

(10.4.6) Theorem. *For any $\mathcal{C}$, $\mathcal{G}$, f, β (given as described above), where $\mathcal{G}$ satisfies* (10.4.3), *one can find a set $T \in \mathcal{G}$ with $f(T) = \min\{f(S) \mid S \in \mathcal{G}\}$ in oracle-polynomial time.*

Proof. I. We may assume that $E \in \mathcal{C}$, since those elements of E not contained in the unique largest member of $\mathcal{C}$ (and hence in any member of $\mathcal{C}$) can be deleted. We may even assume that $\mathcal{C} = 2^E$. Namely, let, for each $X \subseteq E$, $\overline{X}$ denote the unique smallest member of $\mathcal{C}$ containing X. Set

$$B := 2^\beta$$

(so $|f(X)| \leq B$ for all $X \subseteq \mathcal{C}$) and define a setfunction g on 2^E by

$$g(X) := f(\overline{X}) + 2B|\overline{X} \setminus X|.$$

Then g is obviously submodular, and $g(X) = f(X)$ holds for every $X \in \mathcal{C}$. Furthermore, define $\mathcal{G}_0 := \mathcal{G} \cup (2^E \setminus \mathcal{C})$. Then, since $2^E \setminus \mathcal{G}_0 = \mathcal{C} \setminus \mathcal{G}$, (10.4.3) is fulfilled with 2^E in place of $\mathcal{C}$ and $\mathcal{G}_0$ in place of $\mathcal{G}$. By the definition of B, $g(X) > f(Y)$ for any $X \subseteq E$ and any $Y \in \mathcal{C}$, unless $X = \overline{X}$, i. e., $X \in \mathcal{C}$. Hence, if $\mathcal{G} \neq \emptyset$

$$\min\{g(X) \mid X \in \mathcal{G}_0\} = \min\{g(X) \mid X \in \mathcal{G}\} = \min\{f(X) \mid X \in \mathcal{G}\}.$$

Thus we shall assume in the sequel that $\mathcal{C} = 2^E$.

II. Set $T(\mathcal{G}) := T := \{x \in E \mid \{x\} \in \mathcal{G}\}$, $S := \{x \in E \mid E \setminus \{x\} \in \mathcal{G}\}$, $t := |T|$. Let us make a few observations about the sets T and S.

Claim 1. If $\emptyset \notin \mathcal{G}$ and $E \notin \mathcal{G}$, then $S = T$.
In fact if, e. g., $\{x\} \in \mathcal{G}$ but $E \setminus \{x\} \notin \mathcal{G}$ then – in view of (10.4.5) (a) – by (10.4.4), one of $\emptyset$ and E must belong to $\mathcal{G}$.

Claim 2. If $\emptyset \notin \mathcal{G}$, then for any $A \subseteq E$, one has $A \in \mathcal{G}$ iff $A \cap T \in \mathcal{G}$.
For, let $A \cap T \in \mathcal{G}$, and choose a maximal set A' such that $A \cap T \subseteq A' \subseteq A$ and $A' \in \mathcal{G}$. If $A \neq A'$ then choose any $u \in A \setminus A'$. Then $A' \in \mathcal{G}$, $\{u\} \notin \mathcal{G}$ and $A' \cap \{u\} = \emptyset \notin \mathcal{G}$. Hence by (10.4.3), $A' \cup \{u\} \in \mathcal{G}$, which contradicts the maximality of A'. So $A' = A$ and thus $A \in \mathcal{G}$. The reverse implication follows by the same argument.

Claim 3. If $\emptyset, E \notin \mathcal{G}$ then $X \in \mathcal{G} \iff E \setminus X \in \mathcal{G}$. This follows immediately from condition (10.4.3).

III. We now describe the algorithm in the case when $\emptyset \notin \mathcal{G}$ and $E \notin \mathcal{G}$. This will be the most difficult case; in the other cases the necessary modifications will simplify the argument.

So let $E, \emptyset \notin \mathcal{G}$. We shall describe an algorithm to minimize f over $\mathcal{G}$ in time $O(|E|^3 p(|E|, \beta))$ where $p(n, \beta)$ is an upper bound on the time needed to minimize a submodular setfunction over all subsets of a set of cardinality at most n, whose values have encoding length at most β.

Note that if $T = \emptyset$ then $\mathcal{G} = \emptyset$ by Claim 2, and so in this case we are done. If $T \neq \emptyset$ then by Claims 2 and 3, $|T| \geq 2$. First we find a set A such that $T \cap A \neq \emptyset$, $T \setminus A \neq \emptyset$ and $f(A)$ is minimal. This can be done by $t(t-1)$ applications of the submodular function minimization algorithm described after (10.2.7), by applying it to find the minimum of f over all sets A with $\{u\} \subseteq A \subseteq E \setminus \{v\}$ for all pairs $u, v \in T$, $u \neq v$. We call A a **splitter**.

If $A \in \mathcal{G}$ then we are done. In fact, any set $A' \in \mathcal{G}$ satisfies $T \cap A' \neq \emptyset$ and $T \setminus A' \neq \emptyset$, by Claims 2 and 3 above, and so $f(A) \leq f(A')$ for each $A' \in \mathcal{G}$, by the choice of A. So we may assume that $A \notin \mathcal{G}$.

Set $A_1 := A \cap T$, $A_2 := T \setminus A$. By Claim 2, $A_1 \notin \mathcal{G}$. By Claim 3, $E \setminus A_1 \notin \mathcal{G}$, and so by Claim 2, $A_2 = (E \setminus A_1) \cap T \notin \mathcal{G}$. Take two new elements a_1 and a_2, and define, for $i = 1, 2$,

$$E_i := (E \setminus A_i) \cup \{a_i\},$$

$$f_i(X) := \begin{cases} f(X) & \text{if } X \subseteq E \setminus A_i, \\ f((X \setminus \{a_i\}) \cup A_i) & \text{if } X \subseteq E_i,\ a_i \in X, \end{cases}$$

and

$$\mathcal{G}_i := \{X \in \mathcal{G} \mid X \subseteq E \setminus A_i\} \cup \{X \subseteq E_i \mid a_i \in X, (X \setminus \{a_i\}) \cup A_i \in \mathcal{G}\}.$$

It is straightforward to check that f_i is a submodular setfunction on the subsets of E_i and that the $\mathcal{G}_i \subseteq 2^{E_i}$ satisfy (10.4.4). Furthermore, $\emptyset, E_i \notin \mathcal{G}_i$.

We claim that

$$\min\{f(X) \mid X \in \mathcal{G}\} = \min\{\min\{f_i(X) \mid X \in \mathcal{G}_i\} \mid i = 1, 2\}.$$

The sign $\leq$ is obvious: if X is a set minimizing the right hand side, and say $X \in \mathcal{G}_1$, then either $X' := X$ or $X' := X \setminus \{a_i\} \cup A_1$ is a set such that $X' \in \mathcal{G}$ and $f(X') = f_1(X)$.

To show the reverse inequality, let $X \in \mathcal{G}$ be a set minimizing the left hand side. If $X \cap A_1 = \emptyset$ then $X \in \mathcal{G}_1$ and $f(X) = f_1(X)$, and so we are done. So we may assume that $X \cap A_1 \neq \emptyset$. Similarly, if $A_2 \subseteq X$ then $X' := (X \setminus A_2) \cup \{a_2\} \in \mathcal{G}_2$ and $f(X) = f_2(X')$, and the assertion follows again. So we may also assume that $A_2 \not\subseteq X$.

Since $X \in \mathcal{G}$ but $A \notin \mathcal{G}$, it follows by (10.4.4) that $X \cup A \in \mathcal{G}$ or $X \cap A \in \mathcal{G}$. We treat the first case; the second is similar. Since $X \cup A \in \mathcal{G}$, we have that $f(X \cup A) \geq f(X)$ by the choice of X. Furthermore, since $(X \cap A) \cap T = X \cap A_1 \neq \emptyset$, and $T \setminus (X \cap A) \supseteq T \setminus A \neq \emptyset$, we have that $f(X \cap A) \geq f(A)$ by the choice of A. Using the submodularity of f, we get $f(X \cup A) + f(X \cap A) \leq f(X) + f(A)$. So equality must hold everywhere, in particular $f(X \cup A) = f(X)$. So $X \cup A$ is also

minimizing f over $\mathcal{G}$. But then $X' := ((X \cup A) \setminus A_1) \cup \{a_1\} \in \mathcal{G}_1$ and $f_1(X') = f(X)$, whence the assertion follows again.

Thus we have found that in order to find the minimum of f over $\mathcal{G}$, it suffices to find the minimum of f_1 over $\mathcal{G}_1$ and the minimum of f_2 over $\mathcal{G}_2$. Going on similarly, we can split each of these subproblems into two, or find the minimizing set right away. This describes the algorithm to find the minimum of f over $\mathcal{G}$.

We still have to show that this algorithm has the claimed running time. The main observation is that $T(\mathcal{G}_i) = A_i$, i. e., the set of singletons in $\mathcal{G}$ is split into two nonempty parts to obtain the sets of singletons in $\mathcal{G}_1$ and $\mathcal{G}_2$. This implies that the number of splitters to compute during the procedure is at most $t - 1 < |E|$. To compute one splitter takes less than $|E|^2$ submodular function minimizations, and so the whole algorithm takes only about $|E|^3 p(|E|, \beta)$ time. This completes the case when $\emptyset, E \notin \mathcal{G}$.

IV. Suppose that $\emptyset \notin \mathcal{G}$ but $E \in \mathcal{G}$. We follow the same argument as in III with slight modifications. First we find a set A such that $T \cap A \neq \emptyset$ and $f(A)$ is minimal. This is easily achieved by t applications of the submodular function minimization algorithm.

If $A \in \mathcal{G}$ then we are done again. So suppose that $A \notin \mathcal{G}$. Set $A_1 := A \cap T$; take a new element a_1 and define

$$E_1 := (E \setminus A_1) \cup \{a_1\},$$

$$f_1(X) := \begin{cases} f(X) & \text{if } X \subseteq E \setminus A_1, \\ f((X \setminus \{a_1\}) \cup A_1) & \text{if } X \subseteq E_1,\ a_1 \in X, \end{cases}$$

and

$$\mathcal{G}_1 := \{X \in \mathcal{G} \mid X \subseteq E \setminus A_1\} \cup \{X \subseteq E_1 \mid a_1 \in X, (X \setminus \{a_1\}) \cup A_1 \in \mathcal{G}\}.$$

Also define $E_2 := A$, $f_2(X) := f(X)$ for $X \subseteq A$ and $\mathcal{G}_2 := 2^A \cap \mathcal{G}$. Then f_i is submodular on the subsets of E_i and the $\mathcal{G}_i \subseteq 2^{E_i}$ satisfy (10.4.3). It also follows just like in part III that

$$\min\{f(X) \mid X \in \mathcal{G}\} = \min\{\min\{f_i(X) \mid X \in \mathcal{G}_i\} \mid i = 1, 2\}.$$

So again it suffices to minimize f_1 over $\mathcal{G}_1$ and f_2 over $\mathcal{G}_2$. The second of these tasks can be solved in polynomial time by III. Since $|T(\mathcal{G}_1)| = |A_1| < |T|$, we are finished by induction.

V. The cases when $\emptyset \in \mathcal{G}$ but $E \notin \mathcal{G}$ and when $\emptyset, E \in \mathcal{G}$ can be treated similarly. □

Let us rephrase Theorem (10.4.6) for two special cases of set systems $\mathcal{C}$ and $\mathcal{G}$ satisfying (10.4.3) mentioned above.

(10.4.7) Corollary. *For any lattice family $\mathcal{C}$ (given by the encoding* (10.3.2)*), any $a, b \in \mathbb{Z}$ and any submodular function f on $\mathcal{C}$ (given by an oracle), an optimum solution of* $\min\{f(X) \mid X \in \mathcal{C}, |X| \not\equiv a \pmod b\}$ *can be found in oracle-polynomial time.* □

Corollary (10.4.7) in particular implies that the minimum odd cut problem (8.5.7) can be solved in polynomial time, and through this, that the matching problem (8.5.5) is solvable in polynomial time.

(10.4.8) Corollary. *For any lattice family $\mathscr{C}$ (given by the encoding* (10.3.2)*), any antichain $\mathscr{A}$ in $\mathscr{C}$ (given by an oracle), and any submodular function f on $\mathscr{C}$ (given by an oracle), an optimum solution of* $\min\{f(X) \mid X \in \mathscr{C} \setminus \mathscr{A}\}$ *can be found in oracle-polynomial time.* □

References

The page numbers in brackets at the end of a citation refer to the text.

L. Adleman (1983), "On breaking the iterated Merkle-Hellman public key cryptosystem", in: *Advances in Cryptology*, Proceedings of CRYPTO 82, Plenum Press, New York, 1983, 303–308. [221]

A. V. Aho, J. E. Hopcroft and J. D. Ullman (1974), *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Massachusetts, 1974. [21]

L. Babai (1986), "On Lovász' lattice reduction and the nearest lattice point problem", *Combinatorica* 6 (1986) 1–13. [149, 195]

E. Balas, V. Chvátal and J. Nešetřil (1985), "On the maximum-weight clique problem", MSRR No. 518, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1985. [302]

E. Balas and M. W. Padberg (1975), "Set partitioning", in: B. Roy (ed.), *Combinatorial Programming: Methods and Applications*, Reidel, Dordrecht, 1975, 205–258. [301]

E. Balas and C. S. Yu (1984), "Finding a maximum clique in an arbitrary graph", MSRR No. 515, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984. [302]

M. L. Balinski and R. E. Gomory (1964), "A primal method for the assignment and transportation problem", *Management Science* 10 (1964) 578–593. [233]

F. Barahona (1983), "The max cut problem in graphs not contractible to K_5", *Operations Research Letters* 2 (1983) 107–111. [250]

F. Barahona, M. Grötschel, M. Jünger and G. Reinelt (1986), "An application of combinatorial optimization to statistical physics and circuit layout design", Preprint No. 95, Institut für Mathematik, Universität Augsburg, Augsburg, 1986. [250]

F. Barahona, M. Grötschel and A. R. Mahjoub (1985), "Facets of the bipartite subgraph polytope", *Mathematics of Operations Research* 10 (1985) 340–358. [249]

F. Barahona and A. R. Mahjoub (1985), "Composition in the acyclic subgraph polytope", Report No. 85371-OR, Institut für Ökonometrie und Operations Research, Universität Bonn, Bonn, 1985. [254]

F. Barahona and A. R. Mahjoub (1986), "On the cut polytope", *Mathematical Programming* 36 (1986) 157–173. [249, 250]

I. Bárány and Z. Füredi (1986), "Computing the volume is difficult", in: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (Berkeley, California, 1986), ACM, New York, 1986, 442–447. [127]

C. Berge (1958), "Sur le couplage maximum d'un graphe", *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris)* 247 (1958) 258–259. [255]

C. Berge (1960), "Les problèmes de coloration en théorie des graphes", *Publications de l'Institut de Statistique de l'Université de Paris* 9 (1960) 123–160. [280]

C. Berge (1961), "Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung)", *Wissenschaftliche Zeitschrift*, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe (1961) 114–115. [277]

C. Berge (1962), "Sur une conjecture relative au problème des codes optimaux", Communication, 13ème assemblée générale de l'URSI, Tokyo, 1962. [277, 279]

C. Berge (1973), *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973. [16]

C. Berge and V. Chvátal (eds.) (1984), *Topics on Perfect Graphs*, Annals of Discrete Mathematics 21, North-Holland, Amsterdam, 1984. [283]

C. Berge and P. Duchet (1984), "Strongly perfect graphs", *Annals of Discrete Mathematics* 21 (1984) 57–61. [282]

C. Berge and M. Las Vergnas (1970), "Sur un théorème du type König pour hypergraphes", (in: A. Gewirtz and L. V. Quintas (eds.), *International Conference on Combinatorial Mathematics*, New York, 1970), Annals of the New York Academy of Sciences 175 [Article 1] (1970) 32–40. [282]

S. J. Berkowitz (1984), "On computing the determinant in small parallel time using a small number of processors", *Information Processing Letters* 18 (1984) 147–150. [40]

U. Betke and P. Gritzmann (1986), "Projection algorithms for linear programming", Mathematischer Forschungsbericht 176/1986, Universität Gesamthochschule Siegen, Siegen, 1986. [66]

R. E. Bixby (1982), "Matroids and operations research", in: H. Greenberg, F. Murphy and S. Shaw (eds.), *Advanced Techniques in the Practice of Operations Research*, North-Holland, New York, 1982, 333–458. [211]

R. Bland (1977), "New finite pivoting rules for the simplex method,", *Mathematics of Operations Research* 2 (1977), 103–107. [42]

R. G. Bland, D. Goldfarb and M. J. Todd (1981), "The ellipsoid method: a survey", *Operations Research* 29 (1981) 1039–1091. [65, 71, 72, 82, 94, 107, 248]

F. Bock (1971), "An algorithm to construct a minimum directed spanning tree in a directed network", in: B. Avi-Itzhak (ed.), *Developments in Operations Research*, Vol. 1, Gordon and Breach, New York, 1971, 29–44. [242]

B. Bollobás (1978), *Extremal Graph Theory*, Academic Press, London, 1978. [16]

J. A. Bondy and U. S. R. Murty (1976), *Graph Theory with Applications*, American Elsevier, New York, and Macmillan, London, 1976. [16]

T. Bonnesen and W. Fenchel (1934), *Theorie der konvexen Körper*, Springer, Berlin, 1934 (reprinted: Chelsea, New York, 1948). [49]

K. S. Booth and G. S. Luecker (1976), "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms", *Journal of Computing System Sciences* 13 (1976) 335–379. [280]

K.-H. Borgwardt (1982), "Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex method", *Mathematics of Operations Research* 7 (1982) 441–462. [42, 64]

O. Borůvka (1926), "O jistém problému minimálním", ("On a minimal problem"), *Práce Moravske Přírodovědecke Společnosti v Brně (Acta Societ. Scient. Natur. Moravicae)* 3 (1926) 37–58. [247]

M. Boulala and J. P. Uhry (1979), "Polytope des indépendants dans un graphe série-parallèle", *Discrete Mathematics* 27 (1979) 225–243. [274]

E. F. Brickell (1985), "Breaking iterated knapsacks", in: *Advances in Cryptology*, Proceedings of CRYPTO 84, Springer, Berlin, 1985, 342–358. [221]

M. Burlet and J. Fonlupt (1984), "Polynomial algorithm to recognize a Meyniel graph", in: W. R. Pulleyblank (ed.), *Progress in Combinatorial Optimization*, Academic Press, Toronto, 1984, 69–99. [281]

M. Burlet and J. P. Uhry (1982), "Parity graphs", *Annals of Discrete Mathematics* 16 (1982) 1–26. [281]

J. W. S. Cassels (1959), *An Introduction to the Geometry of Numbers*, Springer, Berlin, 1959. [139]

D. Cheriton and R. E. Tarjan (1976), "Finding minimum spanning trees", *SIAM Journal on Computing* 5 (1976) 724–742. [247]

T.-Y. Cheung (1980), "Computational comparison of eight methods for the maximum network flow problem", *ACM Transactions on Mathematical Software* 6 (1980) 1–16. [236]

Y.-J. Chu and T.-H. Liu (1965), "On the shortest arborescence of a directed graph", *Scientia Sinica* 4 (1965) 1396–1400. [201, 242]

V. Chvátal (1973), "Edmonds polytopes and a hierarchy of combinatorial problems", *Discrete Mathematics* 4 (1973) 305–337. [13]

V. Chvátal (1975), "On certain polytopes associated with graphs", *Journal of Combinatorial Theory B* 18 (1975) 138–154. [273, 274]

V. Chvátal (1983), *Linear Programming*, Freeman, New York, 1983. [1]

V. Chvátal (1984), "Perfectly orderable graphs", *Annals of Discrete Mathematics* 21 (1984) 63–65. [281]

V. Chvátal, C. T. Hoang, N. V. R. Mahadev and D. de Werra (1985), "Four classes of perfectly orderable graphs", Tech. Report ORWP 85/3, École Polytechnique Fédérale de Lausanne, Lausanne, 1985. [281]

S. A. Cook (1971), "The complexity of theorem-proving procedures", in: Proceedings of the Third Annual ACM Symposium on Theory of Computing (Shaker Heights, Ohio, 1971), ACM, New York, 1971, 151–158. [28]

W. Cook and W. R. Pulleyblank (1987), "Linear systems for constrained matching problems", *Mathematics of Operations Research*, to appear. [259]

G. Cornuéjols and W. R. Pulleyblank (1980), "A matching problem with side constraints", *Discrete Mathematics* 29 (1980) 135–159. [265]

H. Crowder and M. W. Padberg (1980), "Solving large-scale symmetric travelling salesman problems to optimality", *Management Science* 26 (1980) 495–509. [46, 265]

W. H. Cunningham (1984), "Testing membership in matroid polyhedra", *Journal of Combinatorial Theory (B)* 36 (1984) 161–188. [46, 214, 310]

W. H. Cunningham (1985), "On submodular function minimization", *Combinatorica* 5 (1985) 185–192. [311]

W. H. Cunningham and A. B. Marsh III. (1978), "A primal algorithm for optimum matching", *Mathematical Programming Study* 8 (1978) 50–72. [256]

G. B. Dantzig (1951), "Maximization of a linear function of variables subject to linear inequalities", Chapter 21 in: T. C. Koopmans (ed.), *Activity Analysis of Production and Allocation* (Cowles Commission Monograph No. 13), Wiley, New York, 1951, 339–347. [41]

G. B. Dantzig (1963), *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963. [1]

L. Danzer, B. Grünbaum and V. Klee (1963), "Helly's theorem and its relatives", in: V. L. Klee (ed.), *Convexity*, American Mathematical Society, Providence, Rhode Island, 1963, 101–180. [69]

E. W. Dijkstra (1959), "A note on two problems in connexion with graphs", *Numerische Mathematik* 1 (1959) 269–271. [235, 247]

R. P. Dilworth (1944), "Dependence relations in a semimodular lattice", *Duke Mathematical Journal* 11 (1944) 575–587. [304]

R. P. Dilworth (1950), "A decomposition theorem for partially ordered sets", *Annals of Mathematics (2)* 51 (1950) 161–166. [240]

E. A. Dinits (1970), "Algorithm for solution of a problem of maximum flow in a network with power estimation" (in Russian), *Doklady Akademii Nauk SSSR* 194 (1970) 754–757 (English translation: *Soviet Mathematics Doklady* 11 (1970) 1277–1280). [198, 236]

G. A. Dirac (1952), "A property of 4-chromatic graphs and some remarks on critical graphs", *Journal of the London Mathematical Society* 27 (1952) 85–92. [274]

G. A. Dirac (1961), "On rigid circuit graphs", *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 25 (1961) 71–76. [280]

G. L. Dirichlet (1842), "Verallgemeinerung eines Satzes aus der Lehre von den Kettenbrüchen nebst einigen Anwendungen auf die Theorie der Zahlen", Bericht über die zur Bekanntmachung geeigneten Verhandlungen der Königlich Preussischen Akademie der Wissenschaften zu Berlin (1842) 93–95 (reprinted in: L. Kronecker (ed.), *G. L. Dirichlet's Werke* Vol. I, G. Reimer, Berlin, 1889 (reprinted: Chelsea, New York, 1969), 635-638). [134, 138, 139]

P. D. Domich, R. Kannan and L. E. Trotter (1987), "Hermite normal form computation using modulo determinant arithmetic", *Mathematics of Operations Research* 12 (1987) 50–59. [45]

R. J. Duffin (1965), "Topology of series-parallel networks", *Journal of Mathematical Analysis and Applications* 10 (1965) 303–318. [274]

J. G. Ecker and M. Kupferschmidt (1983), "An ellipsoid algorithm for nonlinear programming", *Mathematical Programming* 27 (1983) 83–106. [64]

J. Edmonds (1965a), "Paths, trees, and flowers", *Canadian Journal of Mathematics* 17 (1965) 449–467. [255]

J. Edmonds (1965b), "Maximum matching and a polyhedron with 0,1-vertices", *Journal of Research of the National Bureau of Standards B* 69 (1965) 125–130. [204, 205, 255]

J. Edmonds (1965c), "Minimum partition of a matroid into independent subsets", *Journal of Research of the National Bureau of Standards B* 69 (1965) 67–72. [312, 313]

J. Edmonds (1967a), "Optimum branchings", *Journal of Research of the National Bureau of Standards B* 71 (1967) 233–240. [201, 202, 242, 245]

J. Edmonds (1967b), "Systems of distinct representatives and linear algebra", *Journal of Research of the National Bureau of Standards B* 71 (1967) 241–245. [37]

J. Edmonds (1968), "Matroid partition", in: G. B. Dantzig and A. F. Veinott (eds.), *Mathematics of the Decision Sciences* (Lectures in Applied Mathematics, Vol. 11), American Mathematical Society, Providence, Rhode Island, 1968, 335–345. [250]

J. Edmonds (1970), "Submodular functions, matroids, and certain polyhedra", in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, 69–87. [46, 213, 214, 248, 304, 306, 308, 316]

J. Edmonds (1971), "Matroids and the greedy algorithm", *Mathematical Programming* 1 (1971) 127–136. [46, 213]

J. Edmonds (1973), "Edge-disjoint branchings", in: R. Rustin (ed.), *Combinatorial Algorithms*, Academic Press, New York, 1973, 91–96. [246]

J. Edmonds (1979), "Matroid intersection", *Annals of Discrete Mathematics* 4 (1979) 39–49. [216]

J. Edmonds and R. Giles (1977), "A min-max relation for submodular functions on graphs", *Annals of Discrete Mathematics* 1 (1977) 185–204. [16, 244, 316, 320]

J. Edmonds and E. L. Johnson (1970), "Matching, a well-solved class of integer linear programs", in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, 89–92. [257, 258]

J. Edmonds and E. L. Johnson (1973), "Matching, Euler tours and the Chinese postman", *Mathematical Programming* 5 (1973) 88–124. [260, 262]

J. Edmonds and R. M. Karp (1972), "Theoretical improvements in algorithmic efficiency for network flow problems", *Journal of the Association for Computing Machinery* 19 (1972) 248–264. [189, 238]

J. Edmonds, L. Lovász and W. R. Pulleyblank (1982), "Brick decompositions and the matching rank of graphs", *Combinatorica* 2 (1982) 247–274. [160, 181]

E. Egerváry (1931), "Matrixok kombinatorius tulajdonságairól" (Hungarian), ("On combinatorial properties of matrices"), *Matematikai és Fizikai Lapok* 38 (1931) 16–28. [200, 231, 232]

G. Elekes (1986), "A geometric inequality and the complexity of computing volume", *Discrete and Computational Geometry* 1 (1986) 289–292. [127]

P. Elias, A. Feinstein and C. E. Shannon (1956), "A note on the maximum flow through a network", *IRE Transactions on Information Theory IT* 2 (1956) 117–119. [199]

P. van Emde Boas (1981), "Another $\mathcal{NP}$-complete partition problem and the complexity of computing short vectors in a lattice", Report 81-04, Mathematical Institute, University of Amsterdam, Amsterdam, 1981. [141]

S. Even and O. Kariv (1975), "An $O(n^{2.5})$ algorithm for maximum matching in general graphs", in: 16th Annual Symposium on Foundations of Computer Science (Berkeley, California, 1975), IEEE, New York, 1975, 100–112. [255]

S. Even, A. Pnueli and A. Lempel (1972), "Permutation graphs and transitive graphs", *Journal of the Association for Computing Machinery* 19 (1972) 400–410. [280]

D. K. Faddeev and V. N. Faddeeva (1963), *Computational Methods of Linear Algebra*, Freeman, San Francisco, 1963. [1]

P. Feofiloff and D. H. Younger (1985), "Directed cut transversal packing for source-sink connected graphs", preprint, 1985. [252]

R. W. Floyd (1962), "Algorithm 97, Shortest Path", *Communications of the ACM* 5 (1962) 345. [235]

J. Fonlupt, A. R. Mahjoub and J.-P. Uhry (1984), "Composition of graphs and the bipartite subgraph polytope", Research Report No. 459, Laboratoire ARTEMIS (IMAG), Université de Grenoble, Grenoble, 1984. [250]

J. Fonlupt and J. P. Uhry (1982), "Transformations which preserve perfectness and h-perfectness of graphs", *Annals of Discrete Mathematics* 16 (1982) 83–95. [274]

L. R. Ford, Jr. and D. R. Fulkerson (1956), "Maximal flow through a network", *Canadian Journal of Mathematics* 8 (1956) 399–404. [199, 279, 280]

L. R. Ford, Jr. and D. R. Fulkerson (1957), "A simple algorithm for finding maximal network flows and an application to the Hitchcock problem", *Canadian Journal of Mathematics* 9 (1957) 210–218. [197, 231]

L. R. Ford, Jr. and D. R. Fulkerson (1962), *Flows in Networks*, Princeton University Press, Princeton, N. J., 1962. [234, 239, 280]

A. Frank (1976), "Some polynomial algorithms for certain graphs and hypergraphs", in: C. St. J. A. Nash-Williams and J. Sheehan (eds.), Proceedings of the 5th British Combinatorial Conference, 1975, Congressus Numerantium No. XV, Utilitas Mathematica, Winnipeg, 1976, 211–226. [280]

A. Frank (1979), "Kernel systems of directed graphs", *Acta Scientiarum Mathematicarum (Szeged)* 41 (1979) 63–76. [324]

A. Frank (1981a), "A weighted matroid intersection algorithm", *Journal of Algorithms* 2 (1981) 328–336. [216]

A. Frank (1981b), "How to make a digraph strongly connected", *Combinatorica* 1 (1981) 145–153. [252]

A. Frank (1984), "Generalized polymatroids", in: A. Hajnal et al. (eds.), *Finite and Infinite Sets* (Colloquia Mathematica Societatis János Bolyai 37), North-Holland, Amsterdam, 1984, 285–294. [324]

A. Frank and É. Tardos (1987), "An application of simultaneous approximation in combinatorial optimization", *Combinatorica* 7 (1987), to appear. [168, 189]

F. G. Frobenius (1912), "Über Matrizen aus nicht negativen Elementen", Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften zu Berlin (1912) 456–477 (reprinted in: J.-P. Serre (ed.), *Ferdinand Georg Frobenius Gesammelte Abhandlungen* Band III, Springer, Berlin, 1968, 546–567). [230]

F. G. Frobenius (1917), "Über zerlegbare Determinanten", Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften zu Berlin (1917) 274–277 (reprinted in: J.-P. Serre (ed.), *Ferdinand Georg Frobenius Gesammelte Abhandlungen* Band III, Springer, Berlin, 1968, 701–704). [203, 230]

M. A. Frumkin (1976a), "An application of modular arithmetic to the construction of algorithms for solving systems of linear equations" (in Russian), *Doklady Akademii Nauk SSSR* 229 (1976) 1067–1070 (English translation: *Soviet Mathematics Doklady* 17 (1976) 1165–1168). [45, 155]

M. A. Frumkin (1976b), "Algorithms for the solution in integers of systems of linear equations" (in Russian), in: A. A. Fridman (ed.), *Issledovaniya po diskretnoi optimizatsii* (Studies in Discrete Optimization), Izdat. "Nauka", Moscow, 1976, 97–127. [45, 155]

M. A. Frumkin (1976c), "An algorithm for the reduction of a matrix of integers to triangular form with power complexity of the computations" (in Russian), *Èkonomika i Matematicheskie Metody* 12 (1976) 173–178. [45, 155]

D. R. Fulkerson (1961), "An out-of-kilter method for minimal cost flow problems", *Journal of the Society for Industrial and Applied Mathematics* 9 (1961) 18–27. [238]

D. R. Fulkerson (1968), "Networks, frames, blocking systems", in: G. B. Dantzig and A. F. Veinott, Jr. (eds.), *Mathematics of the Decision Sciences, Part 1* (Lectures in Applied Mathematics Vol. 11), American Mathematical Society, Providence, Rhode Island, 1968, 303–334. [225, 234]

D. R. Fulkerson (1970), "The perfect graph conjecture and pluperfect graph theorem", in: R. C. Bose, et al. (eds.), *Proceedings of the Second Chapel Hill Conference on Combinatorial Mathematics and Its Applications*, University of North Carolina, Chapel Hill, North Carolina, 1970, 171–175. [277]

D. R. Fulkerson (1971), "Blocking and anti-blocking pairs of polyhedra", *Mathematical Programming* 1 (1971) 168–194. [225, 284]

D. R. Fulkerson (1972), "Anti-blocking polyhedra", *Journal of Combinatorial Theory B* 12 (1972) 50–71. [284]

D. R. Fulkerson (1974), "Packing rooted directed cuts in a weighted directed graph", *Mathematical Programming* 6 (1974) 1–13. [243]

D. R. Fulkerson and O. A. Gross (1965), "Incidence matrices and interval graphs", *Pacific Journal of Mathematics* 15 (1965) 835–855. [280]

H. N. Gabow, Z. Galil, T. Spencer and R. E. Tarjan (1986), "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs", *Combinatorica* 6 (1986) 109–122. [243, 247]

Z. Galil (1981), "On the theoretical efficiency of various network flow algorithms", *Theoretical Computer Science* 14 (1981) 103–112. [236]

Z. Galil (1983), "Efficient algorithms for finding maximal matchings in graphs", preprint, 1983. [231]

T. Gallai (1950), "On factorization of graphs", *Acta Mathematica Hungarica* 1 (1950) 133–153. [255]

T. Gallai (1959), "Über extreme Punkt- und Kantenmengen", *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Mathematica* 2 (1959) 133–138. [230]

T. Gallai (1962), "Graphen mit triangulierbaren ungeraden Vielecken", *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* 7 (1962) 3–36. [281]

T. Gallai (1963), "Neuer Beweis eines Tutte'schen Satzes", *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* 8 (1963) 135–139. [255]

T. Gallai (1964), "Maximale Systeme unabhängiger Kanten", *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* 9 (1964) 401–413. [255]

T. Gallai (1967), "Transitiv orientierbare Graphen", *Acta Mathematicae Academiae Scientiarum Hungaricae* 18 (1967) 25–66. [280]

F. R. Gantmacher (1959), *The Theory of Matrices*, Volumes I, II (translated from the Russian), Chelsea, New York, 1959. [1]

M. R. Garey and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, Freeman, San Francisco, 1979. [21, 28, 219, 272]

S. I. Gass (1975), *Linear Programming Methods and Applications,* fourth edition, McGraw-Hill, New York, 1975. [1]

J. von zur Gathen and M. Sieveking (1976), "Weitere zum Erfüllungsproblem polynomial äquivalente kombinatorische Aufgaben", in: E. Specker and V. Strassen (eds.), *Komplexität von Entscheidungsproblemen,* Lecture Notes in Computer Science 43, Springer, Heidelberg, 1976, 49–71. [45, 155]

F. Gavril (1972), "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph", *SIAM Journal on Computing* 1 (1972) 180–187. [280]

A. M. H. Gerards, L. Lovász, P. Seymour, A. Schrijver and K. Truemper (1987), "Regular matroids from graphs", to appear. [275]

A. M. H. Gerards and A. Schrijver (1986), "Matrices with the Edmonds-Johnson property", *Combinatorica* 6 (1986) 403–417. [274]

G. de Ghellinck and J.-P. Vial (1986), "A polynomial Newton method for linear programming", *Algorithmica* 1 (1986) 425–453. [66]

A. Ghouila-Houri (1962), "Caractérisation des graphes non orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre", *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences. Paris* 254 (1962) 1370–1371. [280]

R. Giles and L. E. Trotter, Jr. (1981), "On stable set polyhedra for $K_{1,3}$-free graphs", *Journal of Combinatorial Theory B* 31 (1981) 313–326. [302]

P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin and M. H. Wright (1986), "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method", *Mathematical Programming* 36 (1986) 183–209. [66]

P. C. Gilmore and A. J. Hoffman (1964), "A characterization of comparability graphs and of interval graphs", *Canadian Journal of Mathematics* 16 (1964) 539–548. [280]

F. Glover, D. Klingman, J. Mote and D. Whitman (1979), "Comprehensive computer evaluation and enhancement of maximum flow algorithms", Research Report 356, Center of Cybernetic Studies, The University of Texas, Austin, Texas, 1979. [236]

J.-L. Goffin (1984), "Variable metric relaxation methods, Part II: The ellipsoid method", *Mathematical Programming* 30 (1984) 147–162. [65, 122]

A. V. Goldberg and R. E. Tarjan (1986), "A new approach to the maximum flow problem", Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (Berkeley, California, 1986), ACM, New York, 1986, 136–146. [237]

M. Golumbic (1980), *Algorithmic Graph Theory and Perfect Graphs,* Academic Press, New York, 1980. [283]

R. E. Gomory (1958), "Outline of an algorithm for integer solutions to linear programs", *Bulletin of the American Mathematical Society* 64 (1958) 275–278. [13]

R. E. Gomory (1960), "Solving linear programming problems in integers", in: R. Bellman and M. Hall, Jr. (eds.), *Combinatorial Analysis,* Proceedings of Symposia in Applied Mathematics, Volume X, American Mathematical Society, Providence, Rhode Island, 1960, 211–216. [13]

R. E. Gomory and T. C. Hu (1961), "Multi-terminal network flows", *Journal of the Society for Industrial and Applied Mathematics* 9 (1961) 551–570. [248]

R. E. Gomory and T. C. Hu (1964), "Synthesis of a communication network", *Journal of the Society for Industrial and Applied Mathematics* 12 (1964) 348–369. [248]

R. L. Graham and P. Hell (1985), "On the history of the minimum spanning tree problem", *Annals of the History of Computing* 7 (1985) 43–57. [248]

C. Greene and T. L. Magnanti (1975), "Some abstract pivot algorithms", *SIAM Journal on Applied Mathematics* 29 (1975) 530–539. [250]

M. Grötschel (1982), "Approaches to hard combinatorial optimization problems", in: B. Korte (ed.), *Modern Applied Mathematics – Optimization and Operations Research,* North-Holland, Amsterdam, 1982, 437–515. [223]

M. Grötschel and O. Holland (1985), "Solving matching problems with linear programming", *Mathematical Programming* 33 (1985) 243–259. [223]

M. Grötschel, M. Jünger and G. Reinelt (1984), "A cutting plane algorithm for the linear ordering problem", *Operations Research* 32 (1984) 1195–1220. [254]

M. Grötschel, M. Jünger and G. Reinelt (1985), "On the acyclic subgraph polytope", *Mathematical Programming* 33 (1985) 28–42. [254]

M. Grötschel, L. Lovász and A. Schrijver (1981), "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica* 1 (1981) 169–197. [65, 107, 159, 218, 283, 285]

M. Grötschel, L. Lovász and A. Schrijver (1984a), "Geometric methods in combinatorial optimization", in: W. R. Pulleyblank (ed.), *Progress in Combinatorial Optimization,* Academic Press, Toronto, 1984, 167–183. [159, 193]

M. Grötschel, L. Lovász and A. Schrijver (1984b), "Polynomial algorithms for perfect graphs", *Annals of Discrete Mathematics* 21 (1984) 325–356. [285]

M. Grötschel, L. Lovász and A. Schrijver (1984c), "Corrigendum to our paper "The ellipsoid method and its consequences in combinatorial optimization"", *Combinatorica* 4 (1984) 291–295. [326]

M. Grötschel, L. Lovász and A. Schrijver (1986), "Relaxations of vertex packing", *Journal of Combinatorial Theory (B)* 40 (1986) 330–343. [285]

M. Grötschel and M. W. Padberg (1985), "Polyhedral theory", in: E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys (eds.), *The Traveling Salesman Problem,* Wiley, Chichester, 1985, 251–305. [263]

M. Grötschel and W. R. Pulleyblank (1981), "Weakly bipartite graphs and the max-cut problem", *Operations Research Letters* 1 (1981) 23–27. [250]

M. Grötschel and W. R. Pulleyblank (1986), "Clique tree inequalities and the symmetric travelling salesman problem", *Mathematics of Operations Research* 11 (1986) 537–569. [265]

B. Grünbaum (1967), *Convex Polytopes,* Interscience-Wiley, London, 1967. [1]

F. Hadlock (1975), "Finding a maximum cut of a planar graph in polynomial time", *SIAM Journal on Computing* 4 (1975) 221–225. [250]

A. Hajnal and J. Surányi (1958), "Über die Auflösung von Graphen in vollständige Teilgraphen", *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Mathematica* 1 (1958) 113–121. [280]

M. Hall, Jr. (1956), "An algorithm for distinct representatives", *American Mathematical Monthly* 63 (1956) 716–717. [231]

D. B. Hartvigsen (1984), "Extensions of matching theory", Ph. D. Thesis, Department of Mathematics, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984. [265]

R. Hassin (1978), "On network flows", Ph. D. Thesis, Yale University, New Haven, Connecticut, 1978. [324]

R. B. Hayward (1985), "Weakly triangulated graphs", *Journal of Combinatorial Theory (B)* 39 (1985) 200–209. [282]

I. Heller (1957), "On linear systems with integral valued solutions", *Pacific Journal of Mathematics* 7 (1957) 1351–1364. [282]

Ch. Hermite (1850), "Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objets de la théorie des nombres", *Journal für die Reine und Angewandte Mathematik* 40 (1850) 261–278, 279–290, 291–307, 308–315 (reprinted in: É. Picard (ed.), *Oeuvres de Charles Hermite* Vol. I, Gauthier-Villars, Paris, 1905, 100–121, 122–135, 136–155, 155–163). [140]

Ch. Hermite (1851), "Sur l'introduction des variables continues dans la théorie des nombres", *Journal für die Reine und Angewandte Mathematik* 41 (1851) 191–216 (reprinted in: K. Hensel (ed.), *Oeuvres de Charles Hermite* Vol. III, Gauthier-Villars, Paris, 1905, 164–192). [43]

A. J. Hoffman (1960), "Some recent applications of the theory of linear inequalities to extremal combinatorial analysis", in: R. Bellman and M. Hall, Jr. (eds.), *Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics*, Volume X, American Mathematical Society, Providence, Rhode Island, 1960, 113–127. [238, 239, 241]

A. J. Hoffman and J. B. Kruskal (1956), "Integral boundary points of convex polyhedra", in: H. W. Kuhn and A. W. Tucker (eds.), *Linear Inequalities and Related Systems*, Princeton University Press, Princeton, New Jersey, 1956, 223–246. [199, 282]

I. Holyer (1981), "The $\mathcal{NP}$-completeness of edge-colouring", *SIAM Journal on Computing* 10 (1981) 718–721. [208]

J. Hopcroft and R. M. Karp (1973), "An $n^{\frac{5}{2}}$ algorithm for maximum matching in bipartite graphs", *SIAM Journal on Computing* 2 (1973) 225–231. [231]

J. Hopcroft and R. E. Tarjan (1974), "Efficient planarity testing", *Journal of the Association for Computing Machinery* 21 (1974) 549–568. [274]

W.-L. Hsu (1981), "How to color claw-free perfect graphs", *Discrete Mathematics* 11 (1981) 189–197. [282]

W.-L. Hsu, Y. Ikura and G. L. Nemhauser (1981), "A polynomial algorithm for maximum weighted vertex packings in graphs without long odd cycles", *Mathematical Programming* 20 (1981) 225–232. [302]

W.-L. Hsu and G. L. Nemhauser (1981), "Algorithms for minimum covering by cliques and maximum clique in claw-free perfect graphs", *Discrete Mathematics* 37 (1981) 181–191. [282]

W.-L. Hsu and G. L. Nemhauser (1982), "A polynomial algorithm for the minimum weighted clique cover problem on claw-free perfect graphs", *Discrete Mathematics* 38 (1982) 65–71. [282]

T. C. Hu (1963), "Multicommodity network flows", *Operations Research* 11 (1963) 344–360. [268]

T. C. Hu (1969), *Integer Programming and Network Flows*, Addison-Wesley, Reading, Massachusetts, 1969. [249]

O. H. Ibarra and C. E. Kim (1975), "Fast approximation algorithms for the knapsack and sum of subset problems", *Journal of the Association for Computing Machinery* 22 (1975) 463–468. [35]

A. W. Ingleton (1959), "A note on independence functions and rank", *Journal of the London Mathematical Society* 34 (1959) 49–56. [304]

M. Iri (1983), "Applications of matroid theory", in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: The State of the Art – Bonn 1982*, Springer, Berlin, 1983, 158–201. [211]

M. Iri and H. Imai (1986), "A multiplicative barrier function method for linear programming", *Algorithmica* 1 (1986) 455–482. [66]

P. M. Jensen and B. Korte (1981), "Complexity of matroid property algorithms", *SIAM Journal on Computing* 11 (1982) 184–190. [309]

F. John (1948), "Extremum problems with inequalities as subsidiary conditions", in: *Studies and Essays, presented to R. Courant on his 60th Birthday, January 8, 1948*, Interscience, New York, 1948, 187–204 (reprinted in: J. Moser (ed.), *Fritz John, Collected Papers Volume 2*, Birkhäuser, Boston, Massachusetts, 1985, 543–560). [69, 124]

D. S. Johnson, C. H. Papadimitriou, P. Seymour and M. Yannakakis (1984), "The complexity of multiway cuts", preprint, 1984. [267]

R. Kannan and A. Bachem (1979), "Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix", *SIAM Journal on Computing* 8 (1979) 499–507. [45, 151]

N. Karmarkar (1984), "A new polynomial-time algorithm for linear programming", in: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (Washington,

D. C., 1984), The Association for Computing Machinery, New York, 1984, 302–311 (revised version: *Combinatorica* 4 (1984) 373–395). [65]

R. M. Karp (1972), "Reducibility among combinatorial problems", in: R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 1972, 85–103. [28, 239, 249]

R. M. Karp and C. H. Papadimitriou (1980), "On linear characterization of combinatorial optimization problems", in: 21th Annual Symposium on Foundations of Computer Science (Syracuse, New York, 1980), IEEE, New York, 1980, 1–9 (final publication: *SIAM Journal on Computing* 11 (1982) 620–632). [65, 265]

L. G. Khachiyan (1979), "A polynomial algorithm in linear programming" (in Russian), *Doklady Akademii Nauk SSSR* 244 (1979) 1093–1096 (English translation: *Soviet Mathematics Doklady* 20 (1979) 191–194). [65, 158]

L. G. Khachiyan (1980), "Polynomial algorithms in linear programming" (in Russian), *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 20 (1980) 51–68 (English translation: *USSR Computational Mathematics and Mathematical Physics* 20 (1980) 53–72). [65]

L. G. Khachiyan, M. K. Kozlov and S. P. Tarasov (1980), "The polynomial solvability of convex quadratic programming" (in Russian), *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 20 (1980) 1319–1323. [181]

A. Khintchine (1956), *Kettenbrüche*, (translated from the Russian, published 1935), Teubner, Leipzig, 1956 (English translation: *Continued Fractions*, Noordhoff, Groningen, 1963). [132, 136]

V. Klee and G. J. Minty (1972), "How good is the simplex algorithm?", in: O. Shisha (ed.), *Inequalities III*, Academic Press, New York, 1972, 159–175. [42, 64]

D. E. Knuth (1973), "Matroid partitioning", Stanford University Report STAN-CS-73-342, Stanford, California, 1973. [250]

D. König (1915), "Vonalrendszerek és determinánsok" (Hungarian), ("Line-systems and determinants"), *Mathematikai és Természettudományi Értesitö* 33 (1915) 221–229. [230]

D. König (1916), "Gráfok és alkalmazásuk a determinánsok és halmazok elméletében" (Hungarian), *Mathematikai és Természettudományi Értesitö* 34 (1916) 104–119 (German translation: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre, *Mathematische Annalen* 77 (1916) 453–465). [209, 230, 279]

D. König (1931), "Graphok és matrixok" (Hungarian), ("Graphs and matrices"), *Matematikai és Fizikai Lapok* 38 (1931) 116–119. [230]

D. König (1933), "Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen)", *Acta Litterarum ac Scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae (Szeged), Sectio Scientiarum Mathematicarum* 6 (1933) 155–179. [230]

D. König (1936), *Theorie der endlichen und unendlichen Graphen*, Teubner, Leipzig, 1936 (reprinted: Chelsea, New York, 1950). [280]

J. B. Kruskal (1956), "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical Society* 7 (1956) 48–50. [247]

H. W. Kuhn (1955), "The Hungarian method for the assignment problem", *Naval Research Logistics Quarterly* 2 (1955) 83–97. [233, 279, 280]

H. W. Kuhn (1956), "Variants of the Hungarian method for assignment problems", *Naval Research Logistics Quarterly* 3 (1956) 253–258. [233]

Mei-Ko Kwan (1960), "Graphic programming using odd or even points" (in Chinese), *Acta Mathematica Sinica* 10 (1960) 263-266 (English translation: *Chinese Mathematics* 1 (1962) 273–277). [262]

J. C. Lagarias (1982), "The computational complexity of simultaneous diophantine approximation problems", in: 23rd Annual Symposium on Foundations of Computer Science, IEEE, New York, 1982, 32–39 (final version: *SIAM Journal on Computing* 14 (1985) 196–209). [138]

J. C. Lagarias (1984), "Knapsack public key cryptosystems and diophantine approximation", *Advances in Cryptology*, Proceedings of CRYPTO 83, Plenum Press, New York, 1984, 3–23. [221]

J. C. Lagarias and A. M. Odlyzko (1983), "Solving low-density subset sum problems", in: 24th Annual Symposium on Foundations of Computer Science, IEEE, New York, 1983, 1–10 (final version: *Journal of the Association for Computing Machinery* 32 (1985) 229–246). [218, 220, 221]

P. Lancaster and M. Tismenetsky (1985), *The Theory of Matrices*, Second Edition, *with Applications*, Academic Press, Orlando, 1985. [1]

E. L. Lawler (1975), "Matroid intersection algorithms", *Mathematical Programming* 9 (1975) 31–56. [216]

E. L. Lawler (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976. [16]

E. L. Lawler (1980), "The great mathematical Sputnik of 1979", *The Mathematical Intelligencer* 2 (1980) 191–198. [65]

E. L. Lawler and C. U. Martel (1982), "Flow network formulations of polymatroid optimization problems", *Annals of Discrete Mathematics* 16 (1982) 189–200. [324]

A. Lehman (1965), "On the width-length inequality", Mimeographic Notes, 1965, published in *Mathematical Programming* 16 (1979) 245–259 (without proof corrections), and 17 (1979) 403–417 (with proof corrections). [225, 226]

C. G. Lekkerkerker (1969), *Geometry of Numbers*, Wolters-Noordhoff, Groningen and North-Holland, Amsterdam, 1969. [139]

C. G. Lekkerkerker and J. Ch. Boland (1962), "Representation of a finite graph by a set of intervals on the real line", *Fundamenta Mathematicae* 51 (1962) 45–64. [280]

A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász (1982), "Factoring polynomials with rational coefficients", *Mathematische Annalen* 261 (1982) 515–534. [133, 149]

H. W. Lenstra, Jr. (1983), "Integer programming with a fixed number of variables", *Mathematics of Operations Research* 8 (1983) 538–548. [124, 193, 195]

A. Yu. Levin (1965), "On an algorithm for the minimization of convex functions" (in Russian), *Doklady Akademii Nauk SSSR* 160 (1965) 1244–1247 (English translation: *Soviet Mathematics Doklady* 6 (1965) 286–290). [65]

M. V. Lomonosov (1979), "Multiflow feasibility depending on cuts", *Graph Theory Newsletter* 9 (1) (1979) 4. [268]

L. Lovász (1970), "Subgraphs with prescribed valencies", *Journal of Combinatorial Theory* 8 (1970) 391–416. [259]

L. Lovász (1972), "Normal hypergraphs and the perfect graph conjecture", *Discrete Mathematics* 2 (1972) 253–267. [277]

L. Lovász (1975), "2-Matchings and 2-covers of hypergraphs", *Acta Mathematica Academiae Scientiarum Hungaricae* 26 (1975) 433–444. [261, 262]

L. Lovász (1976), "On two minimax theorems in graph", *Journal of Combinatorial Theory (B)* 21 (1976) 96–103. [246]

L. Lovász (1979), "On the Shannon capacity of a graph", *IEEE Transactions on Information Theory* 25 (1979) 1–7. [285]

L. Lovász (1980), "Matroid matching and some applications", *Journal of Combinatorial Theory B* 28 (1980) 208–236. [309]

L. Lovász (1983a), "Perfect graphs", in: L. W. Beineke and R. J. Wilson (eds.), *Selected Topics in Graph Theory 2*, Academic Press, London, 1983, 55–87. [283]

L. Lovász (1983b), "Submodular functions and convexity", in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: The State of the Art, Bonn, 1982*, Springer, Berlin, 1983, 235–257. [311]

L. Lovász (1986), *An Algorithmic Theory of Numbers, Graphs and Convexity* (CBMS-NSF Regional Conference Series in Applied Mathematics 50), SIAM, Philadelphia, Pennsylvania, 1986. [195]

L. Lovász and M. D. Plummer (1986), *Matching Theory,* Akadémiai Kiadó, Budapest, 1986. [230]

C. L. Lucchesi (1976), "A minimax equality for directed graphs", Ph. D. Thesis, University of Waterloo, Waterloo, Ontario, 1976. [252]

C. L. Lucchesi and D. H. Younger (1978), "A minimax theorem for directed graphs", *Journal of the London Mathematical Society (2)* 17 (1978) 369–374. [251]

G. Lueker (1974), "Structured breadth first search and chordal graphs", Technical Report TR-158, Princeton University, Princeton, New Jersey, 1974. [281]

W. Mader (1983), "On n-edge-connected digraphs", *Annals of Discrete Mathematics* 17 (1983) 439–441. [225, 247]

M. Marcus and H. Minc (1964), *A Survey of Matrix Theory and Matrix Inequalities,* Allyn and Bacon, Boston, Mass., 1964. [1]

A. B. Marsh (1979), "Matching algorithms", Ph. D. Thesis, Johns Hopkins University, Baltimore, Maryland, 1979. [257, 258]

J. F. Maurras, K. Truemper and M. Akgül (1981), "Polynomial algorithms for a class of linear programs", *Mathematical Programming* 21 (1981) 121–136. [282]

N. Megiddo (1983), "Towards a genuinely polynomial algorithm for linear programming", *SIAM Journal on Computing* 12 (1983) 347–353. [189]

K. Menger (1927), "Zur allgemeinen Kurventheorie", *Fundamenta Mathematicae* 10 (1927) 96–115. [238]

H. Meyniel (1976), "On the perfect graph conjecture", *Discrete Mathematics* 16 (1976) 339–342. [281]

H. Meyniel (1985), "A new property of imperfect graphs and some consequences", Working Paper, E. R. 1975 CNRS, Paris, 1985. [283]

S. Micali and V. V. Vazirani (1980), "An $O(V^{1/2}E)$ algorithm for finding a maximum matching in general graphs", in: 21st Annual Symposium on Foundations of Computer Science (Syracuse, New York), IEEE, New York, 1980, 17–27. [255]

G. J. Minty (1960), "Monotone networks", *Proceedings of the Royal Society of London, Series A* 257 (1960) 194–212. [238]

G. J. Minty (1980), "On maximal independent sets of vertices in claw-free graphs", *Journal of Combinatorial Theory B* 28 (1980) 284–304. [282, 302]

E. F. Moore (1959), "The shortest path through a maze", Proceedings of the International Symposium on Switching Theory 1957, Part II, Harvard University Press, Cambridge, Massachusetts, 1959, 285–292. [235]

J. Munkres (1957), "Algorithms for the assignment and transportation problems", *Journal of the Society for Industrial and Applied Mathematics* 5 (1957) 32–38. [233]

C. St. J. A. Nash-Williams (1961), "Edge-disjoint spanning trees of finite graphs", *Journal of the London Mathematical Society* 36 (1961) 445–450. [251]

C. St. J. A. Nash-Williams (1964), "Decomposition of finite graphs into forests", *Journal of the London Mathematical Society* 39 (1964) 12. [251]

G. L. Nemhauser and L. E. Trotter (1974), "Properties of vertex packing and independence system polyhedra", *Mathematical Programming* 6 (1974) 48–61. [301]

G. L. Nemhauser and L. A. Wolsey (1981), "Maximizing submodular set functions: formulations and analysis of algorithms", *Annals of Discrete Mathematics* 11 (1981) 279–301. [311]

I. Niven and H. S. Zuckerman (1980), *Introduction to the Theory of Numbers* (4-th edition), Wiley, New York, 1980. [134]

T. R. F. Nonweiler (1984), *Computational Mathematics — An Introduction to Numerical Approximation*, Ellis Horwood, Chichester, 1984. [134]

A. M. Odlyzko and H. J. J. te Riele (1985), "Disproof of the Mertens conjecture", *Journal für die Reine und Angewandte Mathematik* 357 (1985) 138–160. [149]

H. Okamura (1983), "Multicommodity flows in graphs", *Discrete Applied Mathematics* 6 (1983) 55–62. [268]

H. Okamura and P. D. Seymour (1981), "Multicommodity flows in planar graphs", *Journal of Combinatorial Theory (B)* 31 (1981) 75–81. [268]

A. Orden (1956), "The transhipment problem", *Management Science* 2 (1956) 276–285. [238]

O. Ore (1957), "Graphs and subgraphs I", *Transactions of the American Mathematical Society* 84 (1957) 109–136. [255]

O. Ore (1959), "Graphs and subgraphs II", *Transactions of the American Mathematical Society* 93 (1959) 185–204. [255]

J. B. Orlin (1987), "Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem", *Operations Research*, to appear. [238]

G. I. Orlova and Y. G. Dorfman (1972), "Finding the maximum cut in a graph" (in Russian), Izvestija Akademii Nauk SSSR, Tehničeskaja Kibernetika (1972) (3) 155–159, (English translation: *Engineering Cybernetics* 10 (3) (1972) 502–506). [250]

M. W. Padberg (1973), "On the facial structure of set packing polyhedra", *Mathematical Programming* 5 (1973) 199–215. [301]

M. W. Padberg (1975), "A note on zero-one programming", *Operations Research* 23 (1975) 833–837. [301]

M. W. Padberg and M. Grötschel (1985), "Polyhedral computations", in: E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys (eds.), *The Traveling Salesman Problem*, Wiley, Chichester, 1985, 307–360. [223, 263, 265]

M. W. Padberg and M. R. Rao (1981), "The Russian method for linear programming III: Bounded integer programming", Research Report 81-39, New York University, Graduate School of Business Administration, New York, 1981. [65]

M. W. Padberg and M. R. Rao (1982), "Odd minimum cut-sets and b-matchings", *Mathematics of Operations Research* 7 (1982) 67–80. [46, 207, 223, 257, 258, 259, 263]

M. W. Padberg and G. Rinaldi (1987), "Optimization of a 532 city symmetric travelling salesman problem by branch and cut", *Operations Research Letters* 6 (1987) 1–7. [265]

M. W. Padberg and L. A. Wolsey (1984), "Fractional covers for forests and matchings", *Mathematical Programming* 29 (1984) 1–14. [217]

C. H. Papadimitriou (1976), "On the complexity of edge traversing", *Journal of the Association for Computing Machinery* 23 (1976) 544–554. [262]

C. H. Papadimitriou and K. Steiglitz (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982. [236, 239]

K. R. Parthasarathy and G. Ravindra (1976), "The strong perfect graph conjecture is true for $K_{1,3}$-free graphs", *Journal of Combinatorial Theory (B)* 21 (1976) 212–223. [282]

O. Perron (1913), *Die Lehre von den Kettenbrüchen*, Teubner, Leipzig, 1913 (2nd revised edition: 1929, (reprinted: Chelsea, New York, 1950), 3rd revised edition: Teubner, Stuttgart, 1954 (Vol. I), 1957 (Vol. II)). [134]

G. Pólya and G. Szegö (1978), *Problems and Theorems in Analysis I: Series, Integral Calculus, Theory of Functions*, (Corrected Printing), Springer-Verlag, Berlin, 1978. [93]

A. van der Poorten (1979), "A proof that Euler missed ... Apéry's proof of the irrationality $\zeta(3)$. An informal report", *The Mathematical Intelligencer* 1 (1978/1979) 195–203. [35]

R. C. Prim (1957), "Shortest connection networks and some generalizations", *The Bell System Technical Journal* 36 (1957) 1389–1401. [235, 247]

W. R. Pulleyblank (1973), "Faces of matching polyhedra", Ph. D. Thesis, University of Waterloo, Waterloo, Ontario, 1973. [257, 258]

W. R. Pulleyblank (1980), "Dual integrality in b-matching problems", *Mathematical Programming Study* 12 (1980) 176–196. [257]

W. R. Pulleyblank (1981), "Total dual integrality and b-matchings", *Operations Research Letters* 1 (1981) 28–30. [257]

R. Rado (1942), "A theorem on independence relations", *The Quarterly Journal of Mathematics* Oxford (2) 13 (1942) 83–89. [304]

R. Rado (1957), "A note on independence functions", *Proceedings of the London Mathematical Society* 7 (1957) 300–320. [304]

A. Recski (1988), *Matroid Theory and its Applications in Electrical Networks and Statics*, Springer, Heidelberg, 1988, to appear. [211]

R. T. Rockafellar (1970), *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970. [1]

A. van Rooij and H. Wilf (1965), "The interchange graph of a finite graph", *Acta Mathematica Academiae Scientiarum Hungaricae* 16 (1965) 263–269. [280]

D. J. Rose and R. E. Tarjan (1975), "Algorithmic aspects of vertex elimination", in: Proceedings of the 7th Annual ACM Symposium on the Theory of Computing (Albuquerque, New Mexico, 1975), ACM, New York, 1975, 245–254. [281]

D. J. Rose, R. E. Tarjan and G. S. Lueker (1976), "Algorithmic aspects of vertex elimination on graphs", *SIAM Journal on Computing* 5 (1976) 266–283. [281]

H. Sachs (1970), "On the Berge conjecture concerning perfect graphs", in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, 377–384. [281]

N. Sbihi (1978), "Étude des stables dans les graphes sans étoile", M. Sc. Thesis, Université Scientifique et Medicale, Grenoble, 1978. [282, 302]

N. Sbihi (1980), "Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans etoile", *Discrete Mathematics* 29 (1980) 53–76. [282, 302]

W. M. Schmidt (1980), *Diophantine Approximation*, Lecture Notes in Mathematics 785, Springer-Verlag, Berlin, 1980. [146]

C. P. Schnorr (1985), "A hierarchy of polynomial time basis reduction algorithms", in: L. Lovász and E. Szemerédi (eds.), *Theory of Algorithms*, North-Holland, Amsterdam, 1985, 375–386. [146]

R. Schrader (1982), "Ellipsoid methods", in: B. Korte (ed.), *Modern Applied Mathematics — Optimization and Operations Research*, North-Holland, Amsterdam, 1982, 265–311. [65]

A. Schrijver (1980a), "On cutting planes", *Annals of Discrete Mathematics* 9 (1980) 291–296. [13]

A. Schrijver (1980b), "A counterexample to a conjecture of Edmonds and Giles", *Discrete Mathematics* 32 (1980) 213–214. [252]

A. Schrijver (1982), "Min-max relations for directed graphs", *Annals of Discrete Mathematics* 16 (1982) 261–280. [252]

A. Schrijver (1983), "Packing and covering of crossing families of cuts", *Journal of Combinatorial Theory (B)* 35 (1983) 104–128. [252]

A. Schrijver (1984), "Total dual integrality from directed graphs, crossing families, and sub- and supermodular functions", in: W. R. Pulleyblank (ed.), *Progress in Combinatorial Optimization*, Academic Press, Toronto, 1984, 315–361. [324]

A. Schrijver (1986), *Theory of Linear and Integer Programming*, Wiley, Chichester, 1986. [1, 199]

A. Schrijver and P. D. Seymour (1977), "A proof of total dual integrality of matching polyhedra", Report ZN 79/77, Mathematisch Centrum, Amsterdam, 1977. [259]

P. D. Seymour (1977), "The matroids with the max-flow min-cut property", *Journal of Combinatorial Theory (B)* 23 (1977) 189–222. [261]

P. D. Seymour (1979), "On multicolourings of cubic graphs, and conjectures of Fulkerson and Tutte", Proceedings of the London Mathematical Society (3) 38 (1979) 423–460. [262]

P. D. Seymour (1980a), "Decomposition of regular matroids", *Journal of Combinatorial Theory B* 28 (1980) 305–359. [200, 282]

P. D. Seymour (1980b), "Four-terminus flows", *Networks* 10 (1980) 79–86. [268]

P. D. Seymour (1981a), "On odd cuts and plane multicommodity flows", Proceedings of the London Mathematical Society (3) 42 (1981) 178–192. [261, 268]

P. D. Seymour (1981b), "Matroids and multicommodity flows", *European Journal of Combinatorics* 2 (1981) 257–290. [250]

N. Z. Shor (1970a), "Utilization of the operation of space dilatation in the minimization of convex functions" (in Russian), *Kibernetika (Kiev)* 1970 (1) (1970) 6–12 (English translation: *Cybernetics* 6 (1970) 7–15). [65]

N. Z. Shor (1970b), "Convergence rate of the gradient descent method with dilatation of the space" (in Russian), *Kibernetika (Kiev)* 1970 (2) (1970) 80–85 (English translation: *Cybernetics* 6 (1970) 102–108). [65]

N. Z. Shor (1977), "Cut-off method with space extension in convex programming problems" (in Russian), *Kibernetika (Kiev)* 1977 (1) (1977) 94–95 (English translation: *Cybernetics* 13 (1977) 94–96). [56, 65]

N. Z. Shor and V. I. Gershovich (1979), "Family of algorithms for solving convex programming problems" (in Russian), *Kibernetika (Kiev)* 1979 (4) (1979) 62–67 (English translation: *Cybernetics* 15 (1979) 502–508). [94]

S. Smale (1983), "On the average number of steps in the simplex method of linear programming", *Mathematical Programming* 27 (1983) 241–262. [42]

G. Sonnevend (1986), "An "analytical centre" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming", in: A. Prékopa, J. Szelezsán and B. Strazicky (eds.), *System Modelling and Optimization* (Proceedings of the 12-th IFIP Conference, Budapest, 1985), Lecture Notes in Control and Information Sciences 84, Springer, Berlin 1986, 866–875. [66]

J. Stoer and C. Witzgall (1970), *Convexity and Optimization in Finite Dimensions I*, Springer, Berlin, 1970. [1]

G. Strang (1980), *Linear Algebra and Its Applications*, (2nd edition), Academic Press, New York, 1980. [1]

V. Strassen (1973), "Vermeidung von Divisionen", *Journal für die Reine und Angewandte Mathematik* 264 (1973) 184–202. [40]

É. Tardos (1985), "A strongly polynomial minimum cost circulation algorithm", *Combinatorica* 5 (1985) 247–255. [189, 238]

É. Tardos (1986), "A strongly polynomial algorithm to solve combinatorial linear programs", *Operations Research* 34 (1986) 250–256. [158, 189, 190]

R. E. Tarjan (1974), "A good algorithm for edge-disjoint branchings", *Information Processing Letters* 3 (1974) 51–53. [246]

R. E. Tarjan (1977), "Finding optimum branchings", *Networks* 7 (1977) 25–35. [242]

R. E. Tarjan (1986), "Algorithms for maximum network flow", *Mathematical Programming Study* 26 (1986) 1–11. [237]

C. Thomassen (1985), "Even cycles in directed graphs", *European Journal of Combinatorics* 6 (1985) 85–89. [236]

L. E. Trotter (1977), "Line perfect graphs", *Mathematical Programming* 12 (1977) 255–259. [281]

K. Truemper (1987), "A decomposition theory for matroids V: testing of matrix total unimodularity", Working Paper, University of Texas at Dallas, Richardson, Texas, January 1987. [201]

W. T. Tutte (1947), "The factorization of linear graphs", *Journal of the London Mathematical Society* 22 (1947) 107–111. [204, 255]

W. T. Tutte (1952), "The factors of graphs", *Canadian Journal of Mathematics* 4 (1952) 314–328. [255, 258]

W. T. Tutte (1954), "A short proof of the factor theorem for finite graphs", *Canadian Journal of Mathematics* 6 (1954) 347–352. [255]

W. T. Tutte (1961), "On the problem of decomposing a graph into n connected factors", *Journal of the London Mathematical Society* 36 (1961) 221–230. [251]

W. T. Tutte (1974), "Spanning subgraphs with specified valencies", *Discrete Mathematics* 9 (1974) 97–108. [259]

W. T. Tutte (1978), "The subgraph problem", *Annals of Discrete Mathematics* 3 (1978) 289–295. [259]

V. G. Vizing (1964), "On an estimate of the chromatic class of a p-graph" (in Russian), *Diskretnyĭ Analiz* 3 (1964) 25–30. [209]

S. Warshall (1962), "A theorem on boolean matrices", *Journal of the Association for Computing Machinery* 9 (1962) 11–12. [235]

D. J. A. Welsh (1976), *Matroid Theory,* Academic Press, London, 1976. [211]

S. H. Whitesides (1984), "A classification of certain graphs with minimal imperfection properties", *Annals of Discrete Mathematics* 21 (1984) 207–218. [281]

M. A. Yakovleva (1959), "A problem on minimum transportation cost", in: V. S. Nemchinov (ed.), *Applications of Mathematics in Economic Research,* Izdat. Social'no-Èkon. Lit., Moscow, 1959, 390–399. [238]

B. Yamnitsky and L. A. Levin (1982), "An old linear programming algorithm runs in polynomial time", in: 23rd Annual Symposium on Foundations of Computer Science, IEEE, New York, 1982, 327–328. [65, 104]

A. C. Yao (1975), "An ($|E| \log\log |V|$) algorithm for finding minimum spanning trees", *Information Processing Letters* 4 (1975) 21–23. [247]

D. B. Yudin and A. S. Nemirovskiĭ (1976a), "Evaluation of the informational complexity of mathematical programming problems" (in Russian), *Èkonomika i Matematicheskie Metody* 12 (1976) 128–142 (English translation: *Matekon* 13 (2) (1976-7) 3–25). [65]

D. B. Yudin and A. S. Nemirovskiĭ (1976b), "Informational complexity and efficient methods for the solution of convex extremal problems" (in Russian), *Èkonomika i Matematicheskie Metody* 12 (1976) 357–369 (English translation: *Matekon* 13 (3) (1977) 25–45). [65, 94, 107]

Notation Index

In the order of first occurrence in the text.

Author Index

Subject Index

Boldface numbers refer to pages on which items are introduced.