MODELS AND ALGORITHMS FOR GLOBAL OPTIMIZATION

Essays Dedicated to Antanas Žilinskas on the Occasion of His 60th Birthday

Aims and Scope
Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics and other sciences.
The series *Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs and state-of-the-art expository works that focus on algorithms for solving optimization problems and also study applications involving such problems. Some of the topics covered include nonlinear optimization (convex and nonconvex), network flow problems, stochastic optimization, optimal control, discrete optimization, multi-objective programming, description of software packages, approximation techniques and heuristic approaches.

MODELS AND ALGORITHMS FOR GLOBAL OPTIMIZATION

Essays Dedicated to Antanas Žilinskas on the Occasion of His 60th Birthday

Edited by

AIMO TÖRN
Åbo Akademi University, Åbo, Finland

JULIUS ŽILINSKAS
Institute of Mathematics and Informatics, Vilnius, Lithuania

Springer

Library of Congress Control Number: 2006930115

ISBN-10: 0-387-36720-9 e-ISBN: 0-387-36721-7

ISBN-13: 978-0-387-36720-0

Printed on acid-free paper.

AMS Subject Classifications: 49-02, 65Kxx

Printed in the United States of America.

9 8 7 6 5 4 3 2 1

springer.com

Dedicated to Antanas Žilinskas on the occasion of his 60th birthday

Antanas Žilinskas

Preface

Antanas Žilinskas was born on January 5, 1946 in Lithuania. He graduated with a gold medal from 2nd Kaunas Gymnasium in 1963 and with a distinction diploma of Electrical Engineering from Kaunas University of Technology in 1968. His Ph.D. studies (aspirantur) at Lithuanian Academy of Sciences lasted from 1970 to 1973. The Candidate of Sciences (Ph.D.) degree in Technical Cybernetics (1973) has been received from Kaunas University of Technology. The Doctor of Mathematical Sciences degree (Habilitation, 1985) has been received from St. Petersburg (Leningrad) University. The title Senior Research Fellow (1980) has been conferred by the Presidium of Academy of Sciences, and the title Professor (1989) by Vilnius Pedagogical University. He has been awarded (with V. Šaltenis and G. Dzemyda) Lithuanian National Award for scientific achievements of 2001 for the research on "Efficient optimization methods and their applications".

A. Žilinskas joined the Institute of Mathematics and Informatics in 1973 starting with a position of junior research associate, worked as a senior research associate reaching the highest rank of principal researcher which is his main position now. Apart from working in the research institute he was a lecturer at Vilnius Pedagogical University 1986–1988, where he founded a department of Informatics in 1988 and held a position of professor and head of this department 1988–1993. He worked later as a professor of this department until 2000. He founded a Department of Applied Informatics at Vytautas Magnus University in 1994, and from then he holds a position of professor and head of this department. A. Žilinskas taught Optimization theory and methods at all levels; Operations research; Analysis of algorithms at all levels; Calculus, Statistics, and Linear algebra for undergraduates.

A. Žilinskas held a visiting Konrad Zuse professorship at Dortmund University (1990/1991 academic year). As a visiting research professor he worked also at Åbo Akademi, Technical University Aachen, Copenhagen University, London University (UCL).

A. Žilinskas is an academician of International Engineering Academy. He is a member of American Mathematical Society, IEEE including Computer

Society and Computational Intelligence Society, IFIP Working Group 7.6 on Computer Aided Optimization Based Modeling and Optimization. He is a member of editorial boards of international journals Journal of Global Optimization, Control and Cybernetics, Informatica. He is a reviewer for Mathematical Reviews, Zentralblatt für Mathematic, book section of INFORMS Interfaces.

Many projects were fulfilled by A. Žilinskas for industry in seventies and eighties; e.g. the results of optimal design of magnetic deflection systems of color TV sets, and of optimal design of pigment mixtures for paint technology are referenced in the book Global Optimization, Springer, 1989, written with A. Törn. He was a chairman of Lithuanian part of international project Computing, Information Services and the Internet, which was fulfilled in 1996–1997 cooperating with Växjo University (Sweden). He was a Managing Director of TEMPUS project Modelling of Economics and Business Systems funded by EU in 1997–2000 with participation of Vytautas Magnus University, Kaunas University of Technology from Lithuania, and Copenhagen University (Denmark), Maastricht University (Netherlands) from EU. He was a partner (with Prof. J. Calvin) in the project Probabilistic Analysis of Global Optimization Algorithms funded by National Research Council (USA) under Collaboration in Basic Science and Engineering Program 1998–2000.

A. Žilinskas has published more than 100 papers mainly on statistical global optimization theory, algorithms and applications, 5 monographs and 6 textbooks; the titles of the monographs follow:

- Žilinskas, A.: Global Optimization: Axiomatic of Statistical Models; Algorithms; Applications. Mokslas (1986) (in Russian),
- Törn, A., Žilinskas, A.: Global Optimization. Springer (1989),
- Šaltenis, V., Žilinskas, A.: Search for Optimum. Nauka (1989),
- Zhigljavsky, A., Žilinskas, A.: Methods of Search for Global Extremum. Nauka (1991) (in Russian),
- Žilinskas, A. (ed) System Analysis, Design and Optimization. Space Technology (1993),

He was a co-editor of the book

- Dzemyda, G., Šaltenis, V., Žilinskas, A. (eds) Stochastic and Global Optimization. Kluwer (2002).

Current research interests of A. Žilinskas are statistical theory of global optimization, optimization based modeling and design, analysis of multidimensional data by means of visualization. Research is oriented to develop statistical models of global optimization, implement and investigate the corresponding algorithms, and apply them to practical problems.

This book is dedicated to A. Žilinskas on the occasion of his 60th birthday. The chapters cover research interests of A. Žilinskas. The book is divided into six parts: I. Advanced Models in Optimization Theory; II. Interval Algorithms;

III. Deterministic Optimization Models and Algorithms; IV. Stochastic Algorithms; V. Educational Aspects; and VI. Applications.

Part I consists of two chapters at the forefront of research. Chinchuluun and Pardalos consider optimality conditions and duality for multiobjective programming problems with generalized convexity. Floudas and Kreinovich consider the problem of selecting the best auxiliary function within a given global optimization technique.

Part II consists of four chapters on interval algorithms for global optimization. Kjøller, Kozine, Madsen and Stauning describe interval global optimization and constraint propagation for solving systems of non-linear equations, and combine them to improve performance of global optimization. Kosheleva discusses optimal data compression under interval uncertainty. Pedamallu, Özdamar and Csendes present interval partitioning algorithm for continuous constrained global optimization problems. Žilinskas and Bogle review estimation of ranges of functions combining interval arithmetic and underestimating interval arithmetic.

Part III consists of four chapters on deterministic optimization models and algorithms. Antamoshkin and Masich consider heuristic algorithms for a constrained pseudo-Boolean optimization problem. Sergeyev, Khalaf and Kvasov review Lipschitz univariate constrained global optimization algorithms for solving problems with multiextremal non-differentiable constraints. Szabó and Specht review models and algorithms for the packing of equal circles in a square. Šaltenis presents simulation of wet film evolution for solving Euclidean Steiner problem.

Part IV consists of four chapters on stochastic algorithms for global optimization. Ali presents a probabilistic hybrid differential evolution algorithm. Calvin concerns nonadaptive univariate optimization using Wiener process model assuming that the function values are corrupted by independent Gaussian noise. Hamilton, Savani and Zhigljavsky consider linear and maximum likelihood estimators of the minimum of a function in global random search methods. Molvalioglu, Zabinsky and Kohn presents a multi-particle version of simulated annealing where points in the population interact with each other.

Part V concerns educational aspects of global optimization. Hendrix shows how the concepts of global optimization algorithms may be introduced to students and researchers. Mockus discusses Internet aided distance graduate studies of the course on optimal sequential decisions with several objective functions.

Part VI consists of five chapters on applications of global optimization. Bernatavičienė, Dzemyda, Kurasova, Marcinkevičius and Medvedev consider visual analysis of multidimensional medical data. Čiegis describes applications where global optimization is essential part of mathematical modeling cycle. Fraga and Papageorgiou apply hybrid optimization for the design of optimal water distribution networks. Kaminskas considers practical issues in the implementation of self-tuning control systems. McAllister, Rajgaria and Floudas address global pairwise sequence alignment problem in a mathematically-detailed, rigorous and generic manner using mixed-integer linear programming.

On behalf of all the contributors of this Festschrift we would like to congratulate Antanas Žilinskas on the occasion of his 60th birthday and to wish him well and continued success in scientific career.

Åbo, Finland
Vilnius, Lithuania
January 2006

Aimo Törn
Julius Žilinskas

Contents

List of Contributors

Montaz M. Ali
Witwatersrand University, South Africa

Alexander Antamoshkin
Siberian State Aerospace University, Russia

Jolita Bernatavičienė
Institute of Mathematics and Informatics, Lithuania

Ian David Lockhart Bogle
University College London, UK

James M. Calvin
New Jersey Institute of Technology, USA

Altannar Chinchuluun
University of Florida, USA

Tibor Csendes
University of Szeged, Hungary

Raimondas Čiegis
Vilnius Gediminas Technical University, Lithuania

Gintautas Dzemyda
Institute of Mathematics and Informatics, Lithuania

Christodoulos A. Floudas
Princeton University, USA

Eric S. Fraga
University College London, UK

Emily Hamilton
Cardiff University, UK

Eligius M.T. Hendrix
Wageningen Universiteit, The Netherlands

Vytautas Kaminskas
Vytautas Magnus University, Lithuania

Falah M.H. Khalaf
University of Calabria, Italy

Steffen Kjøller
Technical University of Denmark

Wolf Kohn
Clearsight Systems Inc., USA

Olga Kosheleva
University of Texas, USA

Pavel Kozine
Technical University of Denmark

Vladik Kreinovich
University of Texas at El Paso, USA

Olga Kurasova
Institute of Mathematics and Informatics, Lithuania

Dmitri E. Kvasov
N.I. Lobatchevsjy State University, Russia;
University of Rome "La Sapienza", Italy

Kaj Madsen
Technical University of Denmark

Igor Masich
Siberian State Aerospace University, Russia

Virginijus Marcinkevičius
Institute of Mathematics and Informatics, Lithuania

Scott R. McAllister
Princeton University, USA

Viktor Medvedev
Institute of Mathematics and Informatics, Lithuania

Jonas Mockus
Institute of Mathematics and Informatics, Lithuania

Orcun Molvalioglu
University of Washington, USA

Linet Özdamar
Izmir Ekonomi Universitesi, Turkey

Lazaros G. Papageorgiou
University College London, UK

Panos M. Pardalos
University of Florida, USA

Chandra Sekhar Pedamallu
Nanyang Technological University, Singapore

Rohit Rajgaria
Princeton University, USA

Vippal Savani
Cardiff University, UK

Yaroslav D. Sergeyev
University of Calabria, Italy;
N.I. Lobatchevsjy State University, Russia

Eckard Specht
University of Magdeburg, Germany

Ole Stauning
Saxo Bank, Denmark

Péter Gábor Szabó
University of Szeged, Hungary

Vydūnas Šaltenis
Institute of Mathematics and Informatics, Lithuania

Zelda B. Zabinsky
University of Washington, USA

Anatoly Zhigljavsky
Cardiff University, UK

Julius Žilinskas
Institute of Mathematics and Informatics, Lithuania

Multiobjective Programming Problems Under Generalized Convexity*

Altannar Chinchuluun and Panos M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL32611, USA altannar@ufl.edu,pardalos@ufl.edu

Summary. In this chapter, we consider optimality conditions and duality for some multiobjective programming problems with generalized convexity. In particularly, the general multiobjective programming, multiobjective fractional programming and multiobjective variational programming will be discussed.

1 Introduction

A multiobjective programming problem can be defined in the form

$$\begin{aligned} \text{(MOP)} \ \min\ & f(x) \\ \text{s.t.}\ & x \in S = \{x \in X \mid g_j(x) \leq 0, j = 1, 2, \ldots, q\}, \end{aligned}$$

where $f(x) = (f_1, f_2, \ldots, f_p)^T$, $f_i(x) = \ : X \to \mathbb{R}$ are differentiable functions on a nonempty open set $X \subseteq \mathbb{R}^n$.

If all the objective functions f_i, $i = 1, \ldots, p$, and the constraint functions g_j, $j = 1, \ldots, q$, are convex, then (MOP) is called a convex multiobjective program. If at least one of the objective functions or the constraint set is nonconvex, then MOP is a nonconvex multiobjective program.

Definition 1. *A feasible point $x_0 \in S$ is efficient if and only if there exists no point $x \in S$ such that $f_i(x) \leq f_i(x_0)$ for all $i = 1, 2, \ldots, p$ and $f_j(x) < f_j(x_0)$ for at least one index $j \in \{1, 2, \ldots, p\}$.*

Definition 2. *A feasible point $x_0 \in S$ is weak efficient if and only if there exists no point $x \in S$ such that $f_i(x) < f_i(x_0)$ for all $i = 1, 2, \ldots, p$.*

We note that every efficient solution is weak efficient, however, the converse is not always true.

Readers are referred to [Mie99, PSZ95, CP05] for more comprehensive survey on multiobjective optimization.

* This research was partially supported by NSF, Air Force, and CRDF grants.

Throughout this chapter, the following notations will be used: For $x_0 \in S$, the index set of the equality constraints is denoted by $I = \{j | g_j(x_0) = 0\}$. If x and $y \in \mathbb{R}^n$, then

$x \leqq y \Leftrightarrow x_i \leq y_i, i = 1, \dots, n;$
$x \leqslant y \Leftrightarrow x_i \leq y_i, i = 1, \dots, n$ and $x \neq y;$
$x < y \Leftrightarrow x_i < y_i, i = 1, \dots, n.$

For any $a, b, c \in \mathbb{R}^s$, we also use the following notation:

$$\frac{ab}{c} = \left(\frac{a_s b_s}{c_s}, \frac{a_s b_s}{c_s}, \dots, \frac{a_s b_s}{c_s} \right).$$

Next, we present well known Karush-Kuhn-Tucker necessary optimality conditions (see, for example, [Mie99]).

Theorem 1. *Let $x_0 \in S$ be a (weak) efficient solution of Problem (MOP). Suppose that a constraint qualification (Karush-Kuhn-Tucker constraint qualification, etc.) is satisfied at $x_0 \in S$. Then the following necessary optimality conditions hold:*

$$\sum_{i=1}^{p} u_i \nabla f_i(x_0) + \sum_{j=1}^{q} v_j \nabla g_j(x_0) = 0 \tag{1}$$

$$v_j g_j(x_0) = 0 \textit{ for all } j = 1, 2, \dots, q, \tag{2}$$

$$u \geqslant 0, \ v \geqq 0. \tag{3}$$

The present chapter is organized as follows. In the next section, we present some generalized convexities. In Sect. 3, sufficient optimality conditions for Problem (MOP) based on generalized convexity assumptions are discussed. We also present Mond-Weir type dual of (MOP) and establish weak and strong duality theorems. A special case of (MOP), a multiobjective fractional programming problem, is considered in Sect. 5 while Sect. 6 discusses a multiobjective variational programming problem.

2 Generalized Convexity

Convexity plays an important role in optimality conditions and duality theory of mathematical programming [Ber95, Roc70]. Various generalizations of convexity have been introduced in the literature in order to relax convexity assumptions [HM87, Pre92, HM82, HM87, Via82, Via83]. Hanson [Han81] introduced the concept of invexity for scalar problems.

Definition 3. *The scalar problem (MOP) ($p = 1$) is invex if there exists a function $\eta : X \times X \to \mathbb{R}^n$ such that*

$$f_1(x) - f_1(x_0) \geq \nabla f(x_0) \eta(x, x_0),$$

$$g(x) - g(x_0) \geqq \nabla g(x_0) \eta(x, x_0), \quad \forall x, x_0 \in X.$$

The definition of invexity in the sense of Hanson reduces to the notion of convexity when $\eta(x, x_0) = x - x_0$.

Later, Jeyakumar and Mond [JM92] introduced V-invexity, which is an extension of invexity, for multiobjective programming problems.

Definition 4. *Let $f : X \to \mathbb{R}^p$ be a real vector function defined on an open set $X \subseteq \mathbb{R}^n$ and each component of f be differentiable at x_0. The function f is said to be V-invex at $x_0 \in X$ if there exist a mapping $\eta : X \times X \to \mathbb{R}^n$ and a function $\alpha_i : X \times X \to \mathbb{R}_+ \setminus \{0\}$ $(i = 1, \dots, p)$ such that, $\forall x \in X$,*

$$f_i(x) - f_i(x_0) \geq \alpha_i(x, x_0) \nabla f_i(x_0)^T \eta(x, x_0).$$

Note that, when $p = 1$, the definition of V-invexity reduces to the notion of convexity in the sense of Hanson with $\alpha_1(x, x_0) = 1$.

Aghezzaf and Hachimi [AH00] considered Problem (MOP) involving type I functions defined by Hanson and Mond [HM87], and established some sufficient optimality conditions and duality results for differentiable multiobjective programming problems.

Definition 5. *(f, g) is said to be type I with respect to η at $x_0 \in S$ if there exists a vector function $\eta(x, x_0)$ defined on $X \times X$ such that, for all $x \in S$,*

$$\begin{aligned} f(x) - f(x_0) &\geqq \nabla f(x_0)^T \eta(x, x_0) \\ -g(x_0) &\geqq \nabla g(x_0)^T \eta(x, x_0). \end{aligned}$$

If (f, g) is invex in the sense of Hanson, then it is also type I. However, the converse is not necessary true.

Preda [Pre92] introduced (F, ρ)-convexity which is an extension of F-convexity defined by Hanson and Mond [HM82] and ρ-convexity defined by [Via83]. Liang et al. introduced (F, α, ρ, d)-convexity, and obtained some corresponding optimality conditions and duality results for scalar nondifferentiable fractional programming problems [LHP01] and differentiable multiobjective fractional programming problems [LHP03]. More recently, Yuan et al. [YLCP05] introduced a unified formulation of generalized convexity, which was called (C, α, ρ, d)-convexity, and considered nondifferentiable minimax fractional programming problems involving the generalized convexity. Therefore, Chinchuluun et al. [CYP05] established optimality conditions and duality theorems for nondifferentiable multiobjective fractional programming problems with (C, α, ρ, d) convexity.

Let $\theta : X \to \mathbb{R}^p$, $\vartheta : X \to \mathbb{R}^q$; $\alpha, \alpha^1 : X \times X \to \mathbb{R}^p_+ \setminus \{0\}$, $\alpha^2 : X \times X \to \mathbb{R}^q_+ \setminus \{0\}$; $d, d^1 : X \times X \to \mathbb{R}^p_+$, $d^2 : X \times X \to \mathbb{R}^q_+$; and $\rho^1 \in \mathbb{R}^p$, $\rho^2 \in \mathbb{R}^q$.

Definition 6. *A differentiable function $\theta : X \to \mathbb{R}$ is said to be (strictly) (C, α, ρ, d)-convex at $x_0 \in S$ if, $\forall\ x \in S$, the inequality*

$$\frac{\theta(x) - \theta(x_0)}{\alpha(x, x_0)} \geqslant (>) C_{(x, x_0)}(\nabla \theta(x_0)) + \rho \frac{d(x, x_0)}{\alpha(x, x_0)},$$

where $C : X \times X \times \mathbb{R}^n \to \mathbb{R}$ is convex on $\mathbb{R}^n$ with respect to the third argument (denoted by $C_{(x,x_0)}(\cdot)$) and $C_{(x,x_0)}(0) = 0$ for any $(x, x_0) \in S \times S$, holds. The function θ is said to be (C, α, ρ, d)-convex over S if, $\forall\, x_0 \in S$, it is (C, α, ρ, d)-convex at x_0. In particular, θ is said to be strongly (C, α, ρ, d)- convex or (C, α)-convex with respect to $\rho > 0$ or $\rho = 0$, respectively.

Motivated by [HM87, HA04, CYP05], Chinchuluun et al. [YCLP05] introduced (C, α, ρ, d)-type I functions and considered nondifferentiable multiobjective programming problems with the generalized convexity.

Definition 7. *(θ, ϑ) is said (C, α, ρ, d)-type I at x_0, if for all $x \in S$ we have*

$$\frac{\theta(x) - \theta(x_0)}{\alpha^1(x, x_0)} \geqq C_{(x,x_0)}(\nabla\theta(x_0)) + \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)},$$
$$\frac{-\vartheta(x_0)}{\alpha^2(x, x_0)} \geqq C_{(x,x_0)}(\nabla\vartheta(x_0)) + \frac{\rho^2 d^2(x, x_0)}{\alpha^2(x, x_0)}.$$

We note that the function $C_{(x,x_0)}$ is applied to each θ_i $(i = 1, \ldots, p)$ and ϑ_j $(j = 1, \ldots, q)$, and results in a vector.

Definition 8. *(θ, ϑ) is said pseudoquasi (strictly pseudoquasi) (C, α, ρ, d) - type I at x_0, if for all $x \in S$ we have*

$$\theta(x) < (\leqq)\theta(x_0) \Rightarrow C_{(x,x_0)}(\nabla\theta(x_0)) + \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)} < 0,$$
$$-\vartheta(x_0) \leqq 0 \Rightarrow C_{(x,x_0)}(\nabla\vartheta(x_0)) + \frac{\rho^2 d^2(x, x_0)}{\alpha^2(x, x_0)} \leqq 0.$$

Definition 9. *(θ, ϑ) is said weak strictly-pseudoquasi (C, α, ρ, d)-type I at x_0, if for all $x \in S$ we have*

$$\theta(x) \leqslant \theta(x_0) \quad \Rightarrow \quad C_{(x,x_0)}(\nabla\theta(x_0)) + \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)} < 0,$$
$$-\vartheta(x_0) \leqq 0 \quad \Rightarrow \quad C_{(x,x_0)}(\nabla\vartheta(x_0)) + \frac{\rho^2 d^2(x, x_0)}{\alpha^2(x, x_0)} \leqq 0.$$

Definition 10. *(θ, ϑ) is said strong pseudoquasi (weak pseudoquasi) (C, α, ρ, d)-type I at x_0, if for all $x \in S$ we have*

$$\theta(x) \leqslant (<)\theta(x_0) \quad \Rightarrow \quad C_{(x,x_0)}(\nabla\theta(x_0)) + \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)} \leqslant 0,$$
$$-\vartheta(x_0) \leqq 0 \quad \Rightarrow \quad C_{(x,x_0)}(\nabla\vartheta(x_0)) + \frac{\rho^2 d^2(x, x_0)}{\alpha^2(x, x_0)} \leqq 0.$$

Note that for the scalar objective functions, the class of pseudoquasi (C, α, ρ, d)-type I, the class of weak strictly-pseudoquasi (C, α, ρ, d)-type I, and the class of strong pseudoquasi (C, α, ρ, d)-type I functions coincide.

3 Sufficient Conditions

In this section, we present sufficient optimality conditions for Problem (MOP) based on the notions of generalized convexity defined in the previous section.

Theorem 2. *Assume that there exist a feasible solution x_0 for (MOP) and vectors $\bar{u} \in \mathbb{R}^p$ and $\bar{v} \in \mathbb{R}^q$ such that*

$$\bar{u}^T \nabla f(x_0) + \bar{v}^T \nabla g(x_0) = 0, \tag{4}$$

$$\bar{v}^T g(x_0) = 0, \tag{5}$$

$$\bar{u} > 0, \bar{v} \geqq 0. \tag{6}$$

If $(f,\, g_I)$ is strong pseudoquasi $(C,\, \alpha,\, \rho,\, d)$-type I at x_0 with

$$\sum_{i=1}^{p} \bar{u}_i \rho_i^1 \frac{d_i^1(x, x_0)}{\alpha_i^1(x, x_0)} + \sum_{j \in I} \bar{v}_j \rho_j^2 \frac{d_j^2(x, x_0)}{\alpha_j^2(x, x_0)} \geqq 0, \tag{7}$$

then x_0 is an efficient solution for (MOP).

Proof. Suppose that x_0 is not an efficient solution of (MOP). Then there exists a feasible solution x such that

$$f(x) \leqslant f(x_0) \text{ and } g_I(x_0) = 0$$

or

$$f(x) \leqslant f(x_0) \text{ and } -g_I(x_0) \leqq 0.$$

According to the strong pseudoquasi assumption on (f, g_I), it follows that

$$C_{(x,x_0)}(\nabla f(x_0)) + \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)} \leqslant 0,$$

$$C_{(x,x_0)}(\nabla g_I(x_0)) + \frac{\rho_I^2 d_I^2(x, x_0)}{\alpha_I^2(x, x_0)} \leqq 0.$$

Using the facts that $\bar{u} > 0$ and convexity of C, from the above inequalities, we can conclude that

$$C_{(x,x_0)}\left(\frac{1}{\tau}\bar{u}^T \nabla f(x_0) + \frac{1}{\tau}\bar{v}_I^T \nabla g_I(x_0)\right) + \frac{1}{\tau}\bar{u}^T \frac{\rho^1 d^1(x, x_0)}{\alpha^1(x, x_0)}$$
$$+\frac{1}{\tau}\bar{v}_I^T \frac{\rho_I^2 d_I^2(x, x_0)}{\alpha_I^2(x, x_0)} < 0,$$

where $\tau = \sum\limits_{i=1}^{l} \bar{u}_i + \sum\limits_{j \in I} \bar{v}_j$. This implies

$$\sum_{i=1}^{l} \bar{u}_i \rho_i^1 \frac{d_i^1(x, x_0)}{\alpha_i^1(x, x_0)} + \sum_{j \in I} \bar{v}_j \rho_j^2 \frac{d_j^2(x, x_0)}{\alpha_j^2(x, x_0)} < 0,$$

which contradicts (7).

In Theorem 1, we require that $\bar{u} > 0$. In order to relax this condition, we need to enforce other convexity conditions on (f, g_I).

Theorem 3. *Assume that there exist a feasible solution x_0 for (MOP) and vectors $\bar{u} \in \mathbb{R}^p$ and $\bar{v} \in \mathbb{R}^q$ such that*

$$\bar{u}^T \nabla f(x_0) + \bar{v}^T \nabla g(x_0) = 0, \tag{8}$$

$$\bar{v}^T g(x_0) = 0, \tag{9}$$

$$\bar{u} \geqslant 0, \bar{v} \geqq 0. \tag{10}$$

If $(f,\ g_I)$ is weak strictly pseudoquasi $(C,\ \alpha,\ \rho,\ d)$-type I at x_0 with

$$\sum_{i=1}^{p} \bar{u}_i \rho_i^1 \frac{d_i^1(x, x_0)}{\alpha_i^1(x, x_0)} + \sum_{j \in I} \bar{v}_j \rho_j^2 \frac{d_j^2(x, x_0)}{\alpha_j^2(x, x_0)} \geqslant 0,$$

then x_0 is an efficient solution for (MOP).

Theorem 4. *Assume that there exist a feasible solution x_0 for (MOP) and vectors $\bar{u} \in \mathbb{R}^p$ and $\bar{v} \in \mathbb{R}^q$ such that the triplet $(x_0, \bar{u}, \bar{v})$ satisfies (8), (9) and (10). If $(f,\ g_I)$ is pseudoquasi $(C,\ \alpha,\ \rho,\ d)$-type I at x_0 with*

$$\sum_{i=1}^{p} \bar{u}_i \rho_i^1 \frac{d_i^1(x, x_0)}{\alpha_i^1(x, x_0)} + \sum_{j \in I} \bar{v}_j \rho_j^2 \frac{d_j^2(x, x_0)}{\alpha_j^2(x, x_0)} \geqslant 0,$$

then x_0 is a weak efficient solution for (MOP).

Remark 1. Proofs of Theorems 3 and 4 are similar to the one of Theorem 2.

Therefore, we can derive similar theorems to the above varying the convexity assumptions on (f, g). These results also can be found in [YCLP05].

4 Duality

In this section, we give some weak and strong duality relations between (MOP) and its dual problems. In 1989, Egudo [Egu89] formulated a Mond-Weir type dual program of a multiobjective programming problem.

$$\text{(MOD)} \quad \max f(y) = (f_1(y), f_2(y), \ldots, f_p(y))^T$$
$$\text{s.t.} \sum_{i=1}^{p} u_i \nabla f_i(y) + \sum_{j=1}^{q} v_j \nabla g_j(y) = 0,$$
$$v^T g(y) \geq 0, v \in \mathbb{R}_+^q,$$
$$\sum_{i=1}^{p} u_i = 1,\ u_i > 0\ (i = 1, 2, \ldots, p),\ y \in X \subseteq \mathbb{R}^n,$$

where f_i, g_j are differentiable functions defined in X.
Next, we present weak and strong duality relations between (MOP) and (MOD).

Theorem 5 (`Weak Duality`). *Let x_0 be a feasible solution of (MOP) and $(y_0, \bar{u}, \bar{v})$ be a feasible solution of (MOD), and let any of the following hold:*

(a) (f, g) is (C, α, ρ, d)-type I at y_0, and

$$\sum_{i=1}^{p} \bar{u}_i \rho_i^1 \frac{d_i^1(x_0, y_0)}{\alpha_i^1(x_0, y_0)} + \sum_{j=1}^{q} \bar{v}_j \rho_j^2 \frac{d_j^2(x_0, y_0)}{\alpha_j^2(x_0, y_0)} \geqslant 0, \tag{11}$$

(b) $(f, v^T g)$ is strong pseudoquasi (C, α, ρ, d)-type I at y_0, and (11) holds.
(c) $(u^T f, v^T g)$ is pseudoquasi (C, α, ρ, d)-type I at y_0, $f_i (i = 1, \cdots, p)$, and

$$\rho^1 \frac{d^1(x_0, y_0)}{\alpha^1(x_0, y_0)} + \rho^2 \frac{d^2(x_0, y_0)}{\alpha^2(x_0, y_0)} \geqslant 0.$$

Then the following cannot hold.

$$f(x_0) \leqslant f(y_0).$$

Proof. Suppose that $f(x_0) \leqslant f(y_0)$. By the hypothesis (a) and convexity of $C_{(x_0, y_0)}$, it follows that

$$\sum_{i=1}^{l} \frac{\bar{\lambda}_i}{\bar{\tau}} \frac{f_i(x_0) - f_i(y_0)}{\alpha_i^1(x_0, y_0)} + \sum_{j=1}^{q} \frac{\bar{\mu}_j}{\bar{\tau}} \frac{g_j(x_0) - g_j(y_0)}{\alpha_j^2(x_0, y_0)} \geqslant \sum_{i=1}^{l} \frac{\bar{\lambda}_i}{\bar{\tau}} \frac{f_i(x_0) - f_i(y_0)}{\alpha_i^1(x_0, y_0)}$$
$$+ \sum_{j=1}^{q} \frac{\bar{\mu}_j}{\bar{\tau}} \frac{-g_j(y_0)}{\alpha_j^2(x_0, y_0)} \geqslant C_{(x_0, y_0)} \left(\frac{1}{\bar{\tau}} (\bar{\lambda}^T \nabla \bar{f}(y_0) + \bar{\mu}^T \nabla \bar{g}(y_0)) \right)$$
$$+ \frac{1}{\bar{\tau}} \left(\sum_{i=1}^{l} \bar{\lambda}_i \rho_i^1 \frac{d_i^1(x_0, y_0)}{\alpha_i^1(x_0, y_0)} + \sum_{j=1}^{q} \bar{\mu}_j \rho_j^2 \frac{d_j^2(x_0, y_0)}{\alpha_i^2(x_0, y_0)} \right),$$

where $\bar{\tau} = 1 + \sum_{j=1}^{m} \bar{\mu}_j$. Since $\bar{\mu}^T g(y_0) \geqslant 0$, we have

$$\sum_{i=1}^{l} \frac{\bar{\lambda}_i}{\bar{\tau}} \frac{f_i(x_0) - f_i(y_0)}{\alpha_i^1(x_0, y_0)} + \sum_{j=1}^{q} \frac{\bar{\mu}_j}{\bar{\tau}} \frac{g_j(x_0) - g_j(y_0)}{\alpha_j^2(x_0, y_0)} < 0.$$

From the last two inequalities and the construction of (MOD), we can conclude that

$$\sum_{i=1}^{l} \bar{\lambda}_i \rho_i^1 \frac{d_i^1(x_0, y_0)}{\alpha_i^1(x_0, y_0)} + \sum_{j=1}^{q} \bar{\mu}_j \rho_j^2 \frac{d_j^2(x_0, y_0)}{\alpha_j^2(x_0, y_0)} < 0,$$

which contradicts the assumption of the theorem. Parts (b) and (c) can be proved similarly.

The following strong duality result can easily be derived from the previous weak duality theorem.

Theorem 6 (Strong Duality). *Assume that a feasible solution $x_0 \in S$ satisfies a generalized constraint qualification [Mae94]. Then there exist $\bar{u} \in \mathbb{R}^p$, $\bar{v} \in \mathbb{R}^q$ such that $(x_0, \bar{u}, \bar{v})$ is a feasible solution of (MOD) and $\bar{v}_j g_j(x_0) = 0$ $(j = 1, 2, \ldots, q)$. Furthermore if the assumptions in the weak duality are satisfied, then $(x_0, \bar{u}, \bar{v})$ is an efficient solution of (MOD).*

Some other types of dual programs were also introduced in the literature including Wolfe dual program [LHP03, AH00, AH01, HA04], generalized Mond-Weir dual program [AH00, YCLP05] and Mixed type dual program [YCLP05, HA04].

5 Multiobjective Fractional Programming

Let us consider the multiobjective fractional programming problem:

$$\begin{aligned}
\text{(MFP)} \min \ & \frac{f(x)}{g(x)} = \left(\frac{f_1(x)}{g_1(x)}, \frac{f_2(x)}{g_2(x)}, \ldots, \frac{f_p(x)}{g_p(x)}\right) \\
s.t. \ & h(x) \leqq 0, \\
& x \in X,
\end{aligned} \tag{12}$$

where X is an open subset of $\mathbb{R}^n$, and $f_i(\cdot)$, $g_i(\cdot)$ $(i = 1, 2, \ldots, p)$ and $h_j(\cdot)$ $(j = 1, 2, \cdots, q)$ are real-valued differentiable functions defined on X. We also assume that $f_i(x) \geqslant 0$, $g_i(x) > 0$ for each $x \in X$.

Recently, Chinchuluun et al. [CYP05] studied the multiobjective fractional programming problem with (C, α, ρ, d)-convexity.

One of the properties of (C, α, ρ, d)-convex functions is given by the following theorem.

Lemma 1. *Let $X \subset \mathbb{R}^N$ be an open set. Suppose that $\theta(x)$ and $\vartheta(x)$ are real-valued differentiable functions on X, and $\theta(x) \geqslant 0$, $\vartheta(x) > 0$ for all $x \in X$. If θ and $-\vartheta$ are (C, α, ρ, d)-convex and regular at $x_0 \in X$, then $\frac{\theta}{\vartheta}$ is $(\bar{C}, \bar{\alpha}, \bar{\rho}, \bar{d})$-convex at x_0, where $\bar{\alpha}(x, x_0) = \frac{\vartheta(x_0)\alpha(x,x_0)}{\vartheta(x)}$, $\bar{\rho} = \rho\frac{\theta(x_0)+\vartheta(x_0)}{\vartheta(x_0)}$, $\bar{d}(x, x_0) = \frac{d(x,x_0)}{\vartheta(x)}$ and $\bar{C}_{(x,x_0)}\left(\nabla\frac{\theta}{\vartheta}(x_0)\right) = \frac{\theta(x_0)+\vartheta(x_0)}{\vartheta^2(x_0)} \cdot C_{(x,x_0)}\left(\frac{\vartheta^2(x_0)}{\theta(x_0)+\vartheta(x_0)}\nabla\frac{\theta}{\vartheta}(x_0)\right)$.*

Proof. We can write the following simple equality for any $x \in X$.

$$\frac{\theta(x)}{\vartheta(x)} - \frac{\theta(x_0)}{\vartheta(x_0)} = \frac{\theta(x) - \theta(x_0)}{\vartheta(x)} - \frac{\theta(x_0)(\vartheta(x) - \vartheta(x_0))}{\vartheta(x)\vartheta(x_0)}.$$

By the definition of (C, α, ρ, d)-convexity, and the fact that $\theta \geqslant 0$, $\theta_2 > 0$, the above equation can be rewritten as follows.

$$\begin{aligned}
\frac{1}{\alpha(x, x_0)}\left(\frac{\theta(x)}{\vartheta(x)} - \frac{\theta(x_0)}{\vartheta(x_0)}\right) \geqslant & \frac{1}{\vartheta(x)}\left(C_{(x,x_0)}(\nabla\theta(x_0)) + \rho\frac{d(x, x_0)}{\alpha(x, x_0)}\right) \\
& + \frac{\theta(x_0)}{\vartheta(x)\vartheta(x_0)}\left(C_{(x,x_0)}(-\nabla\vartheta(x_0)) + \rho\frac{d(x, x_0)}{\alpha(x, x_0)}\right).
\end{aligned}$$

Using the convexity of $C_{(x,x_0)}$, it follows that

$$\frac{1}{\alpha(x,x_0)}\left(\frac{\theta(x)}{\vartheta(x)}-\frac{\theta(x_0)}{\vartheta(x_0)}\right) \geqslant \frac{\theta(x_0)+\vartheta(x_0)}{\vartheta(x)\vartheta(x_0)}\left(C_{(x,x_0)}\left(\frac{\vartheta^2(x_0)}{\theta(x_0)+\vartheta(x_0)}\cdot\left(\frac{\vartheta(x_0)\nabla\theta(x_0)}{\vartheta^2(x_0)}-\frac{\theta(x_0)\nabla\vartheta(x_0)}{\vartheta^2(x_0)}\right)\right)\right) + \rho\frac{\theta(x_0)+\vartheta(x_0)}{\vartheta(x)\vartheta(x_0)}\frac{d(x,x_0)}{\alpha(x,\, x_0)}.$$

Denote

$$\bar{\alpha}(x,x_0)=\frac{\vartheta(x_0)\alpha(x,x_0)}{\vartheta(x)},\bar{\rho}=\rho\frac{\theta(x_0)+\vartheta(x_0)}{\vartheta(x_0)},\bar{d}(x,x_0)=\frac{d(x,x_0)}{\vartheta(x)}$$

and

$$\bar{C}_{(x,x_0)}\left(\nabla\frac{\theta}{\vartheta}(x_0)\right)=\frac{\theta(x_0)+\vartheta(x_0)}{\vartheta^2(x_0)}C_{(x,x_0)}\left(\frac{\vartheta^2(x_0)}{\theta(x_0)+\vartheta(x_0)}\nabla\frac{\theta}{\vartheta}(x_0)\right).$$

It is easy to prove that $\bar{C}$ is convex with respect to variable $\nabla\frac{\theta}{\vartheta}(x_0)$. Therefore $\frac{\theta}{\vartheta}$ is $(\bar{C},\bar{\alpha},\bar{\rho},\bar{d})$-convex at x_0.

Based on Lemma 1, the following theorem can be derived using the similar argument as in Theorem 2.

Theorem 7. *Let x_0 be a feasible solution of Problem (MFP). Suppose that there exist $u=(u_1,u_2,\ldots,u_p)\in\mathbb{R}^p_+$, $u_i>0$ $(i=1,2,\ldots,p)$, $\sum\limits_{i=1}^{p}u_i=1$, and $v=(v_1,v_2,\ldots,v_q)\in\mathbb{R}^q_+$ such that*

$$\sum_{i=1}^{p}u_i\nabla\left(\frac{f_i}{g_i}\right)(x_0)+\sum_{j=1}^{q}v_j\nabla h_j(x_0)=0,$$
$$v_jh_j(x_0)=0,\ j=1,\ldots,q.$$

If f_i, $-g_i$ $(i=1,2,\ldots,p)$ are (C,α_i,ρ_i,d_i)-convex, h_j $(j=1,2,\ldots,q)$ are $(C,\beta_j,\kappa_j,\delta_j)$-convex at x_0, and

$$\sum_{i=1}^{p}u_i\bar{\rho}_i\frac{\bar{d}_i(x,x_0)}{\bar{\alpha}_i(x,x_0)}+\sum_{j=1}^{q}v_j\kappa_j\frac{\delta_j(x,x_0)}{\beta_j(x,x_0)}\geq 0$$

where $\bar{\alpha}_i(x,x_0)=\frac{g_i(x_0)\alpha_i(x,x_0)}{g_i(x)}$, $\bar{\rho}_i=\rho_i\frac{f_i(x_0)+g_i(x_0)}{g_i(x_0)}$, $\bar{d}_i(x,x_0)=\frac{d_i(x,x_0)}{g_i(x)}$, $i=1,2,\ldots,p$. Then x_0 is an efficient solution of Problem (MFP).

The Mond-Weir Dual of the problem (MFP) can be written in the form

$$\begin{aligned}
\text{(MFD)} \max\ & \frac{f(y)}{g(y)} = \left(\frac{f_1(y)}{g_1(y)}, \frac{f_2(y)}{g_2(y)}, \dots, \frac{f_p(y)}{g_p(y)}\right)^T \\
\text{s.t.}\ & \sum_{i=1}^{p} u_i \nabla \left(\frac{f_i}{g_i}\right)(y) + \sum_{j=1}^{q} v_j \nabla h_j(y) = 0, \\
& v^T h(y) \geq 0, \\
& \sum_{i=1}^{p} u_i = 1,\ u_i > 0\ (i = 1, 2, \dots, p),\ u \in \mathbb{R}^p_+, \\
& v \in \mathbb{R}^q_+,\ y \in X.
\end{aligned}$$

The following duality results, in nondifferentiable case, can also be obtained in [CYP05].

Theorem 8 (Weak Duality). *Let x_0 be a feasible solution of Problem (MFP) and $(y_0, \bar{u}, \bar{v})$ be a feasible solution of (MFD). If f_i, $-g_i$ $(i = 1, 2, \dots, p)$ are $(C, \alpha_i, \rho_i, d_i)$-convex at y_0, h_j $(j = 1, 2, \dots, q)$ are $(C, \beta, \kappa_j, \delta_j)$-convex at y_0, and*

$$\sum_{i=1}^{p} \bar{u}_i \bar{\rho}_i \frac{\bar{d}_i(x_0, y_0)}{\bar{\alpha}_i(x_0, y_0)} + \sum_{j=1}^{q} \bar{v}_j \kappa_j \frac{\delta_j(x_0, y_0)}{\beta(x_0, y_0)} \geq 0$$

where $\bar{\alpha}_i(x_0, y_0) = \frac{g_i(y_0)\alpha_i(x_0, y_0)}{g_i(x_0)}$, $\bar{\rho}_i = \rho_i \frac{f_i(y_0) + g_i(y_0)}{g_i(y_0)}$ and $\bar{d}_i(x_0, y_0) = \frac{d_i(x_0, y_0)}{g_i(x_0)}$, $i = 1, 2, \dots, p$. Then the following cannot hold.

$$\frac{f(x_0)}{g(x_0)} \leqslant \frac{f(y_0)}{g(y_0)}.$$

Theorem 9 (Strong Duality). *Let x_0 be an efficient solution of (MFP). Suppose that x_0 satisfies a generalized constraint qualification [Mae94]. Then there exists $(x_0, \bar{u}, \bar{v})$ which is a feasible solution for (MFD). Then the objective function values of (MFP) and (MFD) at the corresponding points are equal.*

6 Multiobjective Variational Programming

In this section, we formulate a multiobjective variational programming problem and derive some optimality and duality results involving generalized convexities. Several authors have been interested in optimality conditions and duality relations for multiobjective variational programming problems [MM94, NN97, MR00, KK02, BS03, KK05, AG05]. Following [MR00, KK02, AG05], we use the following notations. Let $I = [a, b]$ be a real interval and $f : I \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^p$ be a continuously differentiable function. In order to consider $f(t, x, \dot{x})$, where $x : I \to \mathbb{R}^n$ with derivative $\dot{x}$, denote the partial derivative of f with respect to t, x, and $\dot{x}$, respectively, by f_t, f_x, and $f_{\dot{x}}$, such that

$$f_x = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right], \quad f_{\dot{x}} = \left[\frac{\partial f}{\partial \dot{x}_1}, \ldots, \frac{\partial f}{\partial \dot{x}_n}\right].$$

Let $C(I, \mathbb{R}^n)$ denote the space of piecewise smooth functions x with norm $\|x\| = \|x\|_\infty + \|Dx\|_\infty$, where the differentiation operation D is given by

$$u = Dx \quad \Leftrightarrow \quad x(t) = t_0 + \int_a^t u(s)ds,$$

in which t_0 is given boundary value. Therefore, $D = \frac{d}{dt}$ except at discontinuities.

We now consider the following multiobjective continuous programming problem:

$$\begin{aligned} \text{(VP)} \quad & \min \int_a^b f(t,x,\dot{x})dt = \left(\int_a^b f_1(t,x,\dot{x})dt, \ldots, \int_a^b f_p(t,x,\dot{x})dt\right) \\ & \text{s.t. } x(a) = t_0, \ x(b) = t_f, \\ & \quad g(t,x,\dot{x}) \leqq 0, \ t \in I, \end{aligned}$$

where $f_i : I \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$, $g_j : I \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, q$, are continuously differentiable functions.

Let us denote the set of feasible set of (VP) by S, that is,

$$S := \{x \in C(I, \mathbb{R}^n) | x(a) = t_0, x(b) = t_f, g(t,x,\dot{x}) \leqq 0\}.$$

In order to prove the strong duality theorem we will invoke the following lemma due to Chankong and Haimes [CH83].

Lemma 2. *A point $x_0 \in S$ is an efficient solution for (VP) if and only if x_0 solves , $\forall k = 1, \ldots, p$,*

$$\begin{aligned} P_k(x_0) \quad & \min \int_a^b f_k(t,x,\dot{x})dt \\ & s.t. \ x(a) = t_0, \ x(b) = t_f, \\ & \quad g(t,x,\dot{x}) \leqq 0, \\ & \quad \int_a^b f_j(t,x,\dot{x})dt \leqq \int_a^b f_j(t,x_0,\dot{x}_0)dt, \ \forall j \neq k. \end{aligned}$$

Nahak and Nanda [NN97] considered the multiobjective variational problem under (F, ρ)-convexity on the functions involved, and formulated Mond-Weir and Wolfe type dual programs for the problem. Recently, mixed type dual for the problem was introduced and the corresponding duality theorems were derived under (F, ρ)-convexity in [AG05]. Kim and Kim [KK02, KK05] studied the multiobjective variational problem involving generalized convexity called V-type I invex functions. This generalized convexity is based on the generalized convexity, called $V-$invexity, by Jeyakumar and Mond [JM92].

We now study the problem based on (C, ρ, d)-convexity. Let us first redefine (C, ρ, d)-convexity for the multiobjective variational program.

Definition 11. *A function $\theta(t, x, \dot{x}) : I \times X \times X \to \mathbb{R}$ is said to be (C, ρ, d)-convex at $x_0 \in X$ if, $\forall\ x \in X$, the inequality*

$$\begin{aligned}
&\int_a^b \theta(t, x, \dot{x})dt - \int_a^b \theta(t, x_0, \dot{x}_0)dt \\
&\quad \geq \int_a^b C_{(t,x,x_0,\dot{x},\dot{x}_0)}\left(\theta_x(t, x_0, \dot{x}_0) - \frac{d}{dt}\theta_{\dot{x}}(t, x_0, \dot{x}_0)\right) dt \\
&\quad + \rho \int_a^b d(t, x, x_0)dt,
\end{aligned}$$

where $C : I \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is a convex function with respect to the last variable, holds. The function θ is said to be (C, ρ, d)-convex over X if, $\forall\ x_0 \in X$, it is (C, ρ, d)-convex at x_0. In particular, θ is said to be strongly (C, ρ, d)- convex according to $\rho > 0$.

We also assume that the convex function $C_{(t,x,x_0,\dot{x},\dot{x}_0)} : \mathbb{R}^n \to \mathbb{R}$ satisfies $C_{(x,x_0,\dot{x},\dot{x}_0)}(0) = 0$ for any $(t, x, x_0, \dot{x}, \dot{x}_0) \in I \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$.

Definition 12. *We say the problem (VP) is of (C, ρ, d)-type I invex at $x_0 \in S$ if, for all $x \in S$, we have*

$$\begin{aligned}
&\int_a^b f(t, x, \dot{x})dt - \int_a^b f(t, x_0, \dot{x}_0)dt \\
&\quad \geqq \int_a^b C_{(t,x,x_0,\dot{x},\dot{x}_0)}\left(f_x(t, x_0, \dot{x}_0) - \frac{d}{dt}f_{\dot{x}}(t, x_0, \dot{x}_0)\right) dt \\
&\quad + \rho^1 \int_a^b d^1(t, x, x_0)dt \qquad (13)
\end{aligned}$$

and

$$-\int_a^b g(t,x_0,\dot{x}_0)dt$$
$$\geqq \int_a^b C_{(t,x,x_0,\dot{x},\dot{x}_0)}\left(g_x(t,x_0,\dot{x}_0)-\frac{d}{dt}g_x(t,x_0,\dot{x}_0)\right)dt$$
$$+\rho^2\int_a^b d^2(t,x,x_0)dt.$$

Definition 13. *We say that the problem (VP) is of pseudoquasi (C,ρ,d)-type I invex at $x_0\in S$ with respect to u and λ if, for all $x\in S$, there exist some vectors $u\in\mathbb{R}^p$, $u\geqslant 0$, and a piecewise smooth function $\lambda: I\to\mathbb{R}^q$, the implications*

$$\int_a^b\sum_{i=1}^p u_i f_i(t,x,\dot{x})dt<\int_a^b\sum_{i=1}^p u_i f_i(x_0,\dot{x}_0)dt$$
$$\Rightarrow \int_a^b\sum_{i=1}^p u_i C_{(t,x,x_0,\dot{x},\dot{x}_0)}\left(f_{i_x}(t,x_0,\dot{x}_0)-\frac{d}{dt}f_{i\dot{x}}(t,x_0,\dot{x}_0)\right)dt$$
$$+\sum_{i=1}^p u_i\rho_i^1\int_a^b d_i^1(t,x,x_0)dt<0$$

and

$$-\int_a^b\sum_{j=1}^q \lambda_j(t)g_j(t,x_0,\dot{x}_0)dt\leq 0$$
$$\Rightarrow \int_a^b\sum_{j=1}^q \lambda_j(t)C_{(t,x,x_0,\dot{x},\dot{x}_0)}\left(g_{j_x}(t,x_0,\dot{x}_0)-\frac{d}{dt}g_{j\dot{x}}(t,x_0,\dot{x}_0)\right)dt$$
$$+\sum_{j=1}^q \lambda_j(t)\rho_j^2\int_a^b d_j^2(t,x,x_0)dt\leq 0.$$

Let us state the following sufficient optimality conditions for (VP) with pseudoquasi (C,ρ,d)-type I invex functions.

Theorem 10 (`Sufficient Optimality`). *Let $x_0\in S$. Suppose that there exist $\bar{u}\in\mathbb{R}^p$, $\bar{u}>0$, and a piecewise smooth function $\bar{\lambda}: I\to\mathbb{R}^q$, $\bar{\lambda}(t)\geqslant 0$ such that*

$$\sum_{i=1}^{p} \bar{u}_i \left(f_x^i(t, x_0, \dot{x}_0) - \frac{d}{dt} f_{\dot{x}}^i(t, x_0, \dot{x}_0) \right) + \sum_{j=1}^{q} \bar{\lambda}_j(t) \left(g_x^j(t, x_0, \dot{x}_0) - \frac{d}{dt} j_{\dot{x}}^j(t, x_0, \dot{x}_0) \right) = 0, \tag{14}$$

$$\int_a^b \sum_{j=1}^{q} \bar{\lambda}_j(t) g_j(t, x_0, \dot{x}_0) dt = 0, \tag{15}$$

$$\sum_{i=1}^{p} u_i \rho_i^1 \int_a^b d_i^1(t, x, x_0) dt + \sum_{j=1}^{q} \lambda_j(t) \rho_j^2 \int_a^b d_j^2(t, x, x_0) dt \geq 0. \tag{16}$$

If the problem (VP) is pseudoquasi (C, ρ, d)-type I invex at x_0 with respect to $\bar{u}$, $\bar{\lambda}$, then x_0 is an efficient solution for (VP).

Proof. Suppose to the contrary that x_0 is not an efficient solution for (VP). Then, there exists a feasible solution $x \in S$ such that

$$\int_a^b f(t, x, \dot{x}) dt \leqslant \int_a^b f(t, x_0, \dot{x}_0) dt,$$

which implies that

$$\int_a^b \sum_{i=1}^{p} \bar{u}_i f_i(t, x, \dot{x}) dt < \int_a^b \sum_{i=1}^{p} \bar{u}_i f_i(x_0, \dot{x}_0) dt.$$

By the hypothesis, we have

$$\int_a^b \sum_{i=1}^{p} \bar{u}_i C_{(t,x,x_0,\dot{x},\dot{x}_0)} \left(f_x^i(t, x_0, \dot{x}_0) - \frac{d}{dt} f_{\dot{x}}^i(t, x_0, \dot{x}_0) \right) dt + \sum_{i=1}^{p} u_i \rho_i^1 \int_a^b d_i^1(t, x, x_0) dt < 0. \tag{17}$$

On the other hand, we can write (15) as

$$-\int_a^b \sum_{j=1}^{q} \bar{\lambda}_j(t) g_j(t, x_0, \dot{x}_0) dt \leq 0.$$

Then it follows that

$$\int_a^b \sum_{j=1}^q \bar{\lambda}_j(t) C_{(t,x,x_0,\dot{x},\dot{x}_0)} \left(g_x^j(t,x_0,\dot{x}_0) - \frac{d}{dt} g_{\dot{x}}^j(t,x_0,\dot{x}_0) \right) dt$$
$$+ \sum_{j=1}^q \lambda_j(t) \rho_j^2 \int_a^b d_j^2(t,x,x_0) dt \leq 0. \tag{18}$$

We now, adding (17) and (18) together and applying the convexity assumption of C, can have

$$\int_a^b C_{(t,x,x_0,\dot{x},\dot{x}_0)} \left(\sum_{i=1}^p \frac{\bar{u}_i}{\tau} \left(f_x^i(t,x_0,\dot{x}_0) - \frac{d}{dt} f_{\dot{x}}^i(t,x_0,\dot{x}_0) \right) \right.$$
$$\left. + \sum_{j=1}^q \frac{\bar{\lambda}_j(t)}{\tau} \left(g_x^j(t,x_0,\dot{x}_0) - \frac{d}{dt} j_{\dot{x}}^j(t,x_0,\dot{x}_0) \right) \right) dt$$
$$+ \sum_{i=1}^p u_i \rho_i^1 \int_a^b d_i^1(t,x,x_0) dt + \sum_{j=1}^q \lambda_j(t) \rho_j^2 \int_a^b d_j^2(t,x,x_0) dt < 0,$$

where $\tau = \sum_{i=1}^p \bar{u}_i + \sum_{j=1}^q \bar{\lambda}_j(t)$. The last inequality conflicts the given conditions (14) and (16).

Following Kiam and Kim [KK05] and Mishra and Mukherjee [MM94], we consider the following Mond-Weir type dual problem to (VP).

$$\text{(MVD)} \quad \max \left(\int_a^b f_1(t,y,\dot{y}) dt, \ldots, \int_a^b \{ f_p(t,y,\dot{y}) dt \right)$$
$$\text{s.t. } y(a) = t_0, \; y(b) = t_f,$$
$$\sum_{i=1}^p u_i \left\{ f_y^i(t,y,\dot{y}) - \frac{d}{dt} f_{\dot{y}}^i(t,y,\dot{y}) \right\}$$
$$+ \sum_{j=1}^q \lambda_j(t) \left\{ g_y^j(t,y,\dot{y}) - \frac{d}{dt} g_{\dot{y}}^i(t,y,\dot{y}) \right\} = 0$$
$$\int_a^b \lambda_j(t) g_j(t,y,\dot{y}) dt \geqq 0, \; \forall j \in Q$$
$$u \in \mathbb{R}^p, \; u > 0, \; \lambda(t) \geqq 0, \; t \in I,$$

We now state weak and strong duality relations between (VP) and (MVD) without proofs since proofs are similar to that of Theorem 10.

Theorem 11 (Weak Duality). *Let x_0 and $(y_0, \bar{u}, \bar{\lambda})$ be feasible solutions for (VP) and (MVD) respectively. If the problem (VP) is pseudoquasi (C, ρ, d)-type I invex at y_0 with respect to $\bar{u}$, $\bar{\lambda}$ and the condition (16) holds, then the following cannot hold:*

$$\int_a^b f(t, x_0, \dot{x}_0)dt \leqslant \int_a^b f(t, y_0, \dot{y}_0).$$

Theorem 12 (Strong Duality). *Let x_0 be an efficient solution for (VP). We also assume that, for all $k \in P$, a constraint qualification for $P_k(x_0)$ at x_0 is satisfied. Then there exist $\bar{u} > 0$ and piecewise smooth function $\bar{\lambda} : I \to \mathbb{R}^q$, $\bar{\lambda} \geqq 0$ such that $(x_0, \bar{u}, \bar{\lambda})$ is feasible for (MVD). Further if the assumptions of weak duality theorem are satisfied, then $(x_0, \bar{u}, \bar{\lambda})$ is an efficient for (MVD).*

7 Conclusions

In this chapter, we have discussed optimality conditions and duality results for different multiobjective optimization problems under a generalized convexity so called (C, α, ρ, d) convexity. We first established several sufficient optimality conditions for the general multiobjective programming problem. We have also presented a Mond-Weir type dual of the program and presented weak and strong duality theorems. It has been shown that the ratio of two (C, α, ρ, d) convex functions is also (C, α, ρ, d) convex with different characteristics. Based on this result, sufficient optimality conditions and duality theorems for fractional multiobjective programming problems have been presented. In the last section, we have formulated a multiobjective variational programming problem and shown its optimality conditions and duality results based on the generalized convexity.

References

[AG05] Ahmad, I., Gulati, T.R.: Mixed type duality for multiobjective variational problems with generalized (F, ρ)-convexity. Journal of Mathematical Analysis and Applications, **306**, 669–683 (2005)

[AH00] Aghezzaf, B., Hachimi, M.: Generalized invexity and duality in multiobjective programming problems. Journal of Global Optimization, **18**, 91–101 (2000)

[AH01] Aghezzaf, B., Hachimi, M.: Sufficiency and duality in multiobjective programming involving generalized (F,ρ)-convexity. Journal of Mathematical Analysis and Applications, **258**, 617–628 (2001)

[Ber95] Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont, Mass. (1995)

[BS03] Bhatia, D., Sharma, A.: Duality with BF-type I functions for a class of nondifferentiable multiobjective variational problems. Journal of Mathematical Analysis and Applications, **287**, 415–429 (2003)

[CH83] Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making: Theory and Methodology. North-Holland, New York (1983)

[CP05] Chinchuluun, A., Pardalos, P.M.: Recent developments in multiobjective optimization. Annals of Operations Research, to appear (2005)

[CYP05] Chinchuluun, A., Yuan, D.H., Pardalos, P.M.: Optimality conditions and duality for nondifferentiable multiobjective fractional programming with genreralized convexity. Annals of Operations Research, to appear (2005)

[Egu89] Egudo, R.: Efficiency and generalized convex duality for multiobjective programs. Journal of Mathematical Analysis and Applications, **138**, 84–94 (1989)

[HA04] Hachimi, M., Aghezzaf, B.: Sufficiency and duality in differentiable multiobjective programming involving generalized type I functions. Journal of Mathematical Analysis and Applications, **296**, 382–392 (2004)

[Han81] Hanson, M.A.: On sufficient of the Kuhn-Tucker conditions. Journal of Mathematical Analysis and Applications, **80**, 545–550 (1981)

[HM82] Hanson, M.A., Mond, B.: Further generalizations of convexity in mathematical programming. Journal of Information and Optimization Sciences, **3**, 25–32 (1982)

[HM87] Hanson, M.A., Mond, B.: Necessary and sufficient conditions in constrained optimization. Mathematical Programming, **37**, 51–58 (1987)

[JM92] Jeyakumar, V., Mond, B.: On generalized convex mathematical programming. Journal of the Australian Mathematical Society Series B, **34**, 43–53 (1992)

[KK02] Kim, D.S., Kim, A.L.: Optimality and duality for nondiferentiable multiobjective variational problems. Journal of Mathematical Analysis and Applications, **274**, 255–278 (2002)

[KK05] Kim, D.S., Kim, M.H.: Generalized type I invexity and duality in multiobjective variational problems. Journal of Mathematical Analysis and Applications, **307**, 533–554 (2005)

[LHP01] Liang, Z.A., Huang, H.X., Pardalos, P.M.: Optimality conditions and duality for a class of nonlinear fractional programming problems. Journal of Optimization Theory and Applications, **110**, 611–619 (2001)

[LHP03] Liang, Z.A., Huang, H.X., Pardalos, P.M.: Efficiency conditions and duality for a class of multiobjective fractional programming problems. Journal of Global Optimization, **27**, 447–471 (2003)

[Mae94] Maeda, T.: Constraint qualification in multiobjective problems: differentiable case. Journal of Optimization Theory and Applications, **80**, 483–500 (1994)

[Mie99] Miettinen, K.M.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)

[MM94] Mishra, S.K., Mukherjee, R.N.: On efficiency and duality for multiobjective variational problems. Journal of Mathematical Analysis and Applications, **187**, 40–54 (1994)

[MR00] Mukherjee, R.N., Rao, Ch.P.: Mixed type duality for multiobjective variational problems. Journal of Mathematical Analysis and Applications, **252**, 571–586 (2000)

[NN97] Nahak, C., Nanda, S.: On efficiency and duality for multiobjective variational control problems with $(F-\rho)$-convexity. Journal of Mathematical Analysis and Applications, **209**, 415–434 (1997)

[Pre92] Preda, V.: On sufficiency and duality for multiobjective programs. Journal of Mathematical Analysis and Applications, **166**, 365–377 (1992)

[PSZ95] Pardalos, P.M., Siskos, Y., Zopounidis, C.: Advances in Multicriteria Analysis. Kluwer Academic Publishers, Dordrecht (1995)

[Roc70] Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton, NJ (1970)

[Via82] Vial, J.P.: Strong convexity of sets and functions. Journal of Mathematical Economics, **9**, 187–205 (1982)

[Via83] Vial, J.P.: Strong and weak convexity of sets and functions. Mathematics of Operations Research, **8**, 187–205 (1983)

[YCLP05] Yuan, D.H., Chinchuluun, A., Liu, X.L., Pardalos, P.M.: Optimality conditions and duality for multiobjective programming involving (C, α, ρ, d) type-I functions. In: Proceedings of 8th International Symposium on Generalized Convexity Generalized Monotonicity, submitted (2005)

[YLCP05] Yuan, D.H., Liu, X.L., Chinchuluun, A., Pardalos, P.M.: Nondifferentiable minimax fractional programming problems. Journal of Optimization Theory and Applications, to appear (2005)

Towards Optimal Techniques for Solving Global Optimization Problems: Symmetry-Based Approach

Christodoulos A. Floudas[1] and Vladik Kreinovich[2]

[1] Department of Chemical Engineering, Princeton University, Princeton, NJ 08544, USA floudas@titan.princeton.edu
[2] Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA vladik@utep.edu

1 Introduction

1.1 Global Optimization – an Important Practical Problem

In many practical situations, we have several possible actions, and we must choose the best action. For example, we must find the best design of an object, or the best control of a plant. The set of possible actions is usually characterized by parameters $x = (x_1, \ldots, x_n)$, and the result of different actions (controls) is characterized by an *objective function* $f(x)$.

In some cases, the objective function describes losses or expenses; in such cases, the problem of finding the best action (design, or control) can be described as the problem of *global minimization*, i.e., the problem of finding the values x for which the function $f(x)$ attains the smallest possible value.

In other cases, the objective function describes gain; in such cases, the problem of finding the best action can be described as the problem of *global maximization*, i.e., the problem of finding the values x for which the function $f(x)$ attains the largest possible value.

Global minimization and global maximization are particular cases of *global optimization.*

Similar problems arise in *data processing*, when we have a model characterized by several parameters x_i, and we need to find the values of these parameters which provide the best fit for the data, i.e., for which the discrepancy $f(x)$ between the data and the model is the smallest possible.

Actual and potential real-world applications of global optimization are overviewed, e.g., in [Pin96].

1.2 Global Optimization is a Difficult Computational Problem

In general, the problem of finding the exact values x that minimize a given objective function $f(x)$ is computationally difficult (NP-hard); see, e.g., [Vav91].

Crudely speaking, NP-hardness means that (provided that P$\neq$NP) it is not possible to have an algorithm that solves *all* optimization problems in reasonable time. In other words, no matter how good is an algorithm for solving global optimization optimization problems, there will always be cases in which better results are possible.

1.3 Variety of Global Optimization Techniques

Since we cannot hope for a single algorithm for global optimization, new algorithms are constantly designed, and the existing algorithms are constantly modified. As a result, we have a wide variety of different global optimization techniques and methods; see, e.g., [HP95].

There exist classes of objective functions for which efficient algorithms for global optimization are possible. It is therefore natural to try to reduce general hard-to-solve global optimization problems to problems from such classes.

One class for which global optimization is easier-to-solve is the class of quadratic objective functions. Namely, it is known that a global optimum of an objective function $f(x)$ is attained at a point x at which all the partial derivatives of this function are equal to 0. For a quadratic function $f(x)$, we can thus find the desired optimum by solving a system of linear equations $\dfrac{\partial f}{\partial x_i} = 0$. It is therefore natural to find a minimum of $f(x)$ by approximating a function $f(x)$ with a linear or quadratic expression – i.e., in effect, by consider gradient descent-type techniques and/or their second-order analogues.

Another important class is the class of convex functions – for which there are efficient algorithms for finding the global minimum. Not surprisingly, there are numerous effective global optimization techniques that reduce the general global optimization problems to convex ones; see, e.g., [Flo00, TS02].

In many real-life situations, the objective function is complex, and it is difficult to approximate it by a quadratic and/or by a convex objective function on its entire domain. In such situations, it is reasonable to subdivide the original domain into smaller subdomains and approximate $f(x)$ by different functions on different subdomains; see, e.g., [Kea96].

There also exist numerous heuristic and semi-heuristic techniques which emulate the way optimization is done in nature: e.g., genetic algorithms simulate the biological evolution which, in general, leads to the birth and survival individuals and species which are best fit for a given environment; see, e.g., [Mih96].

1.4 Problem: Which Techniques is the Best?

We have already mentioned that there is a wide variety of different global optimization techniques. Because of this variety, every time we have a new optimization problem, we must select the best technique for solving this problem.

This selection problem is made even more complex by the fact that most techniques for solving global optimization problems have *parameters* that need to be adjusted to the problem or to the class of problems. For example, in gradient methods, we can select different step sizes.

When we have a *single* parameter (or few parameters) to choose, it is possible to empirically try many values and come up with an (almost) optimal value. Thus, in such situations, we can come up with optimal version of the corresponding technique.

In other approaches, e.g., in methods like convex underestimators (described in detail in the next section), instead of selecting the value of single *number*-valued parameter, we have select the auxiliary *function*. It is not practically possible to test all possible functions, so it is not easy to come up with an optimal version of the corresponding technique.

1.5 What We Do in This Chapter

In this chapter, we consider the problem of selecting the best auxiliary function within a given global optimization technique. Specifically, we show that in many such selection situations, natural symmetry requirements enable us either to analytically solve the problem of finding the optimal auxiliary function, or at least reduce this problem to the easier-to-solve problem of finding a few parameters.

In particular, for convex understimators, we show that we can thus explain both the αBB method [AAF98, ADAF98, Flo00, MF94] and its modifications recently proposed in [AF04, AF06].

2 Case Study: Selecting Convex Underestimators

2.1 Why Convex Underestimators?

It is well known that convex functions are computationally easier to minimize than non-convex ones; see, e.g., [Flo00]. This relative easiness is not only an empirical fact, it also has a theoretical justification; see, e.g., [KK05, Vav91].

Because of this relative easiness, one of the approaches to minimization of a non-convex function $f(x) = f(x_1, \ldots, x_n)$ (under certain constraints) over a box $[x^L, x^U] = [x_1^L, x_1^U] \times \ldots \times [x_n^L, x_n^U]$ is to first minimize its convex "underestimator", i.e., a convex function $L(x) \leq f(x)$.

- Since the new function $L(x)$ is convex, it is easy to minimize;

- since $L(x)$ is an underestimator, i.e., $L(x) \le f(x)$, the minimum of $L(x)$ is a lower bound for the minimum of $f(x)$.

By selecting $L(x)$ as close to $f(x)$ as possible, we can get estimates for $\min f(x)$ which are as close to the actual minimum as possible.

The quality of approximation improves when the boxes become smaller. So, to get more accurate bounds on $\min f(x)$, we can:

- bisect the box $[x^L, x^U]$ into sub-boxes,
- use the above technique to estimate $\min f$ over each sub-box, and
- return the smallest of these estimates as the lower bound for $\min f$ over the entire box $[x^L, x^U]$.

2.2 Example: αBB Techniques

A known efficient approach to designing a convex underestimator is the αBB global optimization algorithm [AAF98, ADAF98, Flo00, MF94], in which we select an underestimator $L(x) = f(x) + \Phi(x)$, where

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot (x_i - x_i^L) \cdot (x_i^U - x_i). \tag{1}$$

Here, the parameters α_i are selected in such a way that the resulting function $L(x)$ is convex and still not too far away from the original objective function $f(x)$.

2.3 Natural Generalization of αBB Techniques

In many optimization problems, αBB techniques are very efficient, but in some non-convex optimization problems, it is desirable to improve their performance. One way to do that is to provide a more general class of methods, with more parameters to tune.

In the αBB techniques, for each coordinate x_i, we have a single parameter α_i affecting this coordinate. Changing α_i is equivalent to a linear re-scaling of x_i. Indeed, if we change the unit for measuring x_i to a new unit which is λ_i times smaller, then all the numerical values become λ_i times larger: $x_i \to y_i = g_i(x_i)$, where $g_i(x_i) = \lambda_i \cdot x_i$. In principle, we can have two different re-scalings:

- $x_i \to y_i = g_i(x_i) = \lambda_i \cdot x_i$ on the interval $[x_i^L, x_i]$, and
- $x_i \to z_i = h_i(x_i) = \mu_i \cdot x_i$ on the interval $[x_i, x_i^U]$.

If we substitute the new values $y_i = g_i(x_i)$ and $z_i = h_i(x_i)$ into the formula (1), then we get the following expression

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot (g_i(x_i) - g_i(x_i^L)) \cdot (h_i(x_i^U) - h_i(x_i)). \tag{2}$$

For the above linear re-scalings, we get

$$\widetilde{\Phi}(x) = -\sum_{i=1}^{n} \widetilde{\alpha}_i \cdot (x_i - x_i^L) \cdot (x_i^U - x_i),$$

where $\widetilde{\alpha}_i = \alpha_i \cdot \lambda_i \cdot \mu_i$.

From this viewpoint, a natural generalization is to replace *linear* re-scalings $g_i(x_i)$ and $h_i(x_i)$ with *non-linear* ones, i.e., to consider convex underestimators of the type $L(x) = f(x) + \Phi(x)$, where $\Phi(x)$ is described by the formula (2) with non-linear functions $g_i(x_i)$ and $h_i(x_i)$. Now, instead of selecting a number α_i for each coordinate i, we have an additional freedom of choosing arbitrary non-linear functions $g_i(x_i)$ and $h_i(x_i)$. Which are the best choices?

2.4 Empirical Fact: Exponential Functions $g_i(x_i)$ and $h_i(x_i)$ Are the Best

In [AF04, AF06], several different non-linear functions have been tried, and it turned out that among the tested functions, the best results were achieved for the exponential functions $g_i(x_i) = \exp(\gamma_i \cdot x_i)$ and $h_i(x_i) = -\exp(-\gamma_i \cdot x_i)$. For these functions, the expression (2) can be somewhat simplified: indeed,

$$\alpha_i \cdot (g_i(x_i) - g_i(x_i^L)) \cdot (h_i(x_i^U) - h_i(x_i)) =$$

$$\alpha_i \cdot (e^{\gamma_i \cdot x_i} - e^{\gamma_i \cdot x_i^L}) \cdot (-e^{-\gamma_i \cdot x_i^U} + e^{-\gamma_i \cdot x_i}) =$$

$$\widetilde{\alpha}_i \cdot (1 - e^{\gamma_i \cdot (x_i - x_i^L)}) \cdot (1 - e^{\gamma_i \cdot (x_i^U - x_i)}),$$

where $\widetilde{\alpha}_i \overset{\text{def}}{=} \alpha_i \cdot e^{\gamma_i \cdot (x_i^U - x_i^L)}$.

2.5 Questions

Two related questions naturally arise:

- first, a *practical* question: an empirical choice is made by using only finitely many functions; is this choice indeed the best – or there are other, even better functions $g_i(x_i)$ and $h_i(x_i)$, which we did not discover because we did not try them?
- second, a *theoretical* question: how can we explain the above empricial fact?

2.6 Natural Idea of Symmetry: Intuitive Motivations for Shift-Invariance

The starting (0) point for measuring each coordinate x_i is often a matter of arbitrary choice; e.g.:

- Fahrenheit and Celsius scales use different starting points for measuring temperature,
- different calendars use different starting points as Year 0,

etc.

If a selection of the functions $g_i(x_i)$ and $h_i(x_i)$ is "optimal" (in some intuitive sense), then the results of using these optimal functions should not change if we simply change the starting point for measuring x_i – i.e., replace each value x_i with a new value $x_i + s$, where s is the shift in the starting point. Indeed, otherwise, if the "quality" of the resulting convex underestimators changes with shift, we could apply a shift and get better functions $g_i(x_i)$ and $h_i(x_i)$ – which contradicts to our assumption that the selection of $g_i(x_i)$ and $h_i(x_i)$ is already optimal.

So, the "optimal" choices $g_i(x_i)$ and $g_i(x_i)$ can be determined from the requirement that each component $\alpha_i \cdot (g_i(x_i) - g_i(x_i^L)) \cdot (h_i(x_i^U) - h_i(x_i))$ in the sum (2) be invariant under the corresponding shift. Let us describe this requirement in precise terms.

Definition 1. *A pair of smooth functions $(g(x), h(x)))$ from real numbers to real numbers is* shift-invariant *if for every s and α, there exists $\widetilde{\alpha}(\alpha, s)$ such that for every x^L, x, and x^U, we have*

$$\alpha \cdot (g(x) - g(x^L)) \cdot (h(x^U) - h(x)) =$$

$$\widetilde{\alpha}(\alpha, s) \cdot (g(x+s) - g(x^L + s)) \cdot (h(x^U + s) - h(x+s)). \tag{3}$$

Comment. Smoothness is needed because smooth functions are easier to optimize, and we therefore want our techniques to preserve smoothness.

2.7 Consequences of Shift-Invariance

At first glance, shift invariance is a reasonable but weak property. It turns out, however, that this seemingly weak property actually almost uniquely determines the optimal selection of exponential functions:

Proposition 1. *If a pair of functions $(g(x), h(x))$ is shift-invariant, then this pair is either exponential or linear, i.e., each of the functions $g(x)$ and $h(x)$ has the form $g(x) = A + C \cdot \exp(\gamma \cdot x)$ or $g(x) = A + k \cdot x$.*

Comments.

- For reader's convenience, all the proofs are placed in a separate (last) section.
- One can easily see that adding a constant to each of the functions $g(x)$ and $h(x)$ does not change the expression (2), so we can safely assume that each of these functions has the form $g(x) = \exp(\gamma \cdot x)$ and $h(x) = x$.

2.8 Additional Symmetry $x \to -x$ and the Final Result

In addition to shift, another natural symmetry is changing the sign: e.g., for electric charge, the fact that electrons are negatively charged is just a matter of definition; we can as well consider them positively charged. If we require that the expression (2) remain invariant if we change the sign, i.e., replace x by $-x$, then we get the relation between $g(x)$ and $h(x)$: $h(x) = -g(-x)$. So, if a pair $(g(x), h(x)$ is shift-invariant and sign-invariant, then:

- either $g(x) = \exp(\gamma \cdot x)$ and $h(x) = -\exp(-\gamma \cdot x)$,
- or $g(x) = h(x) = x$.

In other words, *the optimal generalized αBB scheme is either the original αBB, or the scheme with exponential functions* described in [AF04, AF06]. Thus, we have answers to both above questions:

- yes, the exponential functions are indeed optimal, and
- yes, we have a theoretical explanation of why they are optimal – because they are the only pair of functions which satisfies the condition of symmetry (shift-invariance and sign-invariance) that optimal pairs should satisfy.

2.9 Auxiliary Result: Scale Invariance

In addition to changing the starting point for x, we can also (as we have mentioned) change a unit for measuring x, i.e., consider *scaling* transformations $x \to \lambda \cdot x$. Shall we require scale-invariance as well? In other words, shall we require that the expression (2) be invariant not only w.r.t. shifts but w.r.t scalings as well?

We already know that there are only two shift-invariant solutions: exponential and linear functions. Out of these two solutions, only the linear solution – corresponding to αBB – is scale-invariant. Thus, if we also require scale-invariance, we restrict ourselves only to αBB techniques – and miss on (often better) exponential generalizations.

Since we cannot require *both* shift- and scale-invariance, a natural next question is: what if we only require scale invariance?

Definition 2. *A pair of smooth functions $(g(x), h(x))$ from real numbers to real numbers is* scale-invariant *if for every λ and α, there exists $\widetilde{\alpha}(\alpha, \lambda)$ such that for every x^L, x, and x^U, we have*

$$\alpha \cdot (g(x) - g(x^L)) \cdot (h(x^U) - h(x)) =$$
$$\widetilde{\alpha}(\alpha, \lambda) \cdot (g(\lambda \cdot x) - g(\lambda \cdot x^L)) \cdot (h(\lambda \cdot x^U) - h(\lambda \cdot x)). \tag{4}$$

Proposition 2. *If a pair of functions $(g(x), h(x))$ is scale-invariant, then this pair is either exponential or linear, i.e., each of the functions $g(x)$ and $h(x)$ has the form $g(x) = A \cdot x^\gamma$ or $g(x) = A + k \cdot \ln(x)$.*

From the theoretical viewpoint, these functions may look as good as the exponential functions coming from shift invariance, and in practice, they do not work so well.

The problem with these solutions is that, as we have mentioned, we want to preserve smoothness. Both linear and exponential functions which come from shift-invariance are infinitely differentiable for all x and hence, adding the corresponding term $\Phi(x)$ will not decrease the smoothness level of the objective function.

In contrast, in general, the functions $g(x) = x^\gamma$ which come from scale invariance are not infinitely differentiable at $x = 0$. They are differentiable only for integer values γ. So, if we use scale invariance to select a convex underestimator, we end up with a new parameter γ which only attains integer-valued values and is, thus, less flexible than the continuous-valued parameters coming from scale-invariance.

2.10 Auxiliary Shift-Invariance Results

Instead of an expression (2), we can consider an even more general expression

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot a_i(a, x^L) \cdot b_i(x_i, x_i^U). \tag{5}$$

Whet can we conclude from shift-invariance in this more general case?

Definition 3. *A pair of smooth functions $(a(x, x^L), b(x, x^U))$ from real numbers to real numbers is* shift-invariant *if for every s and α, there exists $\widetilde{\alpha}(\alpha, s)$ such that for every x^L, x, and x^U, we have*

$$\alpha \cdot a(x, x^L) \cdot b(x, x^U) =$$

$$\widetilde{\alpha}(\alpha, s) \cdot a(x + s, x^L + s) \cdot b(x + s, x^U + s). \tag{6}$$

Proposition 3. *If a pair of functions $(a(x, x^L), b(x, x^U))$ is shift-invariant, then*

$$a(x, x^L) \cdot b(x, x^U) = A(x - x^L) \cdot B(x^U - x) \cdot e^{\gamma \cdot x^L}$$

for some functions $A(x)$ and $B(x)$ and for some real number γ.

Comment. If we additionally require that the expression $a(x, x^L) \cdot b(x, x^U)$ be invariant under $x \to -x$, then we conclude that $B(x) = A(x)$.

Another shift-invariance result comes from the following observation. Both αBB expression $-(x - x^L) \cdot (x^U - x)$ and the generalized expression

$$-(1 - e^{\gamma \cdot (x - x^L)}) \cdot (1 - e^{\gamma \cdot (x^U - x)})$$

have the form $a(x - x^L) \cdot a(x^U - x)$ with $a(0) = 0$. The differences $x - x^L$ and $x^U - x$ come from the fact that we want these expressions to be shift-invariant.

The product form makes sense, since we want the product to be 0 on each border $x = x^L$ and $x = x^U$ of the corresponding interval $[x^L, x^U]$.

On the other hand, it is well known that optimizing a product is more difficult than optimizing a sum; since we will be minimizing the expression $f(x) + \Phi(x)$, it is therefore desirable to be able to reformulate it in terms of the easier-to-minimize sum, e.g., as $b(x - x^L) + b(x^U - x) + c(x^U - x^L)$ for some functions b and c (for minimization purposes, c does not depend on x and is thus a constant). It is worth mentioning that both the αBB expression and its exponential generalization allow such representation:

- from the known equality $a \cdot b = \frac{1}{2}((a+b)^2 - a^2 - b^2)$, we conclude that
$$-(x - x^L) \cdot (x^U - x) = \frac{1}{2} \cdot (x - x^L)^2 + \frac{1}{2} \cdot (x^U - x)^2 - \frac{1}{2} \cdot (x^U - x^L)^2;$$
- for the exponential function, simply multiplying the two sums leads to the desired expression:
$$-(1 - e^{\gamma \cdot (x - x^L)}) \cdot (1 - e^{\gamma \cdot (x^U - x)}) = -1 + e^{\gamma \cdot (x - x^L)} + e^{\gamma \cdot (x^U - x)} - e^{\gamma \cdot (x^U - x^L)}.$$

Interestingly, the above two expressions are the only one which have this easiness-to-compute property:

Definition 4. *We say that a smooth function $a(x)$ from real numbers to real numbers describes an* easy-to-compute *underestimator if $a(0) = 0$, $a'(0) \neq 0$, and there exist smooth functions $b(x)$ and $c(x)$ such that for every x, x^L, and x^U, we have*

$$a(x - x^L) \cdot a(x^U - x) = b(x - x^L) + b(x^U - x) + c(x^U - x^L). \qquad (7)$$

Comment. The condition $a'(0) \neq 0$ comes from the fact that otherwise, for small $\Delta x \overset{\text{def}}{=} x - x^L$ and $x^U - x$, each value $a(x - x^L)$ will be quadratic in $x - x^L$, the resulting product will be fourth order, and we will not be able to compensate for quadratic non-convex terms in the original objective function $f(x)$ – which defeats the purpose of using $f(x) + \Phi(x)$ as a *convex* underestimator.

Proposition 4. *The only functions which describe easy-to-compute underestimators are $a(x) = k \cdot x$ and $a(x) = k \cdot (1 - e^{\gamma \cdot x})$.*

Comment. This is already a second shift-invariance related results which selects linear and exponential functions as "the best" in some reasonable sense. In the following section, we show that this is not an accident: namely, we will prove that *any* "natural" shift-invariant optimality cruetrion on the set of all possible underestimator methods selects either a linear or an exponential function.

3 Selecting Convex Underestimatiors: From Informally "Optimal" to Formally Optimal Selections

3.1 In the Previous Section, We Used Informal "Optimality"

In the above text, we argued that if a selection is optimal (in some reasonable sense), than it is natural to expect that this selection should be shift-invariant. We used this argument to justify the empirical selection of convex underestimators.

In this section, we will go one step further, and explain that the empirical selection is indeed optimal – in the precise mathematical sense of this word.

3.2 What Are We Selecting?

In effect, we are selecting the functions $g(x)$ and $h(x)$. However, as we have mentioned earlier, what we are really interested in is the corresponding family of functions

$$\Phi(x) = -\alpha \cdot (g(x) - g(x^L)) \cdot (h(x^U) - h(x)).$$

The difference is that (as we have mentioned) we can change one (or both) of the functions $g(x)$ and $h(x)$ and still end up with the same class of functions. For example, if we replace the original function $g(x)$ with a new function $\widetilde{g}(x) = A \cdot g(x) + B$, then we end up with the same class of functions $\Phi(x)$. With this in mind, let us introduce the following definition.

Definition 5. *By a* family, *we mean the family of functions*

$$F = \{-\alpha \cdot (g(x) - g(x^L)) \cdot (h(x^U) - h(x))\}, \tag{8}$$

where $g(x)$ and $h(x)$ are fixed, and α goes over all real numbers.

Denotation. We will denote a family generated by functions $g(x)$ and $h(x)$ by $F(g,h)$.

In these terms, the question is how to select, out of all possible families, the family which is optimal in some reasonable sense, i.e., which is optimal in the sense of some optimality criterion.

3.3 What is an Optimality Criterion?

When we say that some *optimality criterion* is given, we mean that, given two different families F and F', we can decide whether the first or the second one is better, or whether these families are equivalent w.r.t. the given criterion. In mathematical terms, this means that we have a *pre-ordering relation* $\preceq$ on the set of all possible families.

3.4 We Want to Solve an Ambitious Problem: Enumerate All Families that are Optimal Relative to Some Natural Criteria

One way to approach the problem of choosing the "best" family F is to select *one* optimality criterion, and to find a family that is the best with respect to this criterion. The main drawback of this approach is that there can be different optimality criteria, and they can lead to different optimal solutions. It is, therefore, desirable not only to describe a family that is optimal relative to some criterion, but to describe *all* families that can be optimal relative to different natural criteria[3]. In this section, we are planning to implement exactly this more ambitious task.

3.5 Examples of Optimality Criteria

Pre-ordering is the general formulation of optimization problems in general, not only of the problem of choosing a family F. In general optimization theory, in which we are comparing arbitrary *alternatives* a', a'', ..., from a given set A, the most frequent case of such a pre-ordering is when a *numerical criterion* is used, i.e., when a function $J : A \to R$ is given for which $a' \preceq a''$ iff $J(a') \leq J(a'')$.

Several natural numerical criteria can be proposed for choosing a function J. For example, we can take, as a criterion, the *average* number of iterations that lead to determining all global minima with a given relative accuracy (average in the sense of some natural probability measure on the set of all problems).

Alternatively, we can fix a class of problems, and take the largest number of iterations for problems of this class as the desired (numerical) optimality criterion.

Many other criteria of this type can be (and have actually been) proposed. For such "worst-case" optimality criteria, it often happens that there are several different alternatives that perform equally well in the worst case, but whose performance differ drastically in the average cases. In this case, it makes sense, among all the alternatives with the optimal worst-case behavior, to choose the one for which the average behavior is the best possible. This very natural idea leads to the optimality criterion that is not described by one numerical optimality criterion $J(a)$: in this case, we need *two* functions: $J_1(a)$ describes the worst-case behavior, $J_2(a)$ describes the average-case behavior, and $a \preceq b$ iff either $J_1(a) < J_1(b)$, or $J_1(a) = J_1(b)$ and $J_2(a) \leq J_2(b)$.

We could further specify the described optimality criterion and end up with *one* natural criterion. However, as we have already mentioned, the goal

[3] In this phrase, the word "natural" is used informally. We basically want to say that from the purely mathematical viewpoint, there can be weird ("unnatural") optimality criteria. In our text, we will only consider criteria that satisfy some requirements that we would, from the common sense viewpoint, consider reasonable and natural.

of this chapter is not to find *one* family that is optimal relative to some criterion, but to describe *all* families that are optimal relative to some natural optimality criteria. In view of this goal, in the following text, we will not specify the criterion, but, vice versa, we will describe a very general class of *natural* optimality criteria.

So, let us formulate what "natural" means.

3.6 What Optimality Criteria are Natural?

We have already mentioned that the value x often represents the value of some measured quantity, and that the numerical value of a measured quantity changes if we select a new starting point. It is natural to require that the relative quality of two families does not depend on the choice of the starting point.

How does replacing a starting point change the family F? If we replace a starting point by a new one that is smaller by a constant s, then the quantity that was initially described by a value x will be described by a new value $x + s$. Correspondingly, x^L is replaced by $x^L + s$, and x^U by $x^U + s$. Thus, after this shift T_s, the original family (8) turns into the new family

$$T_s(F) \stackrel{\text{def}}{=} \{-\alpha \cdot (g(x+s) - g(x^L+s)) \cdot (h(x^U+s) - h(x+s))\}. \quad (9)$$

In these terms, the above requirement is that if F is better than F', then the "shifted" F (i.e., the family $T_s(F)$) should be better than the "shifted" F' (i.e., than $T_s(F')$).

There is one more reasonable requirement for a criterion, that is related with the following idea: If the criterion does not select a single optimal family, i.e., if it considers several different families equally good, then we can always use some other criterion to help select between these "equally good" ones, thus designing a two-step criterion. If this new criterion still does not select a unique family, we can continue this process until we arrive at a combination multi-step criterion for which there is only one optimal family. Therefore, we can always assume that our criterion is *final* in this sense.

Definition 6. *By an* optimality criterion, *we mean a* pre-ordering *(i.e., a transitive reflexive relation) $\preceq$ on the set A of all possible families. An optimality criterion $\preceq$ is called:*

- *shift-invariant if for all F, F', and s, $F \preceq F'$ implies $T_s(F) \preceq T_s(F')$.*
- *final if there exists one and only one family F that is preferable to all the others, i.e., for which $F' \preceq F$ for all $F' \neq F$.*

Proposition 5.

- *If a family F is optimal w.r.t. some shift-invariant final optimality criterion, then this family F is generated by linear or exponential functions $g(x)$ and $h(x)$.*

- *For every two exponential or linear functions $g(x)$ and $h(x)$, there exists a shift-invariant final optimality criterion for which the only optimal family is $F(g,h)$.*

Comments.

- In other words, if the optimality criterion satisfies the above-described natural properties, then *the optimal convex underestimator is generated by linear or exponential functions.*
- If, in addition to shift-invariance, we also require sign-invariance, then we conclude that either both functions $g(x)$ and $h(x)$ are linear (as in αBB), or both are exponential (as in the empirically best generalization of αBB).

4 Other Cases when a Symmetry-Based Approach Leads to Optimal Techniques for Solving Global Optimization Problems

Similar symmetry-based ideas have been applied to produce an optimal auxiliary function in other aspects of global optimization. Let us overview the main results obtained by following this direction.

4.1 Optimal Bisection

As we have mentioned, applying the optimization technique to the original function (or its convex underestimator) on the original box $[x^L, x^U]$ is not always the best strategy. One way to improve the optimization algorithm is to subdivide (e.g., bisect) the box into several sub-boxes and apply optimization techniques to these sub-boxes. Some of these sub-boxes must be further subdivided, etc. Two natural questions arise:

- which box should we select for bisection?
- which variable shall we use to bisect the selected box?

To answer both questions, several heuristic techniques have been proposed, and there has been an extensive empirical comparative analysis of these techniques. It turns out that for both questions, the symmetry-based approach enables us to theoretically justify the empirical selection:

- Until recently, for subdivision, a box B was selected for which the computed lower bound $\underline{f}(B)$ was the smallest possible. Recently (see, e.g, [CG98, CGC00]), it was shown that the optimization algorithms converge much faster if we select, instead, a box B with the largest possible value of the ratio

$$I_0 = \frac{\widetilde{f} - \underline{f}(B)}{\overline{f}(B) - \underline{f}(B)},$$

where $\widetilde{f}$ is a current upper bound on the actual global minimum. In [KC01], we give a symmetry-based theoretical justification for this empirical criterion. Namely, we condider all possible indictaor functions $I(\underline{f}(B), \overline{f}(B), \widetilde{f})$, and we show that:

- first, that the empirically best criterion I_0 is the only one that is *invariant* w.r.t. some reasonable symmetries – namely, shift and scaling; and
- second, that this criterion is *optimal* in some (symmetry-related) reasonable sense.

- We can bisect a given box in n different ways, depending on which of n sides we decided to halve. So, the natural question appears: which side should we cut? i.e., where to bisect a given box? Historically the first idea was to cut the *longest* side (for which $x_i^U - x_i^L \to \max$). It was shown (in [Rat92, Rat94]) that much better results are achieved if we choose a side i for which $|d_i|(x_i^U - x_i^L) \to \max$, where d_i is the known approximation for the partial derivative $\dfrac{\partial f}{\partial x_i}$. In [KK98], we consider arbitrary selection criteria, i.e., functions
$$S(f, d_1, \ldots, d_n, x_1^L, x_1^U, \ldots, x_n^L, x_n^U),$$
which map available information into an index $S \in \{1, 2, \ldots, n\}$, and we show that the empirically best box-splitting strategy is the only scale-invariant one – and is, thus, optimal under any scale-invariant final optimality criterion.

4.2 Optimal Selection of Penalty (Barrier) Functions

A similar approach can be used for reducing constraint optimization to non-constrained one. A well-known Lagrange multiplier method minimizes a function $f(x)$ under a constraint $g(x) = 0$ by reducing it to the unconstrained problem of optimizing a new objective function $f(x) + \lambda \cdot g(x)$. One of the known approaches to solving a similar problem with a constraint $g(x) \geq 0$ is the *penalty* (*barrier*) method in which we reduce the original problem to the un-constrained problem of optimizing a new objective function $f(x) + \lambda \cdot g(x) + \mu \cdot P(g(x))$, for an appropriate (non-linear) penalty function $P(y)$. Traditionally, the most widely used penalty functions are $P(y) = y \cdot \ln(y)$ and $P(y) = y^2$.

In [NK97], we show that the only y-scale-invariant families $\{\lambda \cdot y + \mu \cdot P(y)\}$ are families corresponding to $P(y) = y \cdot \ln(y)$ and $P(y) = y^\alpha$ for some real number α. Thus, under any scale-invariant optimality criterion, the optimal penalty function must indeed take one of these forms.

This example also shows that we can go beyond theoretical justification of empirically best heuristic, towards finding new optimal heuristics: indeed, for penalty functions, instead of single-parameter families $\{\lambda \cdot y + \lambda \cdot P(y)\}$, we can consider multiple-parameter families

$$\{\lambda \cdot y + \mu_1 \cdot P_1(y) + \ldots + \mu_m \cdot P_m(y)\}$$

for several functions $P_1(y), \ldots, P_m(y)$. In this case, the optimal functions have also been theoretically found: they are of the type

$$P_i(y) = L^{\alpha_i} \cdot (\ln(y))^{p_i}$$

for real (or complex) values α_i and non-negative integer values of p_i.

4.3 Other Examples

Similar symmetry-based techniques provide an explanation of several other empirically optimal techniques:

- sometimes, it is beneficial to (slightly) enlarge the original (non-degenerate) box $[x^L, x^U]$ and thus improve the performance of the algorithm; the empirically efficient "epsilon-inflation" technique [Rum80, Rum92]

  $$[x_i^L, x_i^U] \to [(1+\varepsilon)x_i^L - \varepsilon \cdot x_i^U, (1+\varepsilon)x_i^U - \varepsilon \cdot x_i^L]$$

  was proven to be the only shift- and scale-invariant technique and thus, the only one optimal under an arbitrary shift-invariant and scale-invariant optimality criterion [KSM97] (see also [Rum98]);
- by using shift-invariance, we explain why the probability proportional to $\exp(-\gamma \cdot f(x))$ is optimal in simulated annealing [NK97],
- by using scale- and shift-invariance, we explain why exponential and power re-scalings of the objective function are optimal in genetic algorithms [NK97];
- by using appropriate symmetries, we also explain, in [ISKS02], the empirically optimal selection of probabilities in swarm ("ant") optimization (see, e.g., [KES01]).

5 Proofs

5.1 Proof of Proposition 1

For $\alpha = 1$, the condition (3) takes the form

$$(g(x) - g(x^L)) \cdot (h(x^U) - h(x)) =$$

$$C(s) \cdot (g(x+s) - g(x^L + s)) \cdot (h(x^U + s) - h(x+s)), \tag{10}$$

where we denoted $C(s) \overset{\text{def}}{=} \tilde{\alpha}(1, s)$. To simplify this equation, let us separate the variables:

- let us move all terms containing x^L to the left-hand side – by dividing both sides by $(g(x+s) - g(x^L+s))$, and
- let us move all terms containing x^U to the right-hand side – by dividing both sides by $(h(x^U) - h(x))$.

As a result, we arrive at the following equation:

$$\frac{g(x) - g(x^L)}{g(x+s) - g(x^L+s)} = C(s) \cdot \frac{h(x^U+s) - h(x+s)}{h(x^U) - h(x)}. \tag{11}$$

Let us denote the left-hand side of this equation by A. By definition, the value A depends on x, s, and x^L. Since A is equal to the right-hand side, and the right-hand side does not depend on x^L, the expression A cannot depend on x^L, so $A = A(x, s)$, i.e.,

$$\frac{g(x) - g(x^L)}{g(x+s) - g(x^L+s)} = A(x, s). \tag{12}$$

Multiplying both sides by the denominator, we conclude that

$$g(x) - g(x^L) = A(x, s) \cdot (g(x+s) - g(x^L+s)). \tag{13}$$

Differentiating both sides by x^L, we conclude that

$$-g'(x^L) = -A(x, s) \cdot g'(x^L+s), \tag{14}$$

i.e., equivalently,

$$\frac{g'(x^L)}{g'(x^L+s)} = A(x, s). \tag{15}$$

In this equation, the left-hand side does not depend on x, so the right-hand does not depend on x either, i.e., $A(x, s) = A(s)$. Thus, (13) takes the form

$$a(s) \cdot (g(x) - g(x^L)) = (g(x+s) - g(x^L+s)), \tag{16}$$

where we denoted $a(s) \stackrel{\text{def}}{=} 1/A(s)$.

The function $g(x)$ is smooth, hence the function $a(s)$ is smooth too – as the ratio of two smooth functions. Differentiating both sides of (16) with respect to s and taking $s = 0$, we get

$$a \cdot (g(x) - g(x^L)) = (g'(x) - g'(x^L)), \tag{17}$$

where $a \stackrel{\text{def}}{=} a'(0)$.

To simplify this equation, let us separate the variables, i.e., let us move all the term depending on x to the right-hand side and all the terms depending on x^L to the left-hand side. As a result, we arrive at the following:

$$g'(x^L) - a \cdot g(x^L) = g'(x) - a \cdot g(x). \tag{18}$$

The right-hand side is a function of x only, but since it is equal to the left-hand side – which does not depend on x at all – it is simply a constant. If we denote this constant by b, we get the following equation:

$$g'(x) - a \cdot g(x) = b, \tag{19}$$

i.e.,

$$\frac{dg}{dx} = a \cdot g + b \tag{20}$$

and

$$\frac{dg}{a \cdot g + b} = dx. \tag{21}$$

When $a = 0$, integrating both sides of this equation, we get $\frac{1}{b} \cdot g(x) = x + C$, i.e., $g(x) = b \cdot x + b \cdot C$. When $a \neq 0$, then for $\widetilde{g}(x) \stackrel{\text{def}}{=} g(x) + \frac{b}{a}$, we get

$$\frac{d\widetilde{g}}{a \cdot \widetilde{g}} = dx, \tag{22}$$

hence $\frac{1}{a} \cdot \ln(\widetilde{g}(x)) = x + C$ thence $\ln(\widetilde{g}(x)) = a \cdot x + a \cdot C$, so $\widetilde{g}(x) = C \cdot \exp(a \cdot x)$ and $g(x) = \widetilde{g}(x) - \frac{b}{a} = C \cdot \exp(a \cdot x) + C_1$ for some constants C, a, and C_1. The proposition is proven.

5.2 Proof of Proposition 2

By introducing new variables $X = \ln(x)$, $X^L = \ln(x^L)$, and $X^U = \ln(x^U)$ – so that $x = \exp(X)$, $x^L = \exp(X^L)$, and $x^U = \exp(X^U)$, and by introducing new functions $G(X) = g(\exp(x))$ and $H(X) = h(\exp(x))$, one can easily check that if the pair $(g(x), h(x))$ is scale-invariant, then the new pair $(G(X), H(X))$ is shift-invariant.

We already know, from Proposition 1, how shift-invariant pairs look like: we have either $G(X) = A + C \cdot \exp(\gamma \cdot X)$ or $G(X) = A + k \cdot X$. From the definition of $G(X)$, we conclude that $g(x) = G(\ln(x))$; thus, we have either $g(x) = A + C \cdot \exp(\gamma \cdot \ln(x)) = A + C \cdot x^{\gamma}$ or $g(x) = A + k \cdot \ln(x)$. The proposition is proven.

5.3 Proof of Proposition 3

For $\alpha = 1$, the shift invariance requirement (6) takes the form

$$C(s) \cdot a(x + s, x^L + s) \cdot b(x + s, x^U + s) = a(x, x^L) \cdot b(x, x^U), \tag{23}$$

where $C(s) \stackrel{\text{def}}{=} \widetilde{\alpha}(1, s)$. Let us separate the variables by dividing both sides of this equation by $a(x, x^L)$ and $b(x, x^U)$; we then get

$$C(s) \cdot \frac{a(x+s, x^L+s)}{a(x, x^L)} = \frac{b(x+s, x^U+s)}{b(x, x^U)}. \tag{24}$$

The left-hand side ℓ of this equality depends only on x, x^L, and s. Since it is equal to the right-hand side, which does not depend on x^L at all, we can conclude that ℓ only depends on x and s:

$$C(s) \cdot \frac{a(x+s, x^L+s)}{a(x, x^L)} = \ell(x, s), \tag{25}$$

i.e., equivalently,

$$\frac{a(x+s, x^L+s)}{a(x, x^L)} = \widetilde{\ell}(x, s), \tag{26}$$

where $\widetilde{\ell}(x,s) \stackrel{\text{def}}{=} \dfrac{\ell(x,s)}{C(s)}$. For convenience (and without losing generality), we can describe $\widetilde{\ell}$ as depending on x and $x+s$:

$$\frac{a(x+s, x^L+s)}{a(x, x^L)} = N(x, x+s), \tag{27}$$

where $N(x,a) \stackrel{\text{def}}{=} \widetilde{\ell}(x, a-x)$.

We can perform the transition from x to $x+s$ in one step, as above, or we can first go to $x+(-x)=0$, and then to $0+(x+s)=x+s$. We then have

$$N(x, x+s) = \frac{a(x+s, x^L+s)}{a(x, x^L)} =$$
$$\frac{a(0+(x+s), (x^L-x)+(x+s))}{a(0, x^L-x)} \cdot \frac{a(x+(-x), x^L-x)}{a(x, x^L)} = \tag{28}$$
$$N(0, x+s) \cdot N(x, 0),$$

i.e.,

$$N(x, x+s) = N(0, x+s) \cdot N(x, 0). \tag{29}$$

For $s=0$, (27) leads to $N(x,x)=1$, hence from (29), we conclude that $N(0,x) \cdot N(x,0) = 1$ thence $N(x,0) = \dfrac{1}{N(0,x)}$; thus, (29) takes the form

$$N(x, x+s) = \frac{n(x+s)}{n(x)}, \tag{30}$$

where $n(x) \stackrel{\text{def}}{=} N(0,x)$. Substituting (30) into the formula (27), we conclude that

$$\frac{a(x+s, x^L+s)}{n(x+s)} = \frac{a(x, x^L)}{n(x)}. \tag{31}$$

In particular, for $s = -x^L$, we conclude that

$$\frac{a(x, x^L)}{n(x)} = \frac{a(x - X^L, 0)}{n(x - x^L)}, \tag{32}$$

i.e.,

$$a(x, x^L) = A_0(x - x^L) \cdot n(x), \tag{33}$$

where $A_0(z) \overset{\text{def}}{=} \dfrac{a(z,0)}{n(z)}$. Similarly, $b(x, x^U) = B_0(x^U - x) \cdot m(x)$ for some functions $B(z)$ and $m(x)$. Hence,

$$a(x, x^L) \cdot b(x, x^U) = A_0(x - x^L) \cdot B_0(x^U - x) \cdot p(x), \tag{34}$$

where $p(x) \overset{\text{def}}{=} m(x) \cdot n(x)$.

In this expression, the terms $A_0(x - x^L)$ and $B_0(x^U - x)$ are shift-invariant, so shift-invariance (23) of the product (34) means that $C(s) \cdot p(x + s) = p(x)$ for all x and s, i.e., that

$$p(x + s) = c(s) \cdot p(x), \tag{35}$$

where $c(s) \overset{\text{def}}{=} 1/C(s)$. Since the functions a and b are smooth, the functions p and c are smooth as well. Differentiating both sides of (35) w.r.t. s and substituting $s = 0$, we conclude that $p'(x) = \gamma \cdot p(x)$, where $\gamma \overset{\text{def}}{=} c'(0)$, hence $\dfrac{dp}{dx} = \gamma \cdot p$, $\dfrac{dp}{p} = \gamma \cdot dx$, and $\ln(p(x)) = \gamma \cdot x + C_1$; thus, $p(x) = C_2 \cdot \exp(\gamma \cdot x)$.

Since $\exp(\gamma \cdot x) = \exp(\gamma \cdot (x - x^L)) \cdot \exp(\gamma \cdot x^L)$, (34) takes the desired form

$$a(x, x^L) \cdot b(x, x^U) = A(x - x^L) \cdot B_0(x^U - x) \cdot e^{\gamma \cdot x^L}, \tag{36}$$

where $A(z) \overset{\text{def}}{=} A_0(z) \cdot C_2 \cdot \exp(\gamma \cdot z)$. The proposition is proven.

5.4 Proof of Proposition 4

For convenience, let us introduce new variables $X \overset{\text{def}}{=} x - x^L$ and $Y \overset{\text{def}}{=} x^U - x$. In terms of these variables, $x^U - x^L = X + Y$, and thus, the desired formula (7) takes the form

$$a(X) \cdot a(Y) = b(X) + b(Y) + c(X + Y). \tag{37}$$

Differentiating both sides of this equality w.r.t. Y, we conclude that

$$a(X) \cdot a'(Y) = b'(Y) + c'(X + Y). \tag{38}$$

Differentiating once again, this time w.r.t. X, we conclude that

$$a'(X) \cdot a'(Y) = c''(X + Y). \tag{39}$$

In particular, for $Y = 0$, we get

$$a'(X) \cdot a'(0) = c''(X). \tag{40}$$

Substituting this expression for $c''(X)$ into the formula (39), we conclude that

$$a'(X) \cdot a'(Y) = a'(X+Y) \cdot a'(0). \tag{41}$$

Dividing both sides by $a'(0)$, we get

$$\frac{a'(X)}{a'(0)} \cdot \frac{a'(Y)}{a'(0)} = \frac{a'(X+Y)}{a'(0)}, \tag{42}$$

i.e.,

$$A(X+Y) = A(X) \cdot A(Y), \tag{43}$$

where $A(X) \stackrel{\text{def}}{=} \dfrac{a'(X)}{a'(0)}$. Differentiating both sides of (43) by Y and substituting $Y = 0$, we conclude that $A'(X) = \gamma \cdot A(X)$, where $\gamma \stackrel{\text{def}}{=} A'(0)$. Similarly to the proof of Proposition 3, we get $A(X) = C_1 \cdot \exp(\gamma \cdot X)$ for some constant C_1. Therefore, $a'(X) = a'(0) \cdot A(X) = C_2 \cdot \exp(\gamma \cdot X)$, where $C_2 \stackrel{\text{def}}{=} a'(0) \cdot C_1$. Thus:

- If $\gamma = 0$, we get $a'(X) = C_2$, hence $a(X) = C_2 \cdot X + C_3$ for some constant C_3. From the condition $a(0) = 0$, we conclude that $C_3 = 0$.
- If $\gamma \neq 0$, then $a(X) = C_3 \cdot \exp(\gamma \cdot X) + C_4$, where $C_3 \stackrel{\text{def}}{=} \dfrac{C_2}{\gamma}$. Here too, from the condition that $a(0) = 0$, we conclude that $a(X) = C_4 \cdot (1 - \exp(\gamma \cdot X))$.

The proposition is proven.

5.5 Proof of Proposition 5

We have already shown, in the proof of Proposition 1, that:

- for linear or exponential functions, the corresponding family is shift-invariant, and
- vice versa, that if a family is shift-invariant, then it has the form $F(g, h)$ for some linear or exponential functions $g(x)$ and $h(x)$.

1°. To prove the first part of Proposition 5, we thus need to show that for every shift-invariant and final optimality criterion, the corresponding optimal family F_{opt} is shift-invariant, i.e., that $T_s(F_{\text{opt}}) = F_{\text{opt}}$ for all s. Then, the result will follow from Proposition 1.

Indeed, the transformation T_s is invertible, its inverse transformation is a shift by $-s$: $T_s^{-1} = T_{-s}$. Now, from the optimality of F_{opt}, we conclude that for every $F' \in A$, $T_s^{-1}(IF') \preceq F_{\text{opt}}$. From the invariance of the optimality criterion, we can now conclude that $F' \preceq T_s(F_{\text{opt}})$. This is true for all $F' \in A$ and therefore, the family $T(F_{\text{opt}})$ is optimal.

But since the criterion is final, there is only one optimal indicator function; hence, $T_s(F_{\text{opt}}) = F_{\text{opt}}$. So, the optimal family is indeed invariant and hence, due to Proposition 1, it coincides with $F(g, h)$ for some linear or exponential functions $g(x)$ and $h(x)$. The first part is proven.

2°. Let us now prove the second part of Proposition 5. Let $g(x)$ and $h(x)$ be fixed linear or exponential functions, and let $F_0 = F(g, h)$ be the corresponding family. We will then define the optimality criterion as follows: $F \preceq F'$ iff F' is equal to this F_0.

Since the family F_0 is shift-invariant, thus the defined optimality criterion is also shift-invariant. It is also clearly final.

The family F_0 is clearly optimal w.r.t. this shift-invariant and final optimality criterion. The proposition is proven.

Acknowledgement. This chapter was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Army Research Lab grant DATM-05-02-C-0046, Star Award from the University of Texas System, Texas Department of Transportation grant No. 0-5453, and the National Institute of Health grant R01 GM52032. The authors are greatly thankful to the editors and to the anonymous referees for valuable comments.

References

[AAF98] Adjiman, C.S., Androulakis, I., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLP II. Implementation and computational results. Computers and Chemical Engineering, **22**, 1159–1179 (1998)

[ADAF98] Adjiman, C.S., Dallwig, S., Androulakis, I., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLP I. Theoretical aspects. Computers and Chemical Engineering, **22**, 1137–1158 (1998)

[AF04] Akrotirianakis, I.G., Floudas, C.A.: Computational experience with a new class of convex underestimators: box-constrained NLP problems. Journal of Global Optimization, **29**, 249–264 (2004)

[AF06] Akrotirianakis, I.G., Floudas, C.A. A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. Journal of Global Optimization, to appear

[CG98] Casado, L.G., García, I.: New load balancing criterion for parallel interval global optimization algorithm, In: Proc. of the 16th IASTED International Conference, Garmisch-Partenkirchen, Germany, February 1998, 321–323 (1998)

[CGC00] Casado, L.G., García, I., Csendes, T.: A new multisection technique in interval methods for global optimization. Computing, **65**, 263–269 (2000)

[Flo00] Floudas, C.A.: Deterministic Global Optimization: Theory, Methods, and Applications. Kluwer, Dordrecht (2000)

[HP95] Horst, R., Pardalos, P.M. (eds): Handbook of Global Optimization. Kluwer, Dordrecht (1995)

[ISKS02] Iourinski, D., Starks, S.A., Kreinovich, V., Smith, S.F.: Swarm intelligence: theoretical proof that empirical techniques are optimal. In: Proceedings of the 2002 World Automation Congress WAC'2002, Orlando, Florida, June 9–13, 107–112 (2002)

[KC01] Kreinovich, V., Csendes, T.: Theoretical justification of a heuristic subbox selection criterion. Central European Journal of Operations Research, **9**, 255–265 (2001)

[Kea96] Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)

[KES01] Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Francisco, California (2001)

[KK98] Kearfott, R.B., Kreinovich, V.: Where to bisect a box? A theoretical explanation of the experimental results. In: Alefeld, G., Trejo, R.A. (eds) Interval Computations and its Applications to Reasoning Under Uncertainty, Knowledge Representation, and Control Theory. Proceedings of MEXICON'98, Workshop on Interval Computations, 4th World Congress on Expert Systems, México City, México (1998)

[KK05] Kearfott, R.B., Kreinovich, V.: Beyond convex? Global optimization is feasible only for convex objective functions: a theorem. Journal of Global Optimization, **33**, 617–624 (2005)

[KSM97] Kreinovich, V., Starks, S.A., Mayer, G.: On a theoretical justification of the choice of epsilon-inflation in PASCAL-XSC. Reliable Computing, **3**, 437–452 (1997)

[MF94] Maranas, C.D., Floudas, C.A.: Global minimal potential energy conformations for small molecules. Journal of Global Optimization, **4**, 135–170 (1994)

[Mih96] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin (1996)

[NK97] Nguyen, H.T., Kreinovich, V. Applications of Continuous Mathematics to Computer Science. Kluwer, Dordrecht (1997)

[Pin96] Pinter, J.D.: Global Optimization in Action. Kluwer, Dordrecht (1996)

[Rat92] Ratz, D.: Automatische Ergebnisverifikation bei globalen Optimierungsproblemen. Ph.D. dissertation, Universität Karlsruhe (1992)

[Rat94] Ratz, D.: Box-splitting strategies for the interval Gauss–Seidel step in a global optimization method. Computing, **53**, 337–354 (1994)

[Rum80] Rump, S.M.: Kleine Fehlerschranken bei Matrixproblemen. Ph.D. dissertation, Universität Karlsruhe (1980)

[Rum92] Rump, S.M.: On the solution of interval linear systems. Computing, **47**, 337–353 (1992)

[Rum98] Rump, S.M.: A note on epsilon-inflation. Reliable Computing, **4**(4), 371–375 (1998)

[TS02] Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer, Dordrecht (2002)

[Vav91] Vavasis, S.A.: Nonlinear Optimization: Complexity Issues. Oxford University Press, New York (1991)

Non-linear Global Optimization Using Interval Arithmetic and Constraint Propagation

Steffen Kjøller[1], Pavel Kozine[1], Kaj Madsen[1], and Ole Stauning[2]

[1] Informatics and Mathematical Modelling, Technical University of Denmark km@imm.dtu.dk
[2] Saxo Bank, Denmark ols@saxobank.com

1 Introduction

We consider the problem of finding the global minimum of a function $f : D \to \mathbb{R}$ where $D \subseteq \mathbb{R}^n$ is a compact right parallelepiped parallel to the coordinate axes:

$$x^* = \underset{x \in D}{\operatorname{argmin}} \; f(x). \tag{1}$$

In the following a compact right parallelepiped parallel to the coordinate axes is denoted a *box*.

Methods for solving global optimization problems have been investigated for many years, see for instance [Fle87], [HW04], [HPT00], [NW99], [TZ87]. For some classes of problems, e.g. analytically defined functions with a modest number of variables, interval methods have been very successful, [HW04].

In this chapter we describe a new branch-and-bound type method for solving (1). The method is an extension of the classical interval global optimization method (see for instance [HW04], [Moo76], [Ske74]), which is often denoted the *Moore-Skelboe algorithm*. This method iteratively investigates sub-boxes of D using monotonicity tests and interval Newton methods for reducing the set guaranteed to contain all solutions. The extension to be described uses constraint propagation (CP) in each iteration to further reduce this set, without losing solutions. Such a combination has previously been used by several authors, for instance [Kea03], [GBH01], [HMD97], [Mes04]. To the best of our knowledge we are the first, however, to apply CP for finding rigorous bounds for the set of stationary points, i.e., enclosing the solutions to the non-linear set of equations $f'(x) = 0$.

In the classical interval global optimization method such an inclusion is also applied, using some variation of the interval Newton method, [Moo66]. However the two inclusion methods CP and Newton are of quite different natures.

Under non-singularity conditions the classical (iterative) interval method for solving non-linear equations has a quadratic asymptotic convergence rate.

However the initial box $X_{(0)}$ for the iteration often has to be quite narrow. If a box X contains more than one solution then $f''(x)$ is singular for some $x \in X$, and then the interval Newton method cannot be applied.

The CP method for enclosing solutions to a set of non-linear equations is normally not so sensitive to narrow starting boxes and to singularities in f''. However its ultimate convergence rate is often slower than the interval Newton method. Therefore CP may be used to provide an initial reduction whereas the classical interval method provides the ultimate convergence.

We describe the two methods and their combination in Sect. 2. The implementation and two numerical illustrations are described in Sect. 3.

2 Description of the Method

The method is a combination of a version of the Moore-Skelboe algorithm for interval global optimization and a version of the constraint propagation method for solving a system of non-linear equations.

In Subsection 2.1 we describe the interval global optimization algorithm which is a branch-and-bound algorithm combined with the Krawczyk algorithm [Kra69] for solving the non-linear equation $f'(x) = 0$. In Subsection 2.2 the constraint propagation algorithm for finding bounds for the solutions to $f'(x) = 0$ is described, and in Subsection 2.3 this constraint propagation algorithm is incorporated into the Moore-Skelboe algorithm.

2.1 The Basic Interval Method

The algorithm is rigorous, i.e., it is guaranteed that all solutions are located. It is a branch and bound type method. At any stage of the algorithm we have the *candidate set* S. This is a finite set of sub-boxes $S_{(k)} \subseteq D$ having the property that the set of solutions to (1), X^*, is contained in the union of $\{S_{(k)}\}$. The aim is to reduce the candidate set, and to do that, let F be an interval extension of f (see [Moo66]), and notice that

$$\min_k\{L(S_{(k)})\} \;\le\; f^* \;\le\; \min_k\{f(x_k)\} \;\equiv\; \tau, \tag{2}$$

where $L(S_{(k)})$ is the lower bound of $F(S_{(k)})$ and x_k is a random point in $S_{(k)}$.

Therefore, if

$$L(S_{(k)}) > \tau, \tag{3}$$

then $S_{(k)}$ can be discarded from the candidate set. Otherwise, we can get sharper bounds by splitting $S_{(k)}$ into two smaller subregions. If $n = 1$ then the splitting is done by simple bisection, and in multiple dimensions we bisect in the direction of the largest component of the radius of $S_{(k)}$.

Suppose that we not only have access to the interval extension F, but also to an interval extension F' of the *gradient* $f' = \left(\dfrac{\partial f}{\partial x_1}, \dots, \dfrac{\partial f}{\partial x_n}\right)$ and the

Hessian $f' = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)$. This information can be used in two ways:

1. Monotonicity. If $0 \notin \left(F'(S_{(k)})\right)_i$, then f is monotone in the component x_i on the subdomain $S_{(k)}$. Therefore, we can reduce the ith component of the interval vector $S_{(k)}$ to its lower (respectively upper) bound if $\left(F'(S_{(k)})\right)_i > 0$ (respectively $\left(F'(S_{(k)})\right)_i < 0$). Furthermore, if this reduced candidate is interior to the original domain D, then we can discard it from S because all components of the gradient at an interior minimizer are zero.

2. Stationary points. Since an interior minimizer is a solution of the equation $f'(x) = 0$ we can use an interval equation solver to locate this minimizer. We prefer Krawczyk's method, [CMN02], [Kra69], which is a version of the interval Newton method based on the operator

$$K(x, X) \;\equiv\; x - H\, f'(x) + (I - H\, J(X))(X - x). \tag{4}$$

Here, $x \in X$, $I = \text{diag}(1, \ldots, 1)$ is the unit matrix, J is an interval extension the Jacobian of f' (i.e., the Hessian f'') and H is an arbitrary matrix in $\mathbb{R}^{n \times n}$. For efficiency reasons H should be chosen close to the inverse of $f''(x)$. K has the following properties

- If x^* is a solution then $x^* \in X \Rightarrow x^* \in K(x, X)$.
- If $K(x, X) \subseteq X$ then there exists a solution x^* in $K(x, X)$.

Therefore the nested sequence

$$X_{(s+1)} \;=\; X_{(s)} \cap K(x_{(s)}, X_{(s)}) \;\text{ with }\; x_{(s)} \in X_{(s)} \qquad s = 0, 1, \ldots \tag{5}$$

has the properties

- If $x^* \in X_{(0)}$ then $x^* \in X_{(s)}$, $s = 1, 2, \ldots$.
- If $X_{(s)} = \emptyset$ then no solution exists in $X_{(s)}$, and thus no solution exists in $X_{(0)}$.

Furthermore it has been proved that

- If $\{X_{(\cdot)}\}$ is convergent to x^* and $f''(x^*)$ is non-singular then the rate of convergence of (5) is quadratic.

  Using (5) from $X_{(0)} = X = S_{(k)}$ there are three possible results:

  a) If $X_{(s)} = \emptyset$ for some value of s then X contains no root. If X is interior in D, then we can discard X from the candidate set, otherwise we can reduce it to the union of its non-interior edges.

  b) $\{X_{(\cdot)}\}$ converges to x^*. If (5) is stopped after iteration number s then any solution contained in X is also contained in $X_{(s+1)}$. Therefore the set of points in X which are not in $X_{(s+1)}$, $X \setminus X_{(s+1)}$, can be discarded from further search.

  c) The iteration (5) stalls, maybe because X is too wide. Use the splitting strategy to reduce the width.

Now we are ready to outline the algorithm. Let $w(X)$ denote the width of the interval X. We use the condition

$$w(S_{(k)}) \leq \delta \tag{6}$$

to decide when no further search should be done. Sub-boxes satisfying (6) are stored in the result set R. S is the Candidate Set. τ is the threshold value used in (2),(3), and we choose x_k as the mid point of the interval $S_{(k)}$.

In each iteration we wish to pick the most promising box from S. This is chosen as the interval $S_{(k)}$ which has the smallest lower bound $L(S_{(k)})$.

For simplicity we assume that all minimizers are interior in D. Then the algorithm has the following structure:

Algorithm MS:

```
$S_{(1)} := D$
$S := \{S_{(1)}\}$
$\tau := f(\text{mid}(S_{(1)}))$
while $S \neq \emptyset$ do
    $X$ := the most promising box in $S$
    remove $X$ from $S$
    if Monotone($X$) then    { see 1. above }
        $R\_X := \emptyset$
        else if ( 2a or 2b ) then    { see 2. above }
              $X$ is reduced to $R\_X$
        else { split }
          $R\_X := (X_1, X_2)$ where $X = X_1 \cup X_2$
    end
    { R_X contains 0, 1 or 2 elements }
    with all $Z \in R\_X$
        $\tau := \min\{\tau, f(\text{mid}(Z))$
        if $w(F(Z)) \leq \delta$ then $R := R \cup Z$
                                   else $S := S \cup Z$
    end
end { while }
```

This algorithm locates all solutions to (1), i.e., all solutions are contained in R which is the union of sub-boxes each of which having width less than δ. Algorithm MS has proven to be very efficient for problems with a rather modest number of variables (up to 15-20, say). If the number of variables is higher then the computing time may be severe since the worst case complexity of the algorithm is exponential. Under special circumstances, however, the algorithm may be efficient, even for a large number of variables.

2.2 Constraint Propagation

Constrained propagation is here considered as an interval method for solving equations, as described in [KK05] and [Sem94]. It is used as a method to reduce the set of possible candidates for a solution. The reduction is done rigorously, i.e., the method guarantees that no solution contained in the candidate set is lost.

The method is based on the *sub-definite calculations*. In order to solve a desired equation or inequality, the constraint corresponding to the value of the expression needs to be *propagated* through the expression. For instance, if we wish to find which values of $x \in \mathbb{R}$ satisfy the inequality $3x - 2 \geq 5$ then the feasible interval of function values $[5, \infty]$ is propagated through the expression $3x - 2$ until the set of feasible values of x is calculated.

Knowing the bounds of the whole expression, we first calculate the bounds of $3x$ and then the bounds of x. A very convenient way of illustrating how the method works (and actually the way to implement it too) is by constructing a *calculus tree* for the expression and propagating the constraint through it, see Fig. 1. The calculus tree is made in such a way that each node corresponds to an operator in the expression of the function and each leaf in the tree being either a variable or a constant.

The operators can be both binary and unary, i.e., the nodes of the tree can have either one or two children. In the context of global optimization constraint propagation is used to locate the set of stationary points, i.e., solving the equation $f'(x) = 0$. This equation is called the *propagated constraint*.

Thus the initial feasible set of function values, i.e., the interval attached to the root of the calculus tree representing $f'(x)$, is $[0, 0]$. Furthermore, the ranges of values for the independent variables are assigned. Finally interval values of the other nodes are assigned; if nothing is known a priori then the value $[-\inf, +\inf]$ is assigned to these nodes. Thus, all nodes in the tree are assigned an interval value. The constraint propagation method intends to reduce the interval ranges without loosing any solution.

Based on the propagated constraint the method works by walking through the calculation tree and updating the ranges of values attached to the nodes. For each calculation the intersection with the previous range is used. The method continues until no further changes in the values attached to the nodes can be made. The method does not necessarily provide precise bounds for the solutions, however no solution inside the initial interval of the variables is lost.

Following [Sem94] we base the propagation on a so-called *code list*. For each node in the expression tree the code list displays the interaction between the node and its neighbours. Thus each function in the code list consists of just one operator, the one that defines the corresponding node in the tree, or its inverse operator. The technique is illustrated through the following example.
Example 1. We wish to solve the equation

$$(f'(x) \equiv) \quad e^x - x - 2 = 0. \tag{7}$$

The calculus tree for the function $e^x - x - 2$ is shown in Fig. 2. Since the exponential is positive the initial value of the corresponding node, denoted by t_1, is included in the interval $]0, \infty]$. Equation (7) implies that the node

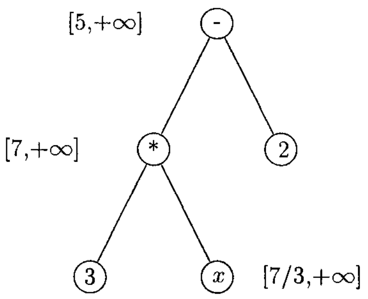


Fig. 1. The calculus tree corresponding to $3x - 2$. The initial bound on the expression and the propagated bounds are shown as intervals

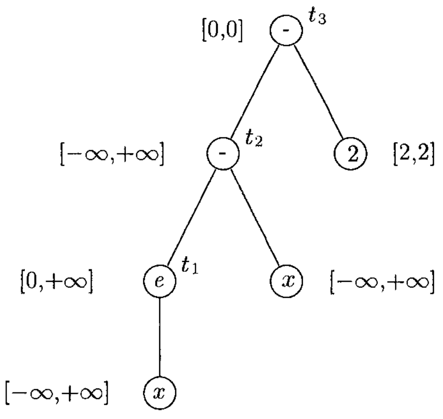


Fig. 2. Calculus tree for the function $e^x - x - 2$ including the initial intervals for the nodes. The node names t_1, t_2, t_3 refer to the code list in Table 1

Table 1. The code list for the function $e^x - x - 2$

$f_1 : t_1 := exp(x) \cap t_1$	$f_2 : x := ln(t_1) \cap x$	$f_3 : t_2 := (t_1 - x) \cap t_2$
$f_4 : t_1 := (t_2 + x) \cap t_1$	$f_5 : x := (t_1 - t_2) \cap x$	$f_6 : t_2 := (t_3 + 2) \cap t_2$
$f_7 : t_3 := (t_2 - 2) \cap t_3$		

denoted by t_3 has the value 0 attached. The variables t_2 and x are a priori only known to belong to the real line $[-\infty, \infty]$.

As one can immediately see the code list is more than just another form for writing the mathematical expression for the function. There are only three operators in (7), however the code list consists of seven expressions. The reason for this is that the functions f_2, f_4, f_6 and f_5 are inferred from f_1, f_3 and f_7, thus giving the opportunity to move up and down the calculus tree and propagate the constraint.

Now the idea is to reduce the intervals using the code list and the known inclusions. This goes on until no further reduction takes place. We start from below in the code list (Table 1) and move upwards: Since $t_3 = 0$ we obtain $t_2 = 2$. Next f_5 gives the interval [-2, ∞] for x. This implies that t_1 is reduced to [exp(-2),∞]= [0.135,∞] by f_1. If we repeat this sweep we obtain further reductions in t_1 and x, however the rate of reduction is rather slow. In the global optimization problem (1) the variables are finitely bounded. Therefore, let us repeat the example starting with finite bounds on x, $x \in [-1000, 1000]$. Again we obtain $t_3 = 0$ and $t_2 = 2$ initially. Then we obtain the values given in Table 2 (in order to be rigorous we round intervals outwards).

Table 2. The first fifteen steps for the example (7)

Step	**Working**	**Sub-definite values**			
num.	**function**	t_3	t_2	t_1	x
0		[0,0]	$[-\infty, +\infty]$	$[0, +\infty]$	$[-1000, +1000]$
1	f_6		$[2, 2]$		
2	f_5				$[-2, 1000]$
3	f_4			$[0, 1002]$	
4	f_2				$[-2, 6.91]$
5	f_1			$[0.13, 1002]$	
6	f_5				$[-1.87, 6.91]$
7	f_4			$[0.13, 8.91]$	
8	f_2				$[-1.87, 2.19]$
9	f_1			$[0.15, 8.91]$	
10	f_5				$[-1.85, 2.19]$
11	f_4			$[0.15, 4.19]$	
12	f_2				$[-1.85, 1.43]$
13	f_1			$[0.15, 4.18]$	
14	f_4			$[0.15, 3.43]$	
15	f_2				$[-1.85, 1.24]$

In practice the code list is easily derived from the corresponding calculus tree. The picture of the tree actually represents how the tree or DAG (Directed Acyclic Graph) looks in our implementation. This is illustrated in Table 1 and Fig. 2.

Schematically the constraint propagation method as described in [KK05], [Sem94] is the following:

Algorithm CP

1. Execute all the functions of the constructed code list.
2. If the value of any variable has been changed after the execution of one of the functions, then mark all functions having this variable as an argument.
3. If the set of marked functions is empty then stop.
4. Select and execute an arbitrary function from the set of marked functions and go to 2.

The algorithm corresponds to going up and down the calculus tree, executing the functions of the code list, until no more changes can be made in the node values. As the propagation can go both upwards and downwards a node that can tentatively be changed is included into the set of marked functions, *the active function set.* It keeps the track of the updates of the node values.

The selection in (4) can be done in numerous ways. If a node value in the calculus tree has been changed, then one could for instance update the values of its children or one could update the values of all of the neighbours of the node. Such a small difference can play a big role in the performance of constraint propagation, however it is not clear which choice is generally the best.

However, our experience indicates that in connection with the global optimization algorithm the most efficient is to perform only one sweep and stop, i.e., in Example 1 we would stop after step number 5. This is because the major changes often take place in the first sweep.

2.3 Algorithm MS Extended with Constraint Propagation

The *Krawczyk method* (5) and the *Algorithm CP* both intend to solve a nonlinear set of equations ($f'(x) = 0$). However they perform quite differently. Krawczyk provides fast convergence to a narrow (machine precision) box when the necessary condition is satisfied. Unfortunately the box $X_{(0)}$ often has to be quite narrow around the solution in order for Krawczyk to converge. The CP is normally not as quickly convergent as Krawczyk and the limit box cannot be expected to be as narrow as that provided by Krawczyk, especially when the function expression is complex. However CP does not necessarily need a narrow initial box. Therefore CP may often be be used to provide an initial reduction whereas Krawczyk provides the ultimate convergence. CP may even reduce boxes with more than one stationary point. This is demonstrated in Fig. 3.

Figure 3 shows the contour plot for the function

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 3.$$

If we apply CP with the starting box $[-2, 2] \times [-2, 2]$, then the algorithm will reduce the box to the small one, just containing the two stationary points.

These observations indicate that a combination of the two methods may exploit the strong sides of each of them. Therefore we use Algorithm CP if

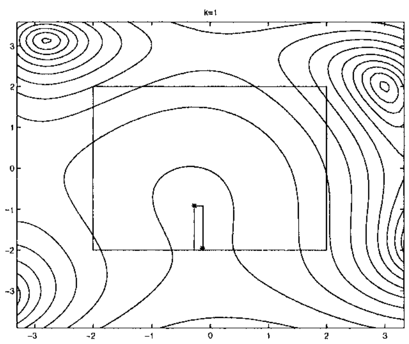


Fig. 3. Contour plots. Constraint propagation tightening the box $[-2, 2] \times [-2, 2]$ to a narrow box around the two stationary points marked as dots

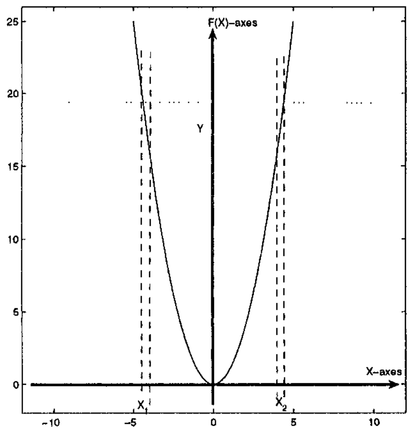


Fig. 4. Illustration of the problem occurring when implementing the inverse of the function $y = x^2$

the Krawczyk method does not provide any reduction in Algorithm MS. In other words we use both methods as root finding methods in Algorithm MS. This is done by first employing one of them; if no progress has been made then we apply the other. If the candidate has still not been reduced then it is split into two. Thus the only difference compared with Algorithm MS is that instead of one interval method for solving $f'(x) = 0$ two interval methods are tried.

The combined algorithm is the following:

```
Algorithm MSCP:
$S_{(1)} := D$
$S := \{S_{(1)}\}$
$\tau := f(\mathrm{mid}(S_{(1)}))$
while $S \neq \emptyset$ do
    $X$ := the most promising box in $S$
    remove $X$ from $S$
    if Monotone($X$) then    { see 1. in Subsection 2.1 }
        $R\_X := \emptyset$
        else if ( 2a or 2b ) then    { see 2. in Subsection 2.1 }
              $X$ is reduced to $R\_X$
        else if ( CP works ) then
              $X$ is reduced to $R\_X$
        else { split }
           $R\_X := (X_1, X_2)$ where $X = X_1 \cup X_2$
    end
    { R_X contains 0, 1 or 2 elements }
    with all $Z \in R\_X$
        $\tau := \min\{\tau, f(\mathrm{mid}(Z))\}$
        if $w(F(Z)) \leq \delta$ then $R := R \cup Z$
                                 else $S := S \cup Z$
    end
end { while }
```

3 Implementation and Numerical Results

Algorithm MSCP has been implemented in C++ using SUN's Interval package (Suninterval). We have tested the programme on several problems. Here we illustrate the results on two examples only.

3.1 Implementation

When implementing Constraint Propagation a tree-structure is used. In fact, the constraint programming implementation is an extension of the automatic differentiation library FADBAD [CFG+02], [SB03], which implements forward and backward differentiation. FADBAD has been extended in such a way that the expression trees used to store functions and derivatives are reused for implementing the constraint propagation.

Thus the integrated programme automatically calculates first and second derivatives as well as the CP code, and the user only has to programme the expression for $f(x)$.

A discussion of some implementation issues can be found in [KK05], Sect. 7. Here we shall only mention one difficulty which is connected with power functions. We illustrate the problem by considering the square operator, $y = x^2$. The problem occurs because the inverse operator splits into two, $+\sqrt{x}$ and $-\sqrt{x}$. Thus in such a case a more complicated tree structure is needed which might give rise to exponential growth in the amount of space needed to store the tree.

The situation is illustrated in Fig. 4. When intersecting the two new intervals $X_1 = +\sqrt{X}$ and $X_2 = -\sqrt{X}$ with the previous interval value X_{prev} we have the following three cases:
If $X_{prev} \cap X_1 = \emptyset$ then $X_{new} := X_{prev} \cap X_2$, and no problem occurs.
If $X_{prev} \cap X_2 = \emptyset$ then $X_{new} := X_{prev} \cap X_1$, and no problem occurs.
If $X_{prev} \cap X_1 \neq \emptyset$ and $X_{prev} \cap X_2 \neq \emptyset$ then X_{new} splits into two unless $0 \in X$. In order to avoid this problem we let X_{new} be the hull of the two non-empty intersections.

Our implementation can be formulated as follows:

$$X_{new} := (X_{old} \cap X_1) \bigcup (X_{old} \cap X_2),$$

where $\bigcup$ is the interval hull of the union.

This implementation is of course rigorous, however it is pessimistic, since we throw away information by taking the hull. This may cause a slower convergence rate of the CP algorithm, and a sharp implementation of inverse power functions is planed in the future.

3.2 Some Numerical Tests

In order to investigate the efficiency of Algorithm MSCP it has been tested on several problems. Some of these tests are described in [KK05] where different variations of the implementation are examined.

The most important results of these tests are the following:

- Since Algorithm CP as described in Subsection 2.2 is running the same simple equations in each CP iteration, it can be expected to provide most of the reduction at the beginning. Does it matter whether Algorithm CP is run until the end, where no further reduction of the interval is possible (as it is described in Subsection 2.2), or is it better to run Algorithm CP just once down the tree, i.e., to execute each working function in the code list only once, as described in [Mes04]?

On the test examples tried it turned out that Algorithm MSCP was about twice as fast when the one-loop implementation of Algorithm CP was used rather than running full CP in each iteration.

- Krawczyk's method (4) was tested in a loop as described in (5) as well as in just one iteration of (5). Like Algorithm CP Krawczyk's method gives the largest reduction in the first iteration. And on the test examples tried in [KK05] it also turned out that Algorithm MSCP was fastest when only one loop of (5) was used, rather than running full iteration each time.

In the test examples below we have used these results: For (5) as well as for Algorithm CP only one loop is run in each iteration of Algorithm MSCP.

The first illustration is the so-called Schwefel test problem [Sch81] which is tested for several values of n:

$$f(x) = \sum_{i=1}^{n}\{-x_i \sin(\sqrt{x_i})\}, \qquad D = [1, 500]^n.$$

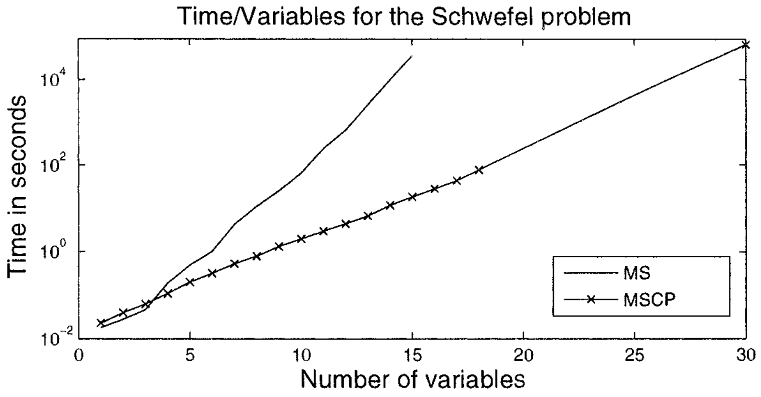
Figure 5 shows the performance of the two algorithms for $n = 1, 2, \ldots, 18, 30$. It is seen that although the computing time is exponential in n for both algorithms, Algorithm MSCP is far better than Algorithm MS. The numbers of splits show the same tendency as the CPU time. The computing times for $n = 15$ are approximately 10 hours for Algorithm MS and 18 seconds for Algorithm MSCP.

Figure 5 illustrates some tendencies we also have seen in other examples: The constraint programming is very efficient when power functions are involved, and it is not so sensitive as Algorithm MS to an increasing number of variables, i.e., the factor in the exponential growth is much smaller. In other words: The more variables the better Algorithm MSCP compares with the classical Algorithm MS.

It is easily seen that the number of splits in Algorithm MS cannot be less than the number of splits in Algorithm MSCP. However, the amount of work per split is higher for Algorithm MSCP, and we have seen examples where the CPU time for the latter method is higher. This is for instance the case for the so-called Levy function, [HW04], when the number of variables is less than 6. It is the following:

$$f(x) = \sin(3\pi x_1) + \sum_{i=1}^{n-1}\{(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1}))\} + (x_n - 1)(1 + \sin^2(3\pi x_{i+1})),$$

where the initial box is $D = [-5, 5]^n$. This function has a very large number of local minimizers: For $n = 5, 6, 7$ it is approximately $10^5, 10^6, 10^8$, respectively. In this case the number of splits for Algorithm MSCP is about half of the number of splits in Algorithm MS. The computing times for Algorithm MS are 11, 83 and 537 seconds for $n = 5, 6, 7$, respectively, whereas the corresponding computing times for Algorithm MSCP are 12, 80 and 330 seconds, respectively.



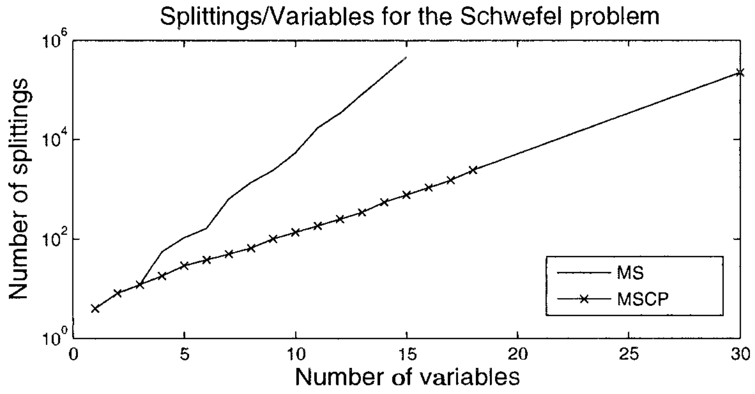


Fig. 5. Plot showing the CPU times and the number of interval splits when applying the two algorithms Algorithm MS and Algorithm MSCP to the Schwefel problem for $n = 1, 2, \ldots, 18, 30$

We have made an experimental comparison with R. Baker Kearfott's optimization package GlobSol [Kea03], release November 2003, which also uses a combination of interval global optimization and constraint propagation. GlobSol has a lot of features and is a much more complex programme than Algorithm MSCP. Like Algorithm MSCP GlobSol includes automatic differentiation. In order to make a fair comparison we used the mode in GlobSol where it is assumed that no solution exists at the boundary of D.

In general it turned out that for many problems GlobSol was faster than Algorithm MSCP. When power functions are involved in calculating f, however, Algorithm MSCP is much faster than GlobSol. For problems where we could increase the number of variables n, both programmes illustrated an exponential growth similar to that displayed in Fig.5.

The two examples in this chapter are typical for the comparisons: For the Levy function, $n = 7$, GlobSol used 54 seconds to find the solution with high accuracy whereas Algorithm MSCP used 5.5 minutes. For the Schwefel problem Algorithm MSCP was much faster when n is large. For $n = 10$ GlobSol used 2.5 minutes to to find the solution with high accuracy whereas Algorithm MSCP used 1 second. For $n = 13$ the CPU times were 50 minutes for GlobSol and 6 seconds for Algorithm MSCP.

4 Conclusion

The classical interval global optimization algorithm has proved to be very efficient for a large class of problems, especially when the number of variables is modest, [HW04]. We combine this method with constrained propagation, as a tool for enclosing the set of stationary points. The combination has been implemented and tested, and two typical test examples are displayed in this chapter. An important fact that can be concluded from the tests in [KK05] is, that the use of constraint propagation makes each iteration of the global optimization algorithm more time consuming, however the number of iterations and bisections is often reduced, sometimes drastically. This fact is crucial when dealing with the problems with high number of variables.

References

[CFG+02] Corliss, G., Faure, C., Griewank, A., Hascoët, L., Naumann, U. (eds): Automatic Differentiation of Algorithms. Springer Verlag, New York (2002)

[CMN02] Caprani, O., Madsen, K., Nielsen, H.B.: Introduction to Interval Analysis. Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark (2002)

[Fle87] Fletcher, R.: Practical Methods of Optimization. 2nd ed. John Wiley and Sons (1987)

[GBH01] Granvilliers, L., Benhamou, F., Huens, E.: Constraint propagation. In:COCONUT Deliverable D1 : Algorithms for Solving Nonlinear Constrained and Optimization Problems : The State of The Art. The COCONUT Project (2001)

[HMD97] Hentenryck, P.V., Michel, L., Deville, Y.: Numerica. MIT Press (1997)

[HPT00] Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. 2nd ed. Nonconvex optimization and its applications Vol. **48**, Kluwer Academic Publishers (2000)

[HW04] Hansen, E., Walster, G.W.: Global Optimization Using Interval Analysis. 2nd ed. Marcel Dekker Inc., New York (2004)

[Kea03] Kearfott, R.B.: The GlobSol Project. Release date 22 november (2003) http://interval.louisiana.edu/private/downloading_instructions.html

[KK05] Kjøller, S., Kozine, P.: Global Optimization Using Constraint Propagation and Krawczyk's method. Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark (2005)

[Kra69] Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nulstellen mit Fehlerschranken. Computing, **4**, 187–201 (1969)

[Mad96] Madsen, K.: Real versus interval methods for global optimization. Presentation at the Conference 'Celebrating the 60th Birthday of M.J.D. Powell', Cambridge, July (1996)

[Mes04] Messine, F.: Deterministic global optimization using interval constraint propagation techniques. RAIRO Operations Research, **38**, 277–293 (2004)

[Moo66] Moore, R.E.: Interval Analysis. Prentice Hall, Englewood Cliffs, N.J. (1966)

[Moo76] Moore, R.E.: On computing the range of values of a rational function of n variables over a bounded region. Computing, **16**, 1–15 (1976)

[MZ00a] Madsen, K., Žilinskas, J.: Testing of Branch-and-Bound Methods for Global Optimization. IMM-REP-2000-05, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark (2000)

[MZ00b] Madsen, K., Žilinskas, J.: Evaluating performance of attraction based subdivision method for global optimization. In: Second International Conference 'Simulation, Gaming, Training and Business Process Reengineering in Operations', RTU, Latvia, 38–42 (2000)

[Neu90] Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press (1990)

[Neu01] Neumaier, A.: Introduction to Numerical Analysis. Cambridge University Press (2001)

[NW99] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Verlag (1999)

[RRM93] Rayward-Smith, V.J., Rush, S.A., McKeown, G.P.: Efficiency considerations in the implementation of parallel branch-and-bound. Annals of Operations Research, **43**, 123–145 (1993)

[SB03] Stauning, O., Bendtsen, C.: Flexible Automatic Differentiation Using Templates and Operator Overloading in ANSI C++. http://www2.imm.dtu.dk/~km/FADBAD/, Technical University of Denmark (2003)

[Sch81] Schwefel, H.: Numerical Optimization of Computer Models. John Wiley and Sons (1981)

[Sem94] Semenov, A.: Solving Integer/Real Equations by Constraint Propagation. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark (1994)

[Ske74] Skelboe, S.: Computation of rational interval functions. BIT, **14**, 87–95 (1974)

[Sta94] Stauning, O.: Automatic Validation of Numerical Solutions. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark (1994)

[TZ87] Törn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, **350**, Springer Verlag (1987)

Towards Optimal Compression of Meteorological Data: A Case Study of Using Interval-Motivated Overestimators in Global Optimization

Olga Kosheleva

Department of Electrical and Computer Engineering and Department of Teacher Education, University of Texas, El Paso, TX 79968, USA olgak@utep.edu

1 Introduction

The existing image and data compression techniques try to minimize the mean square deviation between the original data $f(x,y,z)$ and the compressed-decompressed data $\widetilde{f}(x,y,z)$. In many practical situations, reconstruction that only guaranteed mean square error over the data set is unacceptable.

For example, if we use the meteorological data to plan a best trajectory for a plane, then what we really want to know are the meteorological parameters such as wind, temperature, and pressure along the trajectory. If along this line, the values are not reconstructed accurately enough, the plane may crash; the fact that on average, we get a good reconstruction, does not help.

In general, what we need is a compression that guarantees that for each (x,y), the difference $|f(x,y,z)-\widetilde{f}(x,y,z)|$ is bounded by a given value Δ, i.e., that the actual value $f(x,y,z)$ belongs to the interval

$$[\widetilde{f}(x,y,z)-\Delta,\widetilde{f}(x,y,z)+\Delta].$$

In this chapter, we describe new efficient techniques for data compression under such interval uncertainty.

2 Optimal Data Compression: A Problem

2.1 Data Compression Is Important

At present, so much data is coming from measuring instruments that it is necessary to compress this data before storing and processing. We can gain some storage space by using lossless compression, but often, this gain is not sufficient, so we must use lossy compression as well.

2.2 Successes of 2-D Image Compression

In the last decades, there has been a great progress in image and data compression. In particular, the JPEG2000 standard (see, e.g., [TM02]) uses the wavelet transform methods together with other efficient compression techniques to provide a very efficient compression of 2D images.

Within this standard, we can select different bitrates (i.e., number of bits per pixel that is required, on average, for the compressed image), and depending on the bitrate, get different degrees of compression.

When we select the highest possible bitrate, we get the lossless compressions that enables us to reconstruct the original image precisely. When we decrease the bitrate, we get a lossy compression; the smaller the bitrate, the more the compressed/decompressed image will differ from the original image.

In principle, it is possible to use these compression techniques to compress 2D measurement data as well.

2.3 Compressing 3-D Data: Layer-by-Layer Approach

It is also possible to compress 3D measurement data $f(x, y, z)$, e.g., meteorological measurements taken in different places (x, y) at different heights z.

One possibility is simply to apply the 2D JPEG2000 compression to each horizontal layer $f(x, y, z_0)$.

2.4 Compressing 3-D Data: An Approach that Uses KLT Transform

Another possibility, in accordance with Part 2 of JPEG2000 standard [TM02], is to first apply the Karhunen-Loève (KLT) transform to each vertical line. Specifically, we:

- compute the average value
$$\bar{f}(z) = \frac{1}{N} \cdot \sum_{x,y} f(x, y, z)$$
of the analyzed quantity at a given height z, where N is the overall number of horizontal points (x, y);
- compute the covariances between different heights:
$$V(z_1, z_2) = \frac{1}{N} \cdot \sum_{x,y} (f(x, y, z_1) - \bar{f}(z_1)) \cdot (f(x, y, z_2) - \bar{f}(z_2));$$
- find the eigenvalues λ_k and the eigenvectors $e_k(z)$ of the covariance matrix $V(z_1, z_2)$; we sort these eigenvectors into a sequence
$$e_1(z), e_2(z), \ldots$$
so that
$$|\lambda_1| \geq |\lambda_2| \geq \ldots;$$

- finally, we represent the original 3D data values $f(x, y, z)$ as a linear combination
$$f(x,y,z) = \bar{f}(z) + \sum_k a_k(x,y) \cdot e_k(z)$$
of the eigenvectors $e_k(z)$.

2.5 An Approach that Uses KLT Transform: Details

How can we implement this approach?

- The first two steps are straightforward.
- The third step (computing eigenvalues and eigenvectors) is a known computational problem for which many standard software packages provides an efficient algorithmic solution.
- So, to specify how this method can be implemented, we must describe how the last step can be implemented, i.e., how we can represent the original 3D data as a linear combination of the eigenvectors.

First, why is such a representation possible at all? It is known (see, e.g., [TM02]), that in general, the eigenvalues of a covariance matrix form a orthonormal basis in the space of all N_z-dimensional vectors

$$e = \{e(z)\} = (e(1), e(2), \ldots, e(N_z)).$$

By definition of a basis this means, in particular, for every (x, y), the corresponding difference vector

$$d_{x,y}(z) \stackrel{\text{def}}{=} f(x,y,z) - \bar{f}(z)$$

can be represented as a linear combination of these eigenvalues, i.e., that for every z, we have

$$d_{x,y}(z) = f(x,y,z) - \bar{f}(z) = \sum_k a_k(x,y) \cdot e_k(z),$$

where by $a_k(x, y)$, we denoted the coefficient at $e_k(z)$ in the corresponding expansion. From this formula, we get the desired representation of $f(x, y, z)$.

How can we actually compute this representation? Since the vectors $e_k(z)$ form an orthonormal basis, we can conclude that for the expansion of the vector $d_{x,y}$, each coefficient $a_k(x, y)$ at $e_k(z)$ is a dot (scalar) product of the vector $d_{x,y}$ and the k-th vector $e_k(z)$ from the basis, i.e., that

$$a_k(x,y) = d_{x,y} \cdot e_k \stackrel{\text{def}}{=} \sum_z d_{x,y}(z) \cdot e_k(z).$$

Substituting the expression for $d_{x,y}(z)$ into this formula, we conclude that

$$a_k(x,y) = \sum_z (f(x,y,z) - \bar{f}(z)) \cdot e_k(z).$$

Comment. Actually, instead of using this explicit (thus fast-to-compute) formula, we can use even faster formulas of the so-called *fast KLT transform* (see, e.g., [TM02]).

2.6 KLT Transform: Result

As a result of the KLT approach, we represent the original 3-D data as a sequence of the horizontal *slices* $a_k(x, y)$:

- the first slice $a_1(x, y)$ corresponds to the main (1-st) eigenvalue;
- the second slice $a_2(x, y)$ corresponds to the next (2-nd) eigenvalue;
- etc.,

with the overall intensity decreasing from one slice to the next one.

Next, to each of these slices $a_k(x, y)$, we apply a 2D JPEG2000 compression with the appropriate bit rate b_k depending on the slice k.

2.7 Decompressing 3-D Data: KLT-Based Approach

To reconstruct the data, we so the following:

- First, we apply JPEG2000 decompression to each slice; as a result, we get the values $\widetilde{a}_k^{[b_k]}(x, y)$.
- Second, based on these reconstructed slices, we now reconstruct the original 3-D data data as

$$\widetilde{f}(x, y, z) = \bar{f}(z) + \sum_k \widetilde{a}_k^{[b_k]}(x, y) \cdot e_k(z).$$

2.8 Data Compression as an Optimization Problem: an Informal Introduction

Different bit rate allocations b_k lead to different quality of (lossy) compression, i.e., to different quality of the corresponding decompressed data.

Among all possible compression schemes that guarantee the given reconstruction accuracy (in some reasonable sense), we want to find the scheme for which the average bitrate

$$\bar{b} \stackrel{\text{def}}{=} \frac{1}{N_z} \cdot \sum_k b_k$$

is the smallest possible.

In some cases, the bandwidth is limited, i.e., we know the largest possible average bitrate b_0. In such cases, among all compression schemes with $\bar{b} \le b_0$, we must find a one for which the compression/decompression error is the smallest possible.

2.9 JPEG2000 Approach Is Tailored Towards Image Processing and Towards Mean Square Error (MSE)

In the JPEG2000-based approach, for compressing measurement *data*, we use *image* compression techniques.

The main objective of image compression is to retain the quality of the image. From the viewpoint of visual image quality, the image distortion can be reasonably well described by the mean square difference MSE (a.k.a. L^2-norm) between the original image $I(x, y)$ and the compressed-decompressed image $\widetilde{I}(x, y)$.

As a result, sometimes, under the L^2-optimal compression, an image may be vastly distorted at some points (x, y), and this is OK as long as the overall mean square error is small.

2.10 For Data Compression, MSE May Be a Bad Criterion

When we compress *measurement* results, our objective is to be able to reproduce *each* individual measurement result with a certain *guaranteed* accuracy.

In such a case, reconstruction that only guaranteed mean square error over the data set is unacceptable: for example, if we use the meteorological data to plan a best trajectory for a plane, what we really want to know are the meteorological parameters such as wind, temperature, and pressure along the trajectory.

If along this line, the values are not reconstructed accurately enough, the plane may crash. The fact that on average, we get a good reconstruction, does not help.

2.11 An Appropriate Criterion for Data Compression

What we need is a compression that guarantees the given accuracy for all pixels, i.e., that guarantees that the L^∞-norm

$$\max_{x,y,z} |f(x, y, z) - \widetilde{f}(x, y, z)|$$

is small.

In other words, what we need is a compression that guarantees that for each (x, y), the difference $|f(x, y, z) - \widetilde{f}(x, y, z)|$ is bounded by a given value Δ. i.e., that the actual value $f(x, y, z)$ belongs to the interval

$$[\widetilde{f}(x, y, z) - \Delta, \widetilde{f}(x, y, z) + \Delta].$$

Comment. In engineering applications of interval computations, "interval uncertainty" usually means that the problem's parameters are uncertain *parameters*, known only with interval uncertainty.

In the above data compression problem, we have a *non-parametric* problem, so the traditional engineering meaning of interval uncertainty does not apply. In our problem, by interval uncertainty, we mean guaranteed bounds on the loss of accuracy due to compression.

2.12 Need for Optimal Data Compression under Interval Uncertainty

There exist several compressions that provide guaranteed bounds on the loss of accuracy due to compression. For example, if for each slice, we use the largest possible bitrate (corresponding to lossless compression), then $\widetilde{a}_k(x,y) = a_k(x,y)$ hence $\widetilde{f}(x,y,z) = f(x,y,z)$, i.e., there is no distortion at all.

What we really want is, among all possible compression schemes that guarantee the given upper bound Δ on the compression/decompression error, to find the scheme for which the average bitrate

$$\bar{b} \stackrel{\text{def}}{=} \frac{1}{N_z} \cdot \sum_k b_k$$

is the smallest possible.

In some cases, the bandwidth is limited, i.e., we know the largest possible average bitrate b_0. In such cases, among all compression schemes with $\bar{b} \le b_0$, we must find a one for which the L^∞ compression/decompression error is the smallest possible.

2.13 What We Have Done

In this chapter, on the example of meteorological data, we describe how we can use an interval-motivation overestimator to find a good quality data compression.

3 The Use of Interval-Motivated Overestimator

3.1 What Exactly We Do

Specifically, we use JPEG2000 to compress 3D measurement data with guaranteed accuracy. We are following the general idea of Part 2 of JPEG2000 standard; our main contribution is selecting bitrates which lead to a minimization of L^∞ norm as opposed to the usual L^2-norm.

3.2 Let Us Start Our Analysis with a 2-D Case

Before we describe how to compress 3-D data, let us consider a simpler case of compressing 2-D data $f(x,y)$. In this case, for each bitrate b, we can apply the JPEG2000 compression algorithm corresponding to this bitrate value. After compressing/decompressing the 2-D data, we get the values $\widetilde{f}^{[b]}(x,y)$ which are, in general, slightly different from the original values $f(x,y)$.

In the interval approach, we are interested in the L^∞ error

$$D(b) \overset{\text{def}}{=} \max_{x,y} \left| \widetilde{f}^{[b]}(x,y) - f(x,y) \right| .$$

The larger the bitrate b, the smaller the error $D(b)$. When the bitrate is high enough, so high that we can transmit all the data without any compression, then the error $D(b)$ becomes 0.

Our objective is to find the smallest value b for which the L^∞ error $D(b)$ does not exceed the given threshold Δ. For the 2-D case, we can find this optimal b_{opt} by using the following iterative *bisection* algorithm. In the beginning, we know that the desired bitrate lies between 0 and the bitrate B corresponding to lossless compression; in other words, we know that $b \in [b^-, b^+]$, where $b^- = 0$ and $b^+ = B$.

On each iteration of the bisection algorithm, we start with an interval $[b^-, b^+]$ that contains b_{opt} and produce a new half-size interval still contains b_{opt}. Specifically, we take a midpoint $b_{\text{mid}} \overset{\text{def}}{=} (b^- + b^+)/2$, apply the JPEG2000 compression with this bitrate, and estimate the corresponding value $D(b_{\text{mid}})$. Then:

- If $D(b_{\text{mid}}) \le \Delta$, this means that $b_{\text{opt}} \le b_{\text{mid}}$, so we can replace the original interval $[b^-, b^+]$ with the half-size interval $[b^-, b_{\text{mid}}]$.
- If $D(b_{\text{mid}}) > \Delta$, this means that $b_{\text{opt}} > b_{\text{mid}}$, so we can replace the original interval $[b^-, b^+]$ with the half-size interval $[b_{\text{mid}}, b^+]$.

After each iteration, the size of the interval halves. Thus, after k iterations, we can determine b_{opt} with accuracy 2^{-k}.

3.3 3-D Problem Is Difficult

In the 3-D case, we want to find the bitrate allocation $b_1, \ldots, b_{N_z}$ that lead to the smallest average bit rate b among all the allocations that fit within the given interval, i.e., for which the L^∞ compression/decompression error does not exceed the given value Δ:

$$D(b_1, b_2, \ldots) \le \Delta.$$

For each bit-rate allocation, we can explicitly compute this error, but there are no analytic formulas that describe this dependence, so we end up having to optimize a complex function with a large number of variables b_i.

Such an optimization is known to be a very difficult task, because the computational complexity of most existing optimization algorithms grows exponentially with the number of variables. There are theoretical results showing that in general, this growth may be inevitable; to be more precise, this problem is known to be NP-hard; see, e.g., [Vav91].

3.4 Main Idea: Using an (Interval-Motivated) Overestimator

In our case, the problem is, e.g., to find, among all bitrate allocations $(b_1, b_2, \ldots)$ with $\overline{b} \le b_0$, the one for which the L^∞ compression/decompression error $D(b_1, b_2, \ldots)$ is the smallest possible.

Since it is difficult to minimize the original function $D(b_1, \ldots)$, we:

- design an easier-to-optimize overestimator
$$\widetilde{D}(b_1, b_2, \ldots) \ge D(b_1, b_2, \ldots),$$
and then
- find the values b_i that minimize his overestimator $\widetilde{D}(b_1, \ldots)$.

As a result, we find an allocation b_i guaranteeing that $\widetilde{D}(b_1, \ldots) \le \widetilde{D}_{\mathrm{min}}$ and thus, that $D(b_1, \ldots) \le \widetilde{D}_{\mathrm{min}}$.

Comments.

- Since, in general, $D(b_1, \ldots) \le \widetilde{D}(b_1, \ldots)$, the resulting allocation is only *suboptimal* with respect to $D(b_1, \ldots)$.
- Overestimators and underestimators are a known tool in global optimization; see, e.g., [Flo00].
- The ideas of underestimators and overestimators are in good accordance with the main ideas behind interval computations; see, e.g., Appendix.

3.5 Explicit Formula for the Overestimator

Since we use the KLT, the difference
$$f(x, y, z) - \widetilde{f}(x, y, z)$$
is equal to
$$\sum_k (a_k(x, y) - \widetilde{a}_k^{[b_k]}(x, y)) \cdot e_k(z).$$
Therefore, once we know the L^∞-norms
$$D_k(b_k) \stackrel{\text{def}}{=} \max_{x,y} \left| a_k(x, y) - \widetilde{a}_k^{[b_k]}(x, y) \right|$$
of the compression/decompression errors of each slice, we can conclude that
$$\left| a_k(x, y) - \widetilde{a}_k^{[b_k]}(x, y) \right| \le D_k(b_k),$$
hence, that
$$\left| (a_k(x, y) - \widetilde{a}_k^{[b_k]}(x, y)) \cdot e_k(z) \right| \le D_k(b_k) \cdot E_k,$$
where
$$E_k \stackrel{\text{def}}{=} \max_z |e_k(z)|.$$
Thus, the desired L^∞ error is bounded by
$$\widetilde{D}(b_1, \ldots) \stackrel{\text{def}}{=} \sum_k D_k(b_k) \cdot E_k.$$

3.6 Resulting Algorithm: Derivation

In accordance with the above idea, to get the (suboptimal) bitrate allocation b_i, we must minimize the function

$$\widetilde{D}(b_1,\ldots) = \sum_k D_k(b_k) \cdot E_k$$

under the constraint that

$$\sum_k b_k = N_z \cdot b_0.$$

By using the Kuhn-Tucker approach, we can reduce this problem to the unconstrained problem, namely, to the problem of finding stationary points of the function

$$\sum_k D_k(b_k) \cdot E_k + \lambda \cdot \sum_k b_k - N_z \cdot b_0.$$

By definition of a stationary point, derivatives w.r.t. all the variables b_i should be 0s, so we end up with the equation

$$-D'_k(b_k) = \frac{\lambda}{E_k},$$

where the parameters λ should be selected based on the value b_0.

It can be easily shown that the other problem, the problem of minimizing the average bitrate under the constraint that the compression/decompression error does not exceed Δ, leads to the same equation.

As we have mentioned, the function $D_k(b)$ decreases when b increases, so $D'_k(b) < 0$, with $D'_k(b) \to 0$ as b grows. It is therefore reasonable to represent the desired equation as

$$|D'_k(b_k)| = \frac{\lambda}{E_k}.$$

What are the bounds on λ? The larger b_k, the smaller λ. From the above formula, we conclude that

$$\lambda = |D'_k(b_k)| \cdot E_k,$$

hence

$$\lambda \le \Lambda_k \overset{\text{def}}{=} \left(\max_b |D'_k(b)|\right) \cdot E_k,$$

so

$$\lambda \le \Lambda \overset{\text{def}}{=} \min_k \Lambda_k.$$

3.7 Algorithm: Description

Once we know, for each slice k, the dependence $D_k(b)$ of the corresponding L^∞-error on the bitrate b, we can find the desired (suboptimal) values b_k as follows.

At first, we compute the above-described values Λ_k and Λ. We know that $\lambda \leq [\lambda^-, \lambda^+] := [0, \Lambda]$. We use bisection to sequentially halve the interval containing λ and eventually, find the optimal value λ.

Once we know an interval $[\lambda^-, \lambda^+]$ that contains λ, we pick its midpoint λ_{mid}, and then use bisection to find, for each k, the value b_k for which

$$|D'_k(b_k)| = \frac{\lambda_{\text{mid}}}{E_k}.$$

Based on these b_k, we compute the average bitrate $\bar{b}$.

- If $\bar{b} > b_0$, this means that we have chosen too small λ_{mid}, so we replace the original λ-interval with a half-size interval $[\lambda_{\text{mid}}, \lambda^+]$.
- Similarly, if $\bar{b} < b_0$, we replace the original λ-interval with a half-size interval $[\lambda^-, \lambda_{\text{mid}}]$.

After k iterations, we get λ with the accuracy 2^{-k}, so a few iterations are sufficient. Once we are done, the values b_k corresponding to the final λ_{mid} are returned as the desired bitrates.

The only remaining question is how to determine the dependence $D_k(b)$. In principle, we can try, for each layer k, all possible bitrates b, and thus get an empirical dependence $D_k(b)$.

We have shown, that this dependence can be described by the following analytical formula:

- $A_1 \cdot (b - b_0)^\alpha$ for all the values b until a certain threshold b_0, and
- $A_2 \cdot 2^{-b}$ for $b \geq b_0$.

(This model is a slight modification of a model from [MF98].) So, instead of trying all possible values b, it is sufficient to try a few to determine the values of the parameters A_i, b_0, and α corresponding to the given layer.

3.8 Results

To test our algorithm, we applied it to 3-D meteorological data:

- temperature T,
- pressure P,
- the components U, V, and W of the wind speed vector, and
- the waver vapor missing ratio WV.

For meteorological data, the resulting compression indeed leads to a much smaller L^∞ error bound Δ_{new} than the L^∞ error bound Δ_{MSE} corresponding to the bitrate allocation that optimizes the MSE error. The ratio $\Delta_{\text{new}}/\Delta_{\text{MSE}}$ decreases:

- from 0.7 for $b_0 = 0.1$
- to 0.5 for $b_0 = 0.5$
- to ≤ 0.1 for $b_0 \geq 1$.

These results are shown in Fig. 1 that describes how, for different compression methods, the Maximum Absolute Error (MAE) Δ depends on the average bitrate $\bar{b}$:

- the solid line corresponds to the new method, in which the objective is to minimize the Maximum Absolute Error (MAE) Δ; as we have mentioned, to maximize this error, we must use multiple bit rates (MBR), i.e., different different rates for different horizontal "slices" $a_k(x, y)$;
- the dotted line describes the dependence of MAE Δ on the average bitrate $\bar{b}$ within the Part 2 JPEG2000 approach, when the individual bitrates b_k are selected to minimize the Mean Square Error (MSE);
- for comparison, the dashed line describes the dependence of MAE Δ on the average bitrate $\bar{b}$ if we use the same bit rate $b_k = \bar{b}$ for all slices, i.e., if we use one bit rate (OBR).

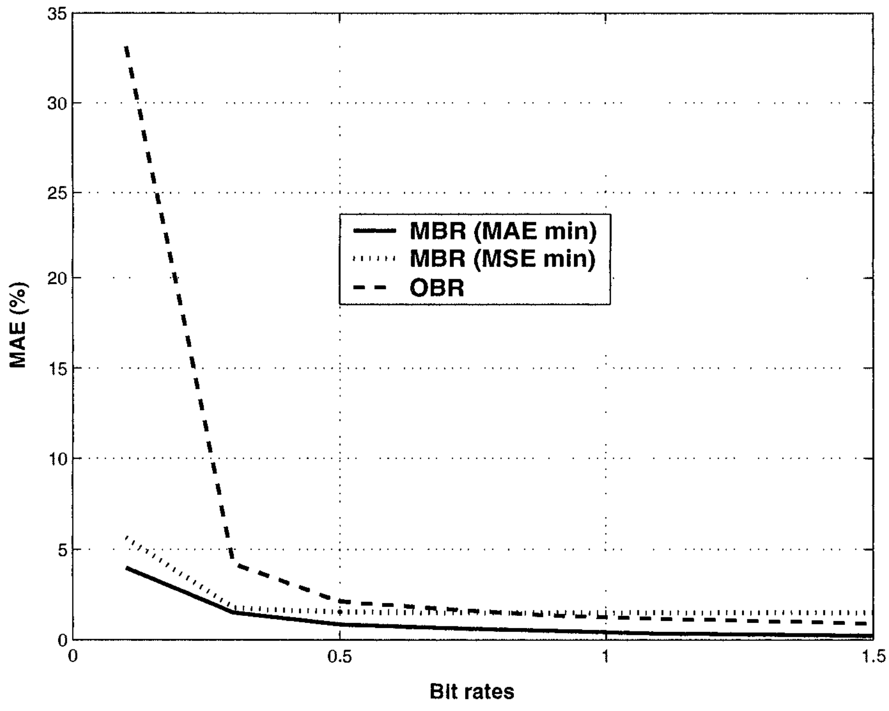


Fig. 1. Dependence of the Maximum Absolute Error Δ on the average bitrate $\bar{b}$ for different compression techniques

Acknowledgement. This chapter was partially supported by NSF Grant CDA-9522207, the ARO grants DAA H04-95-1-0494 and DAA D19-99-1-0012, by the Texas Instruments Grant, and by NASA under cooperative agreement NCC5-209. This chapter is partly based on research performed during O. Kosheleva's visit to Brazil; this visit was sponsored by the Brazilian funding agency CTINFO/CNPq.

The author is thankful to the editors and to the anonymous referees for valuable suggestions.

A The Relation between Interval Computations and the Ideas of Underestimators and Overestimators

The idea of an overestimator is very similar to the experience of interval computations; see, e.g., [JKDW01].

In many areas of science and engineering, we are interested in the value of a physical quantity y that is difficult (or even impossible) to measure directly. To measure such quantities, we find auxiliary easier-to-measure quantities $x_1, \ldots, x_n$ that are related to y by a known algorithm $y = g(x_1, \ldots, x_n)$ – an algorithm that, in general, computes a complex functions with a large number of variables. Then, we measure x_i, and apply the algorithm f to the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of measuring x_i. As a result, we get an estimate $\widetilde{y} = g(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for y.

Since the measured values $\widetilde{x}_i$ are, in general, slightly different from the actual values x_i, a natural question is: what is the error of the resulting estimate?

In many practical situations, the only information that we have about the measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ of direct measurements is the upper bounds Δ_i on $|\Delta x_i|$ guaranteed by the manufacturer of the measuring instrument. In such situations, the only information that we have about the actual value x_i is that x_i lies in the interval $\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this case, the only information that we have about y is that y belongs to the range

$$\mathbf{y} = g(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{g(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1 \,\&\, \ldots \,\&\, x_n \in \mathbf{x}_n\}.$$

It is known that computing this range exactly is an NP-hard problem; see, e.g., [KLRK97]. Crudely speaking, NP-hard means that we cannot have an algorithm that always finished in reasonable time and that always produces the exact range.

The objective of interval computation is find *guaranteed* bounds for the actual value of y. Since we cannot find the exact range $\mathbf{y}$, researchers in interval computations design algorithms that provide us with an *enclosure* $\mathbf{Y} \supseteq \mathbf{y}$ for the actual range. The lower bound of this enclosure is a (guaranteed) underestimator, and the upper of this enclosure is an overestimator.

References

[Cab02] Cabrera, S.D.: Three-dimensional compression of mesoscale meteorological data based on JPEG2000. In: Battlespace Digitization and Network-Centric Warfare II, Proc. SPIE, **4741**, 239–250 (2002)

[Flo00] Floudas, C.A.: Deterministic Global Optimization: Theory, Methods, and Applications. Kluwer, Dordrecht (2000)

[JKDW01] Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics. Springer, London (2001)

[KLRK97] Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht (1997)

[Kos03] Kosheleva, O.M.: Task-Specific Metrics and Optimized Rate Allocation Applied to Part 2 of JPEG2000 and 3-D Meteorological Data. Ph.D. Dissertation, University of Texas at El Paso (2003)

[KUCV04] Kosheleva, O.M., Usevitch, B., Cabrera, S., Vidal, E.Jr.: MSE optimal bit allocation in the application of JPEG2000 Part 2 to meteorological data. In: Proceedings of the 2004 IEEE Data Compression Conference DCC'2004, Snowbird, Utah, March 2004, 546 (2004)

[KUV05] Kosheleva, O.M., Usevitch, B., Vidal, E.Jr.: Compressing 3D measurement data under interval uncertainty. In: Dongarra, J., Madsen, K., Wasniewski, J. (eds) PARA'04 Workshop on State-of-the-Art in Scientific Computing. Springer Lecture Notes in Computer Science, **3732**, 142–150 (2005)

[MF98] Mallat, S., Falzon, F.: Analysis of low bit rate image transform coding. IEEE Trans. Signal Proc., **46**, 1027–1042 (1998)

[Moo79] Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)

[TM02] Taubman, D.S., Marcellin, M.W.: JPEG2000 Image Compression Fundamentals, Standards and Practice. Kluwer, Boston, Dordrecht, London (2002)

[Vav91] Vavasis, S.A.: Nonlinear Optimization: Complexity Issues. Oxford University Press, New York (1991)

An Interval Partitioning Approach for Continuous Constrained Optimization

Chandra Sekhar Pedamallu[1], Linet Özdamar[2], and Tibor Csendes[3]

[1] Nanyang Technological University, School of Mechanical and Aerospace Engineering, Singapore chandra@inf.u-szeged.hu
[2] Izmir Ekonomi Universitesi, Izmir, Turkey linetozdamar@lycos.com
[3] University of Szeged, Institute of Informatics, Szeged, Hungary csendes@inf.u-szeged.hu

Summary. Constrained Optimization Problems (COP's) are encountered in many scientific fields concerned with industrial applications such as kinematics, chemical process optimization, molecular design, etc. When non-linear relationships among variables are defined by problem constraints resulting in non-convex feasible sets, the problem of identifying feasible solutions may become very hard. Consequently, finding the location of the global optimum in the COP is more difficult as compared to bound-constrained global optimization problems.

This chapter proposes a new interval partitioning method for solving the COP. The proposed approach involves a new subdivision direction selection method as well as an adaptive search tree framework where nodes (boxes defining different variable domains) are explored using a restricted hybrid depth-first and best-first branching strategy. This hybrid approach is also used for activating local search in boxes with the aim of identifying different feasible stationary points. The proposed search tree management approach improves the convergence speed of the interval partitioning method that is also supported by the new parallel subdivision direction selection rule (used in selecting the variables to be partitioned in a given box). This rule targets directly the uncertainty degrees of constraints (with respect to feasibility) and the uncertainty degree of the objective function (with respect to optimality). Reducing these uncertainties as such results in the early and reliable detection of infeasible and sub-optimal boxes, thereby diminishing the number of boxes to be assessed. Consequently, chances of identifying local stationary points during the early stages of the search increase.

The effectiveness of the proposed interval partitioning algorithm is illustrated on several practical application problems and compared with professional commercial local and global solvers. Empirical results show that the presented new approach is as good as available COP solvers.

Key words: continuous constrained optimization, interval partitioning approach, practical applications.

1 Introduction

Many important real world problems can be expressed in terms of a set of nonlinear constraints that restrict the domain over which a given performance criterion is optimized, that is, as a Constrained Optimization Problem (COP). In the general COP with a non-convex objective function, discovering the location of the global optimum is NP-hard. Locating feasible solutions in a non-convex feasible space is also NP-hard. Solution approaches using derivatives developed for solving the COP might often be trapped in infeasible and/or sub-optimal sub-spaces if the combined topology of the constraints is too rugged. The same problem exists in the discovery of global optima in non-convex bound-constrained global optimization problems. The COP has augmented complexity as compared to bound-constrained problems due to the restrictions imposed by highly non-linear relationships among variables.

Existing global optimization algorithms can be categorized into deterministic and stochastic methods. Extensive surveys on global optimization exist in the literature ([TZ89], and recently by [PR02]). Although we cannot cover the COP literature in detail within the scope of this chapter, we can cite deterministic approaches including Lipschitzian methods [HJL92, Pin97]; branch and bound methods (e.g., [AS00]); cutting plane methods [TTT85]; outer approximation [HTD92]; primal-dual method [BEG94, FV93]; alpha-Branch and Bound approach [AMF95], reformulation techniques [SP99]; interior point methods [MNW$^+$01, FGW02] and interval methods [CR97, Han92, Kea96c].

Interval Partitioning methods (*IP*) are Branch and Bound techniques (B&B) that use inclusion functions, therefore, we elaborate more on B&B among deterministic methods. B&B are partitioning algorithms that are complete and reliable in the sense that they explore the whole feasible domain and discard sub-spaces in the feasible domain only if they are guaranteed to exclude feasible solutions and/or local stationary points better than the ones already found. B&B are exhaustive algorithms that typically rely on generating lower and upper bounds for boxes in the search tree, where tighter bounds may result in early pruning of nodes. For expediting B&B, feasibility and optimality based variable range reduction techniques [RS95, RS96], convexification [TS02, TS04], outer approximation [BHR92] and constraint programming techniques in pre- and post-processing phases of branching have been developed [RS96]. The latter resulted in an advanced methodology and software called Branch and Reduce algorithm (BARON, [Sah96, Sah03]).

Here, we propose an interval partitioning approach that recursively subdivides the continuous domain over which the COP is defined. This *IP* conducts reliable assessment of sub-domains while searching for the globally optimal solution. Theoretically, *IP* has no difficulties in dealing with the COP, however, interval research on the COP is relatively scarce when compared with bound constrained optimization. Robinson [Rob73] uses interval arithmetic only to obtain bounds for the solution of the COP, but does not attempt to find the global optimum. Hansen and Sengupta [HS80] first use *IP* to solve the inequality COP. A detailed discussion on interval techniques for the general COP with both inequality and equality constraints is provided in [RR88] and [Han92], and some numerical results using these techniques have been published later [Wol94, Kea96a]. An alternative approach is presented in [ZB03] for providing ranges on functions.

Conn et al.[CGT94] transform inequality constraints into a combination of equality constraints and bound constraints and combine the latter with a procedure for handling bound constraints with reduced gradients. Computational examination of feasibility verification and the issue of obtaining rigorous upper bounds are discussed in [Kea94] where the interval Newton method is used for this purpose. In [HW93], interval Newton methods are applied to the Fritz John equations that are used to reduce the size of sub-spaces in the search domain without bisection or other tessellation. Experiments that compare methods of handling bound constraints and methods for normalizing Lagrange multipliers are conducted in [Kea96b]. Dallwig et al. [DNS97] propose software (so called GLOPT) for solving bound constrained optimization and the COP. GLOPT uses a Branch and Bound technique to split the problem recursively into subproblems that are either eliminated or reduced in size. The authors also propose a new reduction technique for boxes and novel techniques for generating feasible points. Kearfott presented GlobSol (e.g. in [Kea03]), which is an *IP* software that is capable of solving bound constrained optimization problems and the COP.

A new *IP* is developed by Markót [Mar03] for solving COP problems with inequalities where new adaptive multi-section rules and a box selection criterion are presented [MFCC05]. Kearfott [Kea04] provides a discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. Empirical results show that linear relaxations are of significant value in validated global optimization. Finally, in order to eliminate the subregion of the search spaces, Kearfott [Kea05] proposes a simplified and improved technique for validation of feasible points in boxes, based on orthogonal decomposition of the normal space to the constraints. In the COP with inequalities, a point, rather than a region, can be used, and for the COP with both equalities and inequalities, the region lies in a smaller-dimensional subspace, giving rise to sharper upper bounds.

In this chapter, we propose a new adaptive tree search method that we used in *IP* to enhance its convergence speed. This approach is generic and it can also be used in non-interval B&B approaches. We also develop a new subdivision direction selection rule for IP. This rule aims at reducing the uncertainty degree in the feasibility of constraints over a given sub-domain as well as the uncertainty in the box's potential for containing the global optimum. We show, on a test bed of practical applications, that the resulting *IP* is a viable method in solving the general COP with equalities and inequalities. The results obtained are compared with commercial softwares such as BARON, Minos and other solvers interfaced with GAMS (`www.gams.com`).

2 Interval Partitioning Algorithm for the COP

2.1 Problem Definition

A COP is defined by an objective function, $f(x_1, \ldots, x_n)$ to be maximized over a set of variables, $V = \{x_1, \ldots, x_n\}$, with finite continuous domains: $X_i = [\underline{X}, \overline{X}]$ for x_i, $i = 1, \ldots, n$, that are restricted by a set of constraints, $C = \{c_1, \ldots, c_r\}$.

Constraints in C are linear or nonlinear equations or inequalities that are represented as follows:

$$g_i(x_1, \ldots, x_n) \leq 0 \quad i = 1, \ldots, k,$$

$$h_i(x_1, \ldots, x_n) = 0 \quad i = k+1, \ldots, r.$$

An optimal solution of a COP is an element x^* of the search space X ($X = X_1 \times \cdots \times X_n$) that meets all the constraints, and whose objective function value, $f(x^*) \geq f(x)$ for all feasible elements $x \in X$.

COP problems are difficult to solve because the only way parts of the search space can be discarded is by proving that they do not contain an optimal solution. It is hard to tackle general nonlinear COP with computer algebra systems, and in general, traditional numeric algorithms cannot guarantee global optimality and completeness in the sense that the solution found may be only a local optimum. It is also possible that the approximate search might result with an infeasible result despite the fact that a global optimum exists. Here, we propose an Interval Partitioning Algorithm (IP) to identify x^* in a reliable manner.

2.2 Basics of Interval Arithmetic and Terminology

Denote the real numbers by $x, y, \ldots$, the set of compact intervals by $\mathbb{I} := \{[a, b] \mid a \leq b,\ a, b \in \mathbb{R}\}$, and the set of n-dimensional intervals (also called simply intervals or boxes) by $\mathbb{I}^n$. Capital letters will be used for intervals. Every interval $X \in \mathbb{I}$ is denoted by $[\underline{X}, \overline{X}]$, where its bounds are defined by $\underline{X} = \inf X$ and $\overline{X} = \sup X$. For every $a \in \mathbb{R}$, the interval point $[a, a]$ is also denoted by a. The width of an interval X is the real number $w(\mathrm{X}) = \underline{\mathrm{X}} - \overline{\mathrm{X}}$. Given two intervals X and Y, X is said to be *tighter than* Y if $w(X) < w(Y)$.

Given $(X_1, \ldots, X_n) \in \mathbb{I}$, the corresponding *box* X is the Cartesian product of intervals, $X = \mathrm{X}_1 \times \ldots \times \mathrm{X}_n$, where $\mathrm{X} \in \mathbb{I}^n$. A subset of X, $Y \subseteq X$, is a sub-box of X. The notion of *width* is defined as follows: $w(X_1 \times \ldots \times X_n) = \max_{1 \leq i \leq n} \quad w(X_i)$, and $w(X_i) = \overline{X_i} - \underline{X_i}$.

Interval arithmetic operations are set theoretic extensions of the corresponding real operations. Given X, Y$\in \mathbb{I}$, and an operation $\omega \in \{+, -, \cdot, \div\}$, we have: $X \omega Y = \{x \omega y \mid x \in X, y \in Y\}$.

Due to properties of monotonicity, these operations can be implemented by real computations over the bounds of intervals. Given two intervals $X = [a, b]$ and $Y = [c, d]$

$$
\begin{aligned}
[a,b] + [c,d] &= [a+c, b+d] \\
[a,b] - [c,d] &= [a-d, b-c] \\
[a,b] \cdot [c,d] &= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\
[a,b] \;/\; [c,d] &= [a,b] \cdot [1/d, 1/c] \text{ if } 0 \notin [c,d].
\end{aligned}
$$

The associative law and the commutative law are preserved over it. However, the distributive law does not hold. In general, only a weaker law is verified, called subdistributivity.

Interval arithmetic is particularly appropriate to represent outer approximations of real quantities. The range of a real function f over an interval X is denoted by $f(X)$, and it can be computed by interval extensions.

Definition 1. *(Interval extension): An interval extension of a real function $f : D_f \subset \mathbb{R}^n \to \mathbb{R}$ is a function $F : \mathbb{I}^n \to \mathbb{I}$ such that $\forall X \in \mathbb{I}^n, X \in D_f \Rightarrow f(X) = \{f(x) \mid x \in X\} \subseteq F(X)$.*

Interval extensions are also called *interval forms* or *inclusion functions.* This definition implies the existence of infinitely many interval extensions of a given real function. In a proper implementation of interval extension based inclusion functions the outward rounding must be made to be able to provide a mathematical strength reliability.

The most common extension is known as the natural extension. It means that procedure when we substitute each occurrence of variables, operations and functions by their interval equivalents [RR88]. Natural extensions are inclusion monotonic (this property follows from the monotonicity of operations). Hence, given a real function f, whose natural extension is denoted by F, and two intervals X and Y such that $X \subseteq Y$, the following holds: $F(X) \subseteq F(Y)$. We denote the lower and upper bounds of the function interval range over a given box Y as $\underline{F}(Y)$ and $\overline{F}(Y)$, respectively.

Here, it is assumed that for the studied COP, the natural interval extensions of f, g and h over X are defined in the real domain. Furthermore, F (and similarly, G and H) are α-convergent over X, that is, for all $Y \subseteq X$, $w(F(Y)) - w(f(Y)) \leq cw(Y)^{\alpha}$, where c and α are positive constants.

An *interval constraint* is built from an interval function and a relation symbol, which is extended to intervals. A constraint being defined by its expression (atomic formula and relation symbols), its variables, and their domains, we will consider that an interval constraint has interval variables (variables that take interval values), and that each associated domain is an interval.

The main guarantee of interval constraints is that if its solution set is empty, it has no solution over a given box Y, then it follows that the solution set of the COP is also empty and the box Y can be reliably discarded. In a similar manner, if the upper bound of the objective function range, $\overline{F}(Y)$, over a given box Y is less than or equal to the objective function value of a known feasible solution, (the Current Lower Bound, CLB) then Y can be reliably discarded since it cannot contain a better solution than the CLB.

Below we formally provide the conditions where a given box Y can be discarded reliably based on the ranges of interval constraints and the objective function.

In a partitioning algorithm, each box Y is assessed for its optimality and feasibility status by calculating the ranges for F, G, and H over the domain of Y.

Definition 2. *(Cut-off test based on optimality:) If $\overline{F}(Y) < CLB$, then box Y is called a sub-optimal box.*

Definition 3. *(Cut-off test based on feasibility:) If $\underline{G}_i(Y) > 0$, or $0 \notin H_i(Y)$ for any i, then box Y is called an infeasible box.*

Definition 4. *If $\underline{F}(Y) \leq CLB$, and $\overline{F}(Y) > CLB$, then Y is called an indeterminate box with regard to optimality. Such a box holds the potential of containing x^* if it is not an infeasible box.*

Definition 5. *If $(\underline{G}_i(Y) < 0$, and $\overline{G}_i(Y) > 0)$, or $(0 \in H_i(Y) \neq 0)$ for some i, and other constraints are consistent over Y, then Y is called an indeterminate box with regard to feasibility and it holds the potential of containing x^* if it is not a sub-optimal box.*

Definition 6. *The degree of uncertainty of an indeterminate box with respect to optimality is defined as: $PF_Y = \overline{F}(Y) - CLB$.*

Definition 7. *The degree of uncertainty, PG_Y^i (PH_Y^i) of an indeterminate inequality (equality) constraint with regard to feasibility is as: $PG_Y^i = \overline{G_i}(Y)$, and $PH_Y^i = \overline{H_i}(Y) + |\underline{H}_i(Y)|$.*

Definition 8. *The total feasibility uncertainty degree of a box, INF_Y, is the sum of uncertainty degrees of equalities and inequalities that are indeterminate over Y.*

The proposed subdivision direction selection rule (Interval Inference Rule, *IIR*) targets an immediate reduction in INF_Y and PF_Y and chooses those specific variables to bisect a given parent box. The IP described in the following section uses the feasibility and optimality cut-off tests in discarding boxes and applies the new rule IIR in partitioning boxes.

2.3 Interval Partitioning Algorithm

Under reasonable assumptions, IP is a reliable convergent algorithm that subdivides indeterminate boxes to reduce INF_Y and PF_Y by nested partitioning. In terms of subdivision direction selection, convergence depends on whether the direction selection rule is balanced [CR97]. The contraction and the α-convergence properties enable this. The reduction in the uncertainty levels of boxes finally lead to their elimination due to sub-optimality or infeasibility while helping IP in ranking remaining boxes in a better fashion.

A box that becomes feasible after nested partitioning still can have uncertainty with regard to optimality unless it is proven that it is sub-optimal. The convergence rate of IP might be very slow if we require nested partitioning to reduce a box to a point interval that is to the global optimum. Hence, since a box with a high PF_Y holds the promise of containing the global optimum, we propose to use a local search procedure that can identify stationary points in such boxes.

Usually, IP continues to subdivide available indeterminate and feasible boxes until either they are all deleted or interval sizes of all variables in existing boxes are less than a given tolerance. Termination can also be forced by limiting the number of function evaluations and/or CPU time. In the following, we describe our proposed IP that has a flexible stage-wise tree management feature. Our IP terminates if the CLB does not improve at the end of a tree stage as compared with the previous stage. This stage-wise tree also enables us to apply the best-first box selection rule within a restricted subtree (economizing memory usage) as well as to invoke local search in a set of boxes.

The tree management system in the proposed IP maintains a stage-wise branching scheme that is conceptually similar to the iterative deepening approach [Kor85]. The iterative deepening approach explores all nodes generated at a given tree level (stage) before it starts assessing the nodes at the next stage. Exploration of boxes at the same stage can be done in any order, the sweep may start from best-first box or the one on the most right or most left of that stage. On the other hand, in the proposed adaptive tree management system, a node (parent box) at the current stage is permitted to grow a sub-tree forming partial succeeding tree levels and to explore nodes in this sub-tree before exhausting the nodes at the current stage.

If a feasible solution (and CLB) is not identified yet, boxes in the subtree are ranked according to descending INF_Y, otherwise they are ranked in descending order of $\overline{F}(Y)$. A box is selected among the children of the same parent according to either box selection criterion, and the child box is partitioned again continuing to build the same sub-tree. This sub-tree grows until the Total Area Deleted (TAD) by discarding boxes fails to improve in two consecutive partitioning iterations in this sub-tree. Such failure triggers a call to local search where all boxes not previously subjected to local search are processed by the procedure Feasible Sequential Quadratic Programming (FSQP, [ZT96, LZT97]), after which they are placed back in the list of pending boxes and exploration is resumed among the nodes at the current stage. Feasible and improving solutions found by FSQP are stored (that is, if a feasible solution with a better objective function value is found, CLB is updated and the solution is stored).

The above adaptive tree management scheme is achieved by maintaining two lists of boxes, B_s and B_{s+1} that are the lists of boxes to be explored at the current stage s and the next stage $s+1$, respectively. Initially, the set of indeterminate or feasible boxes in the pending list B_s consists only of X and B_{s+1} is empty. As child boxes are added to a selected parent box, they are ordered according to the current ranking criterion. Boxes in the sub-tree stemming from the selected parent at the current stage are explored and partitioned until there is no improvement in TAD in two consecutive partitioning iterations.

At that point, partitioning of the selected parent box is stopped and all boxes that have not been processed by local search are sent to the FSQP module and processed to identify feasible and improving point solutions if FSQP is successful in doing so. From that moment onwards, child boxes generated from any other selected parent in B_s are stored in B_{s+1} irrespective of further calls to FSQP in the current stage. When all boxes in B_s have been assessed (discarded or partitioned), the search moves to the next stage, *s+1*, starting to explore the boxes stored in B_{s+1}.

In this manner, a lesser number of boxes (those in the current stage) are maintained in primary memory and the search is allowed to go down to deeper levels within the same stage, increasing the chances to discard boxes. On the other hand, by enabling the search to also explore boxes horizontally across at the current stage, it might be possible to find feasible improving solutions faster by not partitioning parent boxes that are not so promising (because we are able to observe a larger number of boxes).

The tree continues to grow in this manner taking up the list of boxes of the next stage after the current stage's list of boxes is exhausted. The algorithm stops either when there are no boxes remaining in B_s and B_{s+1} or when there is no improvement in CLB as compared with the previous stage. The proposed IP algorithm is described below.

IP with adaptive tree management

```
Step 0. Set tree stage, $s = 1$. Set future stage, $r = 1$. Set non-improvement
    counter for $TAD$: $nc = 0$. Set $B_s$, the list of pending boxes at stage $s$
    equal to $X$, $B_s = \{X\}$, and $B_{s+1} = \emptyset$.
Step 1. If $B_s = \emptyset$ and $CLB$ has not improved as compared to the stage $s-1$,
    or, both $B_s = \emptyset$ and $B_{s+1} = \emptyset$, then STOP.
    Else, if $B_s = \emptyset$ and $B_{s+1} \neq \emptyset$, then set $s \leftarrow s+1$, set $r \leftarrow s$, and continue.
    Pick the first box $Y$ in $B_s$ and continue.
    1.1 If $Y$ is infeasible or suboptimal, discard $Y$, and go to Step 1.
    1.2 If $Y$ is sufficiently small, evaluate $m$, its mid-point, and if it is a feasible
    improving solution, update $CLB$, reset $nc \leftarrow 0$, and store $m$. Remove $Y$
    from $B_s$ and go to Step 1.
Step 2. Select variable(s) to partition (use the subdivision direction selection
    rule $IIR$). Set $v$ = number of variables to partition.
Step 3. Partition $Y$ into $2^v$ non-overlapping child boxes. Check $TAD$, if it
    improves, then reset $nc \leftarrow 0$, else set $nc \leftarrow nc+1$.
Step 4. Remove $Y$ from $B_s$, add $2^v$ boxes to $B_r$.
```

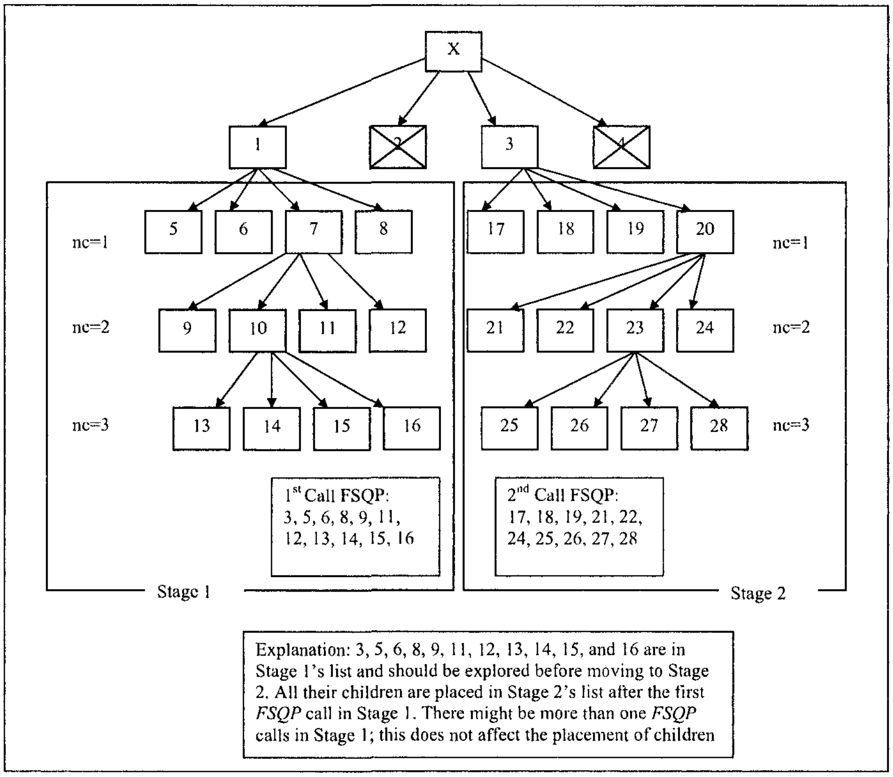

X
1
2
3
4
nc=1
5
6
7
8
17
18
19
20
nc=1
nc=2
9
10
11
12
21
22
23
24
nc=2
nc=3
13
14
15
16
25
26
27
28
nc=3
1st Call FSQP: 3, 5, 6, 8, 9, 11, 12, 13, 14, 15, 16
2nd Call FSQP: 17, 18, 19, 21, 22, 24, 25, 26, 27, 28
Stage 1
Stage 2
Explanation: 3, 5, 6, 8, 9, 11, 12, 13, 14, 15, and 16 are in Stage 1's list and should be explored before moving to Stage 2. All their children are placed in Stage 2's list after the first *FSQP* call in Stage 1. There might be more than one *FSQP* calls in Stage 1; this does not affect the placement of children

Fig. 1. Implementation of the adaptive iterative deepening procedure

4.1. If $nc > 2$, apply FSQP to all (previously unprocessed by FSQP) boxes in B_s and B_{s+1}, reset $nc \leftarrow 0$. If FSQP is called for the first time in stage s, then set $r \leftarrow s + 1$. Go to Step 1.

4.2. Else, go to Step 1. The adaptive tree management system in IP is illustrated in Fig. 1 on a small tree where node labels indicate the order of nodes visited.

The adaptive tree management system in IP is illustrated in Fig. 1 on a small tree where node labels indicate the order of nodes visited.

2.4 A New Subdivision Direction Selection Rule for *IP*

The order in which variable domains are partitioned has an impact on the convergence rate of IP. In general, variable selection is made according to widest variable domain rule or largest function rate of change in the box. Here, we develop a new numerical subdivision direction selection rule, Interval Inference Rule (IIR), to improve IP's convergence rate by partitioning in parallel, those variable domains that reduce PF_Y and INF_Y in immediate child boxes. Hence, new boxes are formed with an appropriate partitioning sequence resulting in diminished uncertainty caused by the overestimation in the indeterminate objective function range and constraint ranges.

Before *IIR* is applied, the objective f and each constraint g and h are interpreted as binary trees that represent recursive sub-expressions hierarchically. Such binary trees enable interval propagation over all sub-expressions of the constraints and the objective function [BMV94]. Interval propagation and function trees are used by [Kea91] in improving interval Newton approach by decomposition and variable expansion, by [SP99] in automated problem reformulation, by [Sah03] and by [TS04] where feasibility based range reduction is achieved by tightening variable bounds.

After interval propagation is carried out over the sub-expressions in a binary tree, *IIR* traverses this tree to label its nodes so as to identify the pair of variables (source variables) that are most influential on the constraint's or the objective's uncertainty degree. The presented interval subdivision direction selection rule is an alternative of earlier rules as those published in [CR97], [RC95], and [CGC00]. This pair of variables are identified for each constraint and the objective function, and placed in the pool of variables whose domains will be possibly partitioned in the next iteration. We make sure that the pool at least contains the source variables for the objective function and therefore, the number of variables to be bisected in parallel is at least two. The total pool resulting from the traversal of f, g and h is screened and its size is reduced by allocating weights to variables and re-assessing them.

2.4.1 Interval Partitioning Algorithm

Before the *labeling* process *IIR_Tree* can be applied on a constraint expression, it has to be parsed and intervals have to be propagated through all sub-expression levels. This is achieved by calling an Interval Library at each (molecular) sub-expression level of the binary tree from bottom to top starting from atomic levels (variables or constants).

A binary tree representing a constraint is built as follows. Leaves of the binary tree are atomic elements, i.e., they are either variables or constants. All other nodes represent binary expressions of the form (Left Θ Right). A binary operator "Θ" is an arithmetic operator ($\cdot$, $+$, $-$, $\div$) having two branches ("Left", "Right") that are themselves recursive binary sub-trees. However, mathematical functions such as ln, exp, sin, etc. are unary operators. In such cases, the argument of the function is always placed in the "Left" branch. For instance, the binary tree for the expression $1 - (10x_1 + 6x_1x_2 - 6x_3x_4) = 0$ is illustrated in Fig. 2.

Variable intervals in the box are $x_1 = [-2.0, 4.0]$, $x_2 = [0.0, 10.0]$, $x_3 = [-2.0, 1.0]$, and $x_4 = [-10.0, 0.0]$. In Fig. 2, dotted arrows linking arguments with operator nodes show how intervals are propagated starting from the bottom leaves (variables). Once a level of the tree is completed and the

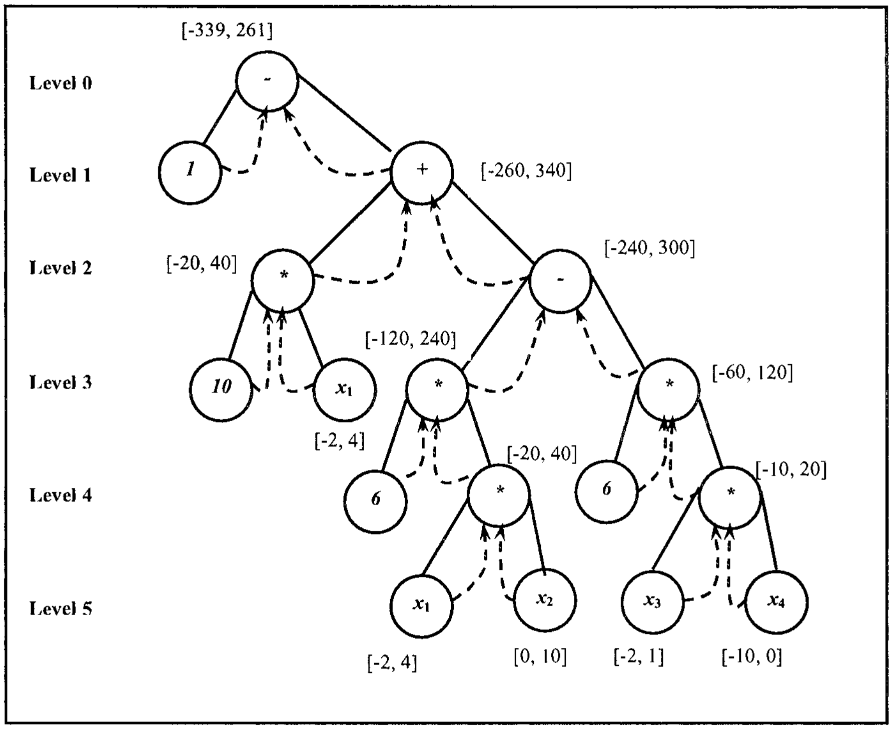


Fig. 2. Interval propagation for the expression $1 - (10x_1 + 6x_1x_2 - 6x_3x_4) = 0$

corresponding sub-expression intervals are calculated according to basic interval operations, they are linked by next level operators. This procedure goes on until the topmost "root" node representing the whole constraint is reached resulting in the constraint range of [-339, 261].

We now describe the *IIR_Tree* labeling procedure. Suppose a binary tree is constructed for a constraint and its source variables have to be identified over a given box Y. The labeling procedure called *IIR*_Tree accomplishes this by tree traversal. We take the expression depicted in Fig. 2 as an indeterminate equality constraint. In Fig. 3, the path constructed by *IIR*_Tree on this example is illustrated graphically. Straight lines in the figure indicate the propagation tree, dashed arrows indicate binary decisions, and arrows with curvature indicate the path constructed by *IIR_Tree*.

For illustrating how *IIR*_Tree works on a given constraint or objective function over domain Y, we introduce the following notation.
Q^k: a parent sub-expression at tree level k ($k = 0$ is root node),
L^{k+1} and R^{k+1}: immediate Left and Right sub-expressions of Q^k at level $k+1$,
$[\underline{Q}^k, \overline{Q}^k]$: interval bounds of the parent sub-expression Q^k,

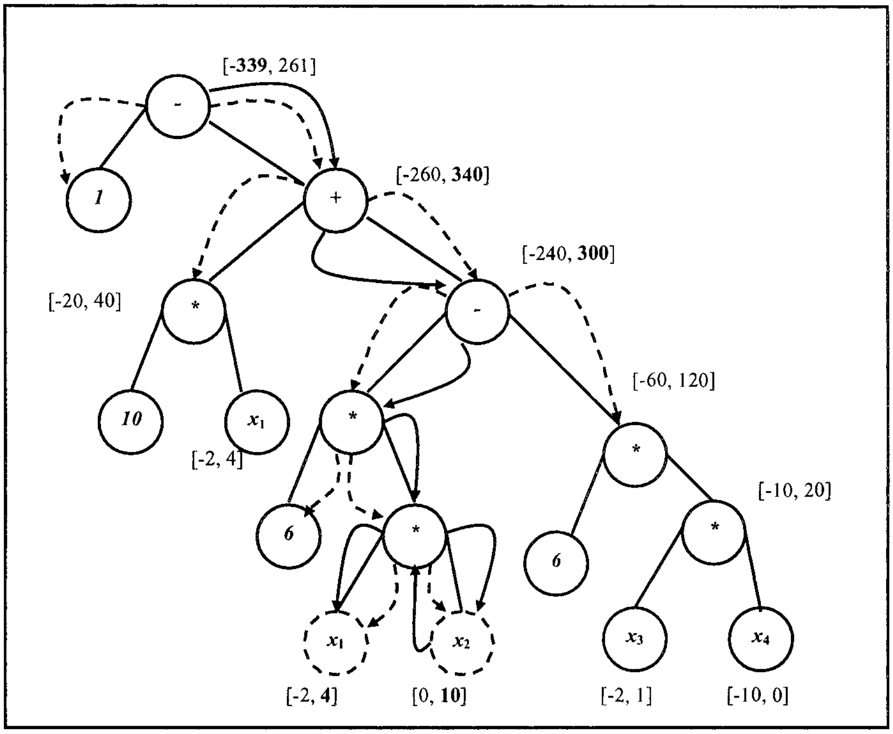


Fig. 3. Implementation of *IIR_Tree* over the binary tree for $1 - (10x_1 + 6x_1x_2 - 6x_3x_4) = 0$

$[\underline{L}^{k+1}, \overline{L}^{k+1}]$ and $[\underline{R}^{k+1}, \overline{R}^{k+1}]$: interval bounds of immediate left and right sub-expressions,

L^k: labeled bound at level k.

IIR starts by labeling $\overline{Q}^0$ when the constraint is an inequality. Hence, the target is $\overline{G}_i(Y)$ for inequalities so as to reduce PG^i_Y, and in equalities the target is the $\max\{\underline{Q}^0, \overline{Q}^0\}$. That is, $\max\{|\underline{H}_i(Y)|, \overline{H}_i(Y)\}$ is targeted to reduce PH^i_Y. If the expression concerns the objective function f, then *IIR*_Tree labels $\overline{F}(Y)$ at the root node in order to minimize PF_Y.

Here, we have an equality (in Fig. 2) and hence, *IIR* labels $\underline{Q}^0$. That is, we label the bound that gives max {|-339|, |261|}, which is -339, as Λ^0 at the root node. Next, we determine the pair of interval bounds $\{\underline{L}^1 - \overline{R}^1\}$ that results in -339. Hence, $\underline{L}^1 \Theta \overline{R}^1 = \underline{Q}^0$. We then compare the absolute values of individual bounds in this pair and take their maximum as the label at level $k+1$. That is, $\Lambda^1 = \max\{|\underline{L}^1|, |\overline{R}^1|\} = \overline{R}^1 = 340$.

The procedure is applied recursively from top to bottom; each time searching for the bound pair resulting in the labeled bound Λ^{k+1} till a leaf (a variable) is hit. Once this forward tree traversal is over, all leaves in the tree corresponding to the selected variable are set to "Closed" status. The procedure then backtracks to the next higher level of the tree to identify the other leaf in the couple of variables that produce the labeled bound. All steps of the labeling procedure carried out in the example are provided below in detail.

Level 0: $[\underline{Q}^0, \overline{Q}^0] = [-339, 261]$. $\Lambda^0 = \underline{Q}^0$.

$a\Theta b$ = $\underline{L}^1$ - $\overline{R}^1$= {1-340} = -339. Λ^1 = max{$|\underline{L}^1|$, $|\overline{R}^1|$} = max{|1|, |340|} = 340 = $\overline{R}^1$.

Level 1: $[\underline{Q}^1, \overline{Q}^1] = [-260, 340]$

$a\Theta b = \{-20 + (-240) \text{ or } 40 + 300\} = 340 \Rightarrow a\Theta b = \overline{L}^2 + \bar{R}^2$. $\Lambda^2 = \max\{|\overline{L}^2|, |\overline{R}^2|\} = \max\{|40|, |300|\} = 300 \Rightarrow \overline{R}^2$.

Level 2: $[\underline{Q}^2, \overline{Q}^2] = [-240, 300]$

$a\Theta b = \{(-120) - 120 \text{ or } 240 - (-60)\} = 300 \Rightarrow a\Theta b = \overline{L}^3 - \underline{R}^3$. $\Lambda^3 = \max\{|\overline{L}^3|, |\underline{R}^3|\} = \max\{|240|, |-60|\} = 240 \Rightarrow \overline{L}^3$.

Level 3: $[\underline{Q}^3, \overline{Q}^3] = [-120, 240]$

$a\Theta b = \{6 * (-20) \text{ or } 6 * 40\} = 240 \Rightarrow a\Theta b = \overline{L}^4 * \overline{R}^4$. $\Lambda^4 = \max\{|\overline{L}^4|, |\overline{R}^4|\} = \max\{|6|, |40|\} = 40 \Rightarrow \overline{R}^4$.

Level 4: $[\underline{Q}^5, \overline{Q}^5] = [-20, 40]$

$a\Theta b = \{-2 * 0 \text{ or } -2 * 10 \text{ or } 4 * 0 \text{ or } 4 * 10\} = 40 \Rightarrow a\Theta b = \overline{L}^5 * \overline{R}^5$. $\Lambda^5 = \max\{|\overline{L}^5|, |\overline{R}^5|\} = \max\{|4|, |10|\} = 10 \Rightarrow \overline{R}^5$.

The bound $\overline{R}^5$ leads to leaf x_2. The leaf pertaining to x_2 is "Closed" from here onwards, and the procedure backtracks to Level 4. Then, the labeling procedure leads to the second source variable, x_1.

Note that the uncertainty degree of the parent box is 600 whereas when it is sub-divided into four sibling boxes by bisecting the two source variables, the uncertainty degrees of sibling boxes become 300, 330, 420, and 390. If the parent box were sub-divided using the largest width variable rule (x_2 and x_4), then the sibling uncertainty degrees would have been 510, 600, 330, and 420.

2.4.2 Restricting Parallelism in Multi-variable Partitioning

When the number of constraints is large, there might be a large set of variables resulting from the local application of *IIR*_Tree to the objective function and each constraint. Here, we develop a priority allocation scheme to narrow down the set of variables (selected by *IIR*) to be partitioned in parallel. In this approach, all variable pairs identified by *IIR*_Tree are merged into a single set Z. Then, a weight w_j is assigned to each variable $x_j \in Z$ and the average $\bar{w}$ is calculated. The final set of variables to be partitioned is composed of the two source variables of f and all other source variables $x_j \in Z$ with $w_j > \overline{w}$ pertaining to the constraints.

Then w_j is defined as a function of several criteria: PG_Y^i (or PH_Y^i) of constraint g_i for which x_j is identified as a source variable, the number of times x_j exists in g_i, total number of multiplicative terms in which x_j is involved within g_i. Furthermore, the existence of x_j in a trigonometric and/or even power sub-expression in g_i is included in w_j by inserting corresponding flag variables. When a variable x_j is a source variable to more than one constraint, the weight calculated for each such constraint is added to result in a total w_j defined as

$$w_j = \sum_{i \in IC_j} [PF_Y^i/PH_{\max} + PG_Y^i/PG_{\max} + e_{ji}/e_{j,\max} + a_{ji}/a_{j,\max} + t_{ji} + p_{ji}]/5$$

where

IC_j:	set of indeterminate constraints (over Y) where x_j is a source variable,
TIC:	total set of indeterminate constraints,
PH_{max}:	$\max_{i \in TIC} \{PH_Y^i\}$,
PG_{max}:	$\max_{i \in TIC} \{PG_Y^i\}$,
e_{ji}:	number of times x_j exists in constraint $i \in IC_j$,
$e_{j,max}$:	$\max_{i \in ICj} \{e_{ji}\}$,
a_{ji}:	number of multiplicative terms x_j is involved in constraint $i \in IC_j$,
$a_{j,max}$:	$\max_{i \in ICj} \{a_{ji}\}$,
t_{ji}:	binary parameter indicating that x_j exists in a trigonometric expression in constraint $i \in IC_j$,
p_{ji}:	binary parameter indicating that x_j exists in an even power or abs expression in constraint $i \in IC_j$.

3 Solving COP Applications with *IP*

3.1 Selected Applications

The following applications have been selected to test the performance of the proposed IP.

1. Planar Truss Design [HWY03]

Consider the planar truss with parallel chords shown in Fig. 4 under the action of a uniformly distributed factored load of $p = 25$ kN/m, including the dead weight of approximately 1 kN/m. The truss is constructed from bars of square hollowed cross-section made of steel 37. For chord members, limiting tensile stresses are 190 MPa, for other truss members 165 MPa.

The members are divided into four groups according to the indices shown in Fig. 4. The objective of this problem is to minimize the volume of the

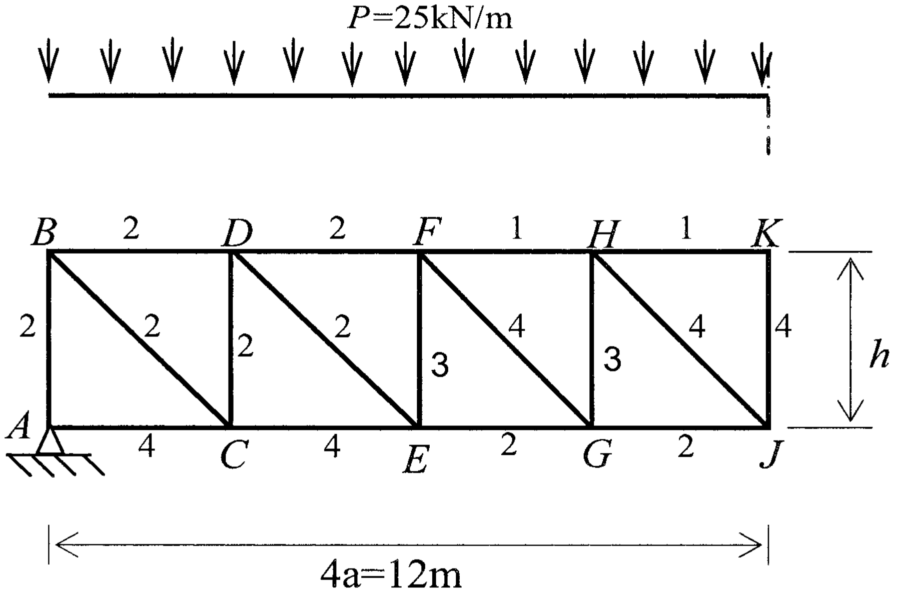


Fig. 4. Optimal design of a planar truss with parallel chords

truss, subject to stress and deflection constraints. Substituting material property parameters and the maximum allowable deflection that is 3.77 cm, the optimization model can be simplified. However, the original model is a discrete constrained optimization problem [HWY03], which is converted into a continuous optimization problem described below.

Maximize $f = -(600 * A_1 + 2910.4 * A_2 + 750 * A_3 + 1747.9 * A_4)$ (cm^3), subject to:
$A_1 \geq 30.0$ cm^2,
$A_2 \geq 24.0$ cm^2,
$A_3 \geq 14.4$ cm^2,
$A_4 \geq 11.2$ cm^2, and
$313920/A_1 + 497245/A_2 + 22500/A_3 + 67326/A_4 \leq 25200$ (kN-cm).

The first four inequalities are the stress constraints, while the last one is the deflection constraint.

The search space is: $A_1 = [30, 1000]$, $A_2 = [24, 1000]$, $A_3 = [14.4, 1000]$, $A_4 = [11.2, 1000]$.

Here A_i is the area in cm$_2$ with indices $i = 1, 2, 3, 4$. The objective function is a simple linear function, but the deflection constraint turns the feasible domain into a non-convex one.

Hsu et. al [HWY03] report an optimal design point for the original discrete model as $A = (55, 37.5, 15, 15)$, and the minimum volume of the truss for this solution is 179,608.5. However, for a continuous model, we find a minimum volume of the truss as 176,645.373 using the *IP* and other solvers used in the comparison.

2. Pressure Vessel Design [HWY03]

Figure 5 shows a cylindrical pressure vessel capped at both ends by hemispherical heads. This compressed air tank has a working pressure of 3,000 psi and a minimum volume of 750 feet^3. The design variables are the thickness of the shell and head, and the inner radius and length of the cylindrical section. These variables are denoted by x_1, x_2, x_3 and x_4, respectively. The objective function to be minimized is the total cost, including the material and forming costs expressed in the first two terms, and the welding cost in the last two terms. The first constraint restricts the minimum shell thickness and the second one, the spherical head thickness. The 3rd and 4th constraints represent minimum volume required and the maximum shell length of the tank, respectively. However, the last constraint is redundant due to the given search domain. The original model for this application is again a discrete constrained optimization problem. The continuous model is provided below.

Maximize $f = -(0.6224x_1x_3x_4 + 1.7781x_1x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3)$
subject to:
$-x_1 + 0.0193x_3 \leq 0,$
$-x_2 + 0.00954x_3 \leq 0,$
$(-\pi x_3^2x_4 - 4\pi x_3^3/3)/1296000 + 1 \leq 0,$ and
$x_4 - 240 \leq 0.$

The search space is: $x_1 = [1.125, 2]$, $x_2 = [0.625, 2]$, $x_3 = [40, 60]$, $x_4 = [40, 120]$. Hsu et. al [HWY03] list the reported optimal costs obtained by different formulations as given in Table 1. The least cost reported by *IIR* for designing a pressure vessel subjected to the given constraints is 7198.006. Other solvers report the same objective function value.

Table 1. List of the reported optimal costs obtained by different formulations

Problem Formulation	Reported optimal solution	Reference
Continuous	-7198.01	[HWY03]
Discrete	-7442.02	[HWY03]
Discrete	-8129.14	[San88]
Mixed discrete	-7198.04	[KK94]

3. Simplified Alkylation Process [BLW80, Pri]

This model describes a simplified alkylation process. The nonlinearities are bounded in a narrow range and introduce no additional computational burden.

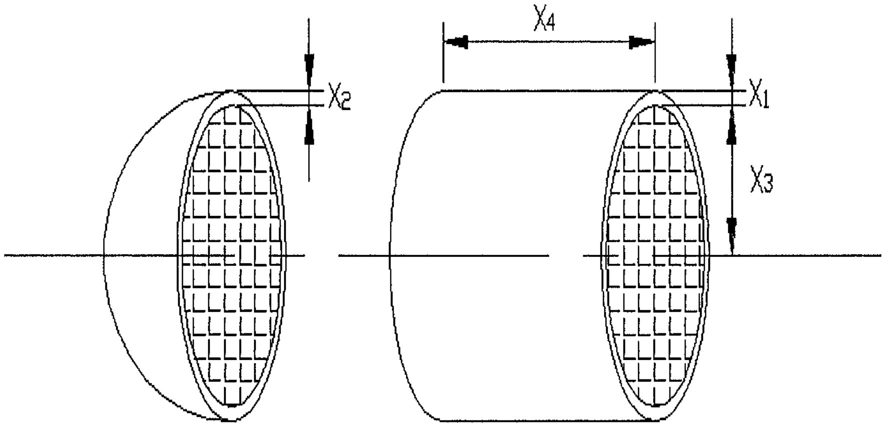


Fig. 5. Pressure vessel design

The design variables for the simplified alkylation process are olefins feed, isobutane recycle, acid feed, alkylate yield, isobutane makeup, acid strength, octane number, iC4 olefin ratio, acid dilution factor, F4 performance number, alkyate error, octane error, acid strength error, and F4 performance number error. The objective function maximizes profit per day. The constraints represent alkylate volumetric shrinkage equation, acid material balance, isobutane component balance, alkylate definition, octane number definition, acid dilution factor definition, and F4 performance number definition. The model is provided below.

Maximize $f = 5.04x_3x_4 + 0.35x_{13} + 3.36x_{14} - 6.3x_1x_2$
subject to:
$x_1 - 0.819672131147541 01x_3 - 0.81967213114754101x_{14} = 0,$
$-3x_2 + x_8x_{12} = -1.33,$
$22.2x_8 + x_7x_{11} = 35.82$ (acid material balance),
$-0.325x_5 - 0.01098x_6 + 0.00038x_6^2 + x_2x_{10} = 0.57425,$
$0.98x_4 - x_5(x_4 + 0.01x_1x_7) = 0,$
$x_1x_9 - x_3(0.13167x_6 - 0.0067x_6^2 + 1.12) = 0,$
$10x_{13} + x_{14} - x_3x_6 = 0$ (isobutane component balance).

The search space is: $x_1 = [1,5]$, $x_2 = [0.9, 0.95]$, $x_3 = [0,2]$, $x_4 = [0,1.2]$, $x_5 = [0.85, 0.93]$, $x_6 = [3,12]$, $x_7 = [1.2,4]$, $x_8 = [1.45,1.62]$, $x_9 = [0.99, 1.0\dot{1}]$, $x_{10} = [0.99, 1.0\dot{1}]$, $x_{11} = [0.9, 1.112]$, $x_{12} = [0.99, 1.0\dot{1}]$, $x_{13} = [0, 1.6]$, $x_{14} = [0, 2]$. The optimal solution for alkylation process is 1.765.

4. Stratified Sample Design [Pri]

The problem is to find a sampling plan that minimizes the related cost and yields variances of the population limited by an upper bound.

Maximize $f = -(x_1 + x_2 + x_3 + x_4)$
subject to
$0.16/x_1 + 0.36/x_2 + 0.64/x_3 + 0.64/x_4 - 0.010085 \leq 0,$
$4/x_1 + 2.25/x_2 + 1/x_3 + 0.25/x_1 - 0.0401 \leq 0.$

The search space is $x_1 = [100, 400000]$, $x_2 = [100, 300000]$, $x_3 = [100, 200000]$, and $x_4 = [100, 100000]$. The optimum value for this problem is -725.479.

5. Robot [Pri, BFG87]

This model is designed for the analytical trajectory optimization of a robot with seven degrees of freedom.

$$\text{Maximize } f = -((x_1 - x_8)^2 + (x_2 - x_9)^2 + (x_3 - x_{10})^2 + (x_4 - x_{11})^2 + (x_5 - x_{12})^2 + (x_6 - x_{13})^2 + (x_7 - x_{14})^2)$$

subject to
$\cos x_1 + \cos x_2 + \cos x_3 + \cos x_4 + \cos x_5 + \cos x_6 + 0.5 * \cos x_7 = 4,$
$\sin x_1 + \sin x_2 + \sin x_3 + \sin x_4 + \sin x_5 + \sin x_6 + 0.5 * \sin x_7 = 4.$

The search space is $x_i = [-10, 10]$, $i = 1, 2, 3, 4, \ldots, 14$. The optimal solution for this problem is 0.0.

4 Numerical Results

The numerical results are provided in Table 2. We compare *IP* results with five different solvers that are linked to the commercial software GAMS (`www.gams.com`) and FSQP [ZT96, LZT97], the code was provided by AEM [Aem]. The solvers used in this comparison are BARON [Sah03], Conopt [Dru96], LGO [Pin97], Minos [MS87], and Snopt [GMS97].

For each application we report the absolute deviation from the global optimum obtained at the end of the run and the CPU time necessary for each run in new Standard Time Units (STU, [SNS+02] 10^5 times as defined in [TZ89]). All runs were executed on a PC with 256 MB RAM, 2.4 GHz P4 Intel CPU, on Windows platform. The *IP* code was developed with Visual C++ 6.0 interfaced with the PROFIL interval arithmetic library [Knu94] and FSQP. One new STU is equivalent to 229.819 seconds on our machine. GAMS solvers were run until each solver terminates on its own without restricting the CPU time used or the number of iterations made. FSQP was run with a maximum number of iterations allowed, that is 100 in this case. However, in these applications FSQP never reached this iteration limit. *IP* was run until no improvement in the *CLB* was obtained as compared with the previous stage of the search tree. However, if a feasible solution has not been found yet, the stopping criterion became the least feasibility degree of uncertainty, INF_Y.

Table 2. Comparison of results obtained. The problem names (global optimum values) are Pressure Vessel (-7198.006), Planar Truss (-176645.373), Alkyl (1.765), and Sample (-725.479), respectively. For the IIR method the number of stages, FSQP calls, and the average number of variables in parallel (maximal/minimal), and the number of function calls were (2, 18, 3.04/4.2, 402), (2, 27, 3/4.2, 257), (2, 631, 4.04/5.3, 8798), and (6, 917, 3.56/4.3, 12000), respectively. The summary of the average results for the first 4 problems is: for the number of stages is 3.000, the number of FSQP calls is 398.250, and the average number of function calls is 5364.25. For the 5., robot optimization the global optimum was zero, and the efficiency measures (1, 1, 2.25/3.2, 62). The final summary provides the following average figures: the number of stages is 2.600, the number of FSQP calls is 318.800, and the average number of function calls is 4303.800

Probl.	Dim.	Performance	IIR	FSQP	Baron	Conopt	LGO	Minos	Snopt
		Deviation	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	4	CPU(STU)	0.000	0.000	0.001	0.000	0.000	0.000	0.000
		Deviation	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	4	CPU(STU)	0.001	0.000	0.001	0.000	0.000	0.000	0.000
		Deviation	0.000	0.000	0.000	0.000	0.000	1.765	0.000
3	14	CPU(STU)	0.263	0.000	0.292	0.000	0.003	0.000	0.000
		Deviation	1.200	26706.5	1.158	1.201	∞	1.168	0.000
4	4	CPU(STU)	0.056	0.000	0.000	0.000	0.002	0.000	0.000
Summary		Avg. deviation	0.300	6676.625	0.290	0.300	0.000	0.733	0.000
		Std. dev. for it	0.600	13353.25	0.579	0.601	0.000	0.881	0.000
		Avg. CPU time	0.080	0.000	0.073	0.000	0.001	0.000	0.000
		# best solutions	3	3	3	3	3	2	4
		# unsolved probl.	0	0	0	0	1	0	0
		Deviation	0.000	0.000	NA	27.1	5.463	343.022	13.391
5	14	CPU(STU)	0.000	0.000		0.000	0.046	0.001	0.000
Final sum.		Avg. deviation	0.240	5341.300	0.290	5.659	1.366	69.191	2.678
		Std. dev. for it	0.537	11943.51	0.579	11.994	2.732	153.078	5.989
		Avg. CPU time	0.064	0.000	0.073	0.000	0.010	0.000	0.000
		# best solutions	4	4	3	3	3	2	4
		# unsolved probl.	0	0	1	0	1	0	0

In Table 2, we report additional information for IP. For each application, we report the number of tree stages where IP stops, the number of times FSQP is invoked, the average number of variables partitioned in parallel for a parent box (the maximum and minimum numbers are also indicated in parenthesis), and the number of function calls invoked outside FSQP. We provide two summaries of results obtained excluding and including the robot application. The reason for it is that BARON is not enabled to solve trigonometric models.

When we analyze the results, we observe that Snopt identifies the optimum solution for four of the applications excluding the robot problem. However, its performance is inferior for the robot problem as compared to IP and FSQP.

For the robot application, FSQP identifies the global optimum solution in the initial box itself (stage zero in IP). That is why IP stops at the first stage. The performance of the local optimizers, Minos and Conopt, is significantly inferior in this problem. In the pressure vessel and planar truss applications, all GAMS solvers, IP and FSQP identify the global optimum with short CPU times (IP and BARON take longer CPU time). For the Alkyl problem, Minos is stuck at a local stationary point while BARON and IP take longer CPU times. In the Sample application, LGO does not converge and FSQP ends up with a very inferior solution. On the other hand, IP runs for 6 tree stages and results in an absolute deviation that is comparable with those of BARON, Conopt and Minos.

When the final summary of the results is analyzed, we observe that IP's performance is as good as BARON's (which is a complete and reliable solver) in identifying the global optimum and regarding CPU time. The use of FSQP in IP (rather than the Generalized Reduced Gradient local search procedure available in BARON) becomes an advantage for IP in the solution of the robot problem. Furthermore, IP does not have any restrictions in dealing with trigonometric functions. The impact of interval partitioning on performance is particularly observed in the Sample application where FSQP fails to converge. For these applications, the number of tree stages that IP has to run for is quite small (two) except for the Sample problem. The average number of variables partitioned in parallel in IP varies between 2 and 4. The dynamic parallelism imposed by the weighting method seems to be effective as it is observed that different scales of parallelism are adopted for different applications.

5 Conclusion

A new interval partitioning approach (IP) is proposed for solving constrained optimization applications. This approach has two supportive features: a flexible tree search management strategy and a new variable selection rule for partitioning parent boxes. Furthermore, the proposed IP method is interfaced with the local search FSQP that is invoked when no improvement is detected regarding the area of disposed boxes. FSQP is capable of identifying feasible stationary points quickly within the restricted areas of boxes.

The tree management strategy proposed here can also be used in non-interval partitioning algorithms such as BARON and LGO. It is effective in the sense that it allows going deeper into selected promising parent boxes while providing a larger perspective on how promising a parent box is by comparing it to all other boxes available in the box list of the current stage. The proposed variable selection rule is able to make an inference related to the pair of variables that have most impact on the uncertainty of a box's potential to contain feasible and optimal solutions. By partitioning the selected maximum impact variables these uncertainties are reduced in the immediate sibling boxes after the parent is partitioned. The latter results in earlier disposal of boxes due to their sub-optimality and infeasibility.

This whole framework enhances the convergence speed of the interval partitioning algorithm in the solution of COP problems. In the numerical results it is demonstrated that the proposed *IP* can compete with available commercial solvers on several COP applications. The methodology developed here is generic and can be applied to other areas of global optimization such as the continuous Constraint Satisfaction Problem (CSP) and box-constrained global optimization.

Acknowledgement. The present work has been partially supported by the grants OTKA T 048377, and T 046822. We wish to thank Professor Andre Tits (Electrical Engineering and the Institute for Systems Research, University of Maryland, USA) for providing the source code of *CFSQP*.

References

[Aem] www.aemdesign.com/FSQPmanyobj.htm

[AH83] Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Academic Press Inc. New York, USA (1983)

[AMF95] Androulakis, I.P., Maranas, C.D., Floudas, C.A.: AlphaBB: a global optimization method for general constrained nonconvex problems. Journal of Global Optimization, **7**, 337-363 (1995)

[AS00] Al-Khayyal, F.A., Sherali, H. D.: On finitely terminating branch-and-bound algorithms for some global optimization problems. SIAM Journal on Optimization, **10**, 1049–1057 (2000)

[BEG94] Ben-Tal, A., Eiger, G., Gershovitz, V.: Global optimization by reducing the duality gap. Mathematical Programming, **63**, 193–212 (1994)

[BFG87] Benhabib, B., Fenton, R.G., Goldberg, A.A.: Analytical trajectory optimization of seven degrees of freedom redundant robot. Transactions of the Canadian Society for Mechanical Engineering, **11**, 197–200 (1987)

[BHR92] Burkard, R.E., Hamacher, H., Rote, G.: Sandwich approximation of univariate convex functions with an application to separable convex programming. Naval Research Logistics, **38**, 911–924 (1992)

[BLW80] Berna, T., Locke, M., Westerberg, A.W.: A new approach to optimization of chemical processes. American Institute of Chemical Engineers Journal, **26**, 37–43 (1980)

[BMV94] Benhamou, F., McAllester, D., Van Hentenryck, P.: CLP(Intervals) revisited. Proc. of ILPS'94, International Logic Programming Symposium, 124–138 (1994)

[CGC00] Casado, L.G., García, I., Csendes, T.: A new multisection technique in interval methods for global optimization. Computing, **65**, 263–269 (2000)

[CGT94] Conn, A.R., Gould, N., Toint, Ph.L.: A note on exploiting structure when using slack variables. Mathematical Programming, **67**, 89–99 (1994)

[CR97] Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. on Numerical Analysis, **34**, 922–938 (1997)

[DNS97] Dallwig, S., Neumaier, A., Schichl, H.: GLOPT - a program for constrained global optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds) Developments in Global Optimization. Kluwer, Dordrecht, 19–36 (1997)

[Dru96] Drud, A.S.: CONOPT: A System for Large Scale Nonlinear Optimization. Reference Manual for CONOPT Subroutine Library, ARKI Consulting and Development A/S, Bagsvaerd, Denmark (1996)

[FGW02] Forsgren, A., Gill, P.E., Wright, M.H.: Interior methods for nonlinear optimization. SIAM Review., **44**, 525–597 (2002)

[FV93] Floudas, C.A., Visweswaran, V.: Primal-relaxed dual global optimization approach. J. Opt. Th. Appl., **78**, 187–225 (1993)

[GMS97] Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP Algorithm for Large-scale Constrained Optimization. Numerical Analysis Report 97-2, Department of Mathematics, University of California, San Diego, La Jolla, CA (1997)

[Han92] Hansen, E.R.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (1992)

[HJL92] Hansen, P., Jaumard, B., Lu, S.: Global optimization of univariate Lipschitz functions: I. Survey and properties. Mathematical Programming, **55**, 251–272 (1992)

[HS80] Hansen, E., Sengupta, S.: Global constrained optimization using interval analysis. In: Nickel, K.L. (ed) Interval Mathematics. Academic Press, New York (1980)

[HTD92] Horst, R., Thoai, N.V., De Vries, J.: A new simplicial cover technique in constrained global optimization. J. Global Opt., **2**, 1–19 (1992)

[HW93] Hansen, E., Walster, G.W.: Bounds for Lagrange multipliers and optimal points. Comput. Math. Appl., **25**, 59 (1993)

[HWY03] Hsu, Y-L., Wang, S-G., Yu, C-C.: A sequential approximation method using neural networks for engineering design optimization problems. Engineering Optimization, **35**, 489–511 (2003)

[Kea91] Kearfott, R.B.: Decompostion of arithmetic expressions to improve the behaviour of interval iteration for nonlinear systems. Computing, **47**, 169–191 (1991)

[Kea94] Kearfott, R.B.: On Verifying Feasibility in Equality Constrained Optimization Problems. preprint (1994)

[Kea96a] Kearfott, R.B.: A review of techniques in the verified solution of constrained global optimization problems. In: Kearfott, R.B., Kreinovich, V. (eds) Applications of Interval Computations. Kluwer, Dordrecht, Netherlands, pp. 23–60 (1996)

[Kea96b] Kearfott, R.B.: Test results for an interval branch and bound algorithm for equality-constrained optimization. In: Floudas, C., Pardalos, P. (eds) State of the Art in Global Optimization: Computational Methods and Applications. Kluwer, Dordrecht, Netherlands, pp. 181–200 (1996)

[Kea96c] Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)

[Kea03] Kearfott, R.B.: An Overview of the GlobSol Package for Verified Global Optimization. talk given for the Department of Computing and Software, McMaster University, Ontario, Canada (2003)

[Kea04] Kearfott, R.B.: Empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. Optimization Methods and Software, **accepted** (2004)

[Kea05] Kearfott, R.B.: Improved and simplified validation of feasible points: inequality and equality constrained problems. Mathematical Programming, **submitted** (2005)

[KK94] Kannan, B.K., Kramer, S.N.: An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Journal of Mechanical Design, **116**, 405–411 (1994)

[Knu94] Knuppel, O.: PROFIL/BIAS – a fast interval library. Computing, **53**, 277–287 (1994)

[Kor85] Korf, R.E.: Depth-first iterative deepening: An optimal admissible tree search. Artificial Intelligence, **27**, 97–109 (1985)

[LZT97] Lawrence, C.T., Zhou, J.L., Tits, A.L.: User's Guide for CFSQP Version 2.5: A Code for Solving (Large Scale) Constrained Nonlinear (minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Institute for Systems Research, University of Maryland, College Park, MD (1997)

[Mar03] Markót, M.C.: Reliable Global Optimization Methods for Constrained Problems and Their Application for Solving Circle Packing Problems. PhD dissertation, University of Szeged, Hungary (2003)

[MFCC05] Markót, M.C., Fernandez, J., Casado, L.G., Csendes, T.: New interval methods for constrained global optimization. Mathematical Programming, **accepted** (2005)

[MNW$^+$01] Morales, J.L., Nocedal, J., Waltz, R., Liu, G., Goux, J.P.: Assessing the Potential of Interior Methods for Nonlinear Optimization. Optimization Technology Center, Northwestern University, USA (2001)

[Moo66] Moore, R.E.: Interval Anlaysis. Prentice-Hall, Englewood Cliffs, New Jersey (1966)

[MS87] Murtagh, B.A., Saunders, M.A.: MINOS 5.0 User's Guide. Report SOL 83-20, Department of Operations Research, Stanford University, USA (1987)

[Pin97] Pintér, J.D.: LGO – a program system for continuous and Lipschitz global optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds) Developments in Global Optimization. Kluwer Academic Publishers, Boston/Dordrecht/London, pp. 183–197 (1997)

[PR02] Pardalos, P.M., Romeijn, H.E.: Handbook of Global Optimization Volume 2. Nonconvex Optimization and Its Applications. Springer, Boston/Dordrecht/London (2002)

[Pri] PrincetonLib.: Princeton Library of Nonlinear Programming Models. www.gamsworld.org/performance/princetonlib/princetonlib.htm

[RC95] Ratz, D., Csendes, T.: On the selection of subdivision directions in interval branch-and-bound methods for global optimization. J. Global Optimization, **7**, 183–207 (1995)

[Rob73] Robinson, S.M.: Computable error bounds for nonlinear programming. Mathematical Programming, **5**, 235–242 (1973)

[RR88] Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. Ellis Horwood, Chichester (1988)

[RS95] Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers and Chemical Engineering, **19**, 551–566 (1995)

[RS96] Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. Journal of Global Optimization, **8**, 107–139 (1996)

[Sah96] Sahinidis, N.V.: BARON: A general purpose global optimization software package. Journal of Global Optimization, **8**, 201–205 (1996)

[Sah03] Sahinidis, N.V.: Global optimization and constraint satisfaction: the branch-and-reduce approach. In: Bliek, C., Jermann, C., Neumaier, A. (eds) COCOS 2002. LNCS, **2861**, 1–16 (2003)

[San88] Sandgren, E.: Nonlinear integer and discrete programming in mechanical design. Proceeding of the ASME Design Technology Conference, Kissimmee, FL, 95–105 (1988)

[SNS$^+$02] Scherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.-H., Nguyen, T.-V.: Benchmarking global optimization and constraint satisfaction codes. Global Optimization and Constraint Satisfaction: First International Workshop on Global Constraint Optimization and Constraint Satisfaction, COCOS 2002, Valbonne-Sophia Antipolis, France (2002)

[SP99] Smith, E.M.B., Pantelides, C.C.: A symbolic reformulation/spatial branch and bound algorithm for the global optimization of nonconvex MINLP's. Computers and Chemical Eng., **23**, 457–478 (1999)

[TS02] Tawarmalani, M., Sahinidis, N.V.: Convex extensions and envelopes of lower semi-continuous functions. Mathematical Programming, **93**, 247–263 (2002)

[TS04] Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming, **99**, 563–591 (2004)

[TTT85] Tuy, H., Thieu, T.V., Thai., N.Q.: A conical algorithm for globally minimizing a concave function over a closed convex set. Mathematics of Operations Research, **10**, 498 (1985)

[TZ89] Törn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, **350**, Springer, Berlin (1989)

[Wol94] Wolfe, M.A.: An interval algorithm for constrained global optimization. J. Comput. Appl. Math., **50**, 605–612 (1994)

[ZB03] Zilinskas, J., Bogle, I.D.L.: Evaluation ranges of functions using balanced random interval arithmetic. Informatica, **14**, 403–416 (2003)

[ZT96] Zhou, J.L., Tits, A.L.: An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. SIAM J. on Optimization, **6**, 461–487 (1996)

A Survey of Methods for the Estimation Ranges of Functions Using Interval Arithmetic

Julius Žilinskas[1] and Ian David Lockhart Bogle[2]

[1] Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania julius.zilinskas@mii.lt

[2] Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK d.bogle@ucl.ac.uk

1 Introduction

Interval arithmetic is a valuable tool in numerical analysis and modeling. Interval arithmetic operates with intervals defined by two real numbers and produces intervals containing all possible results of corresponding real operations with real numbers from each interval. An interval function can be constructed replacing the usual arithmetic operations by interval arithmetic operations in the algorithm calculating values of functions. An interval value of a function can be evaluated using the interval function with interval arguments and determines the lower and upper bounds for the function in the region defined by the vector of interval arguments.

A disadvantage of interval arithmetic is the dependency problem which causes widening of computed intervals and overestimation, i.e. the interval value of the function is much wider than the range of real function values. In this chapter the ways to estimate ranges of functions using standard interval arithmetic and special inner interval arithmetic are reviewed.

2 Interval Arithmetic

Interval arithmetic is proposed in [Moo66]. Interval arithmetic operates with real intervals $\underline{\overline{x}} = [\underline{x}, \overline{x}] = \{x \in \Re | \underline{x} \leq x \leq \overline{x}\}$, defined by two real numbers $\underline{x} \in \Re$ and $\overline{x} \in \Re$, $\underline{x} \leq \overline{x}$. In this chapter interval arithmetic proposed in [Moo66] is called standard interval arithmetic to distinguish from other interval arithmetic reviewed. For any real arithmetic operation $x \circ y$ the corresponding standard interval arithmetic operation $\underline{\overline{x}} \circ \underline{\overline{y}}$ is defined as an operation whose result is an interval containing every possible number produced by the real operation with the real numbers from each interval. The standard interval arithmetic operations are defined as:

$$\begin{aligned}
\underline{\overline{x}} + \underline{\overline{y}} &= \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right], \\
\underline{\overline{x}} - \underline{\overline{y}} &= \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right], \\
\underline{\overline{x}} \times \underline{\overline{y}} &= \begin{cases}
\left[\underline{x}\underline{y}, \overline{x}\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} > 0, \\
\left[\overline{x}\underline{y}, \overline{x}\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} \ni 0, \\
\left[\overline{x}\underline{y}, \underline{x}\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} < 0, \\
\left[\underline{x}\overline{y}, \overline{x}\overline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} > 0, \\
\left[\min(\underline{x}\overline{y}, \overline{x}\underline{y}), \max(\underline{x}\underline{y}, \overline{x}\overline{y})\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} \ni 0, \\
\left[\overline{x}\underline{y}, \underline{x}\underline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} < 0, \\
\left[\underline{x}\overline{y}, \overline{x}\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} > 0, \\
\left[\underline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} \ni 0, \\
\left[\overline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} < 0,
\end{cases} \\
\underline{\overline{x}} / \underline{\overline{y}} &= \begin{cases}
\left[\underline{x}/\overline{y}, \overline{x}/\underline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} > 0, \\
\left[\overline{x}/\overline{y}, \underline{x}/\underline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} < 0, \\
\left[\underline{x}/\underline{y}, \overline{x}/\underline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} > 0, \\
\left[\overline{x}/\overline{y}, \underline{x}/\overline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} < 0, \\
\left[\underline{x}/\underline{y}, \overline{x}/\overline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} > 0, \\
\left[\overline{x}/\underline{y}, \underline{x}/\overline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} < 0.
\end{cases}
\end{aligned}$$

Computers with limited precision can not represent all real numbers exactly. To guarantee the interval enclosure outward rounding should be used: during interval computation the number representing the lower end of the resulting interval ($\underline{x}$) should be rounded toward $-\infty$ and the number representing the upper end of the resulting interval ($\overline{x}$) should be rounded toward $+\infty$. Standard interval arithmetic with outward rounding may be used to find guaranteed interval enclosure of the result of real operations, while real arithmetic may produce incorrect results because of rounding errors in real computations. Although computers with limited precision can not represent all real numbers exactly, they can represent guaranteed enclosures using intervals, for example $\underline{\overline{\pi}} = [3.1415, 3.1416]$.

Because of the necessary outward rounding in interval arithmetic, the implementation of interval arithmetic is not straightforward. Publicly available C and C++ packages for interval arithmetic are investigated and compared experimentally in [Žil05]. The speed of interval arithmetic operations and width of resulting intervals have been investigated with experimental comparisons. Experiments have shown that commercial SUN Forte C++ interval library [Sun01] is the fastest and most accurate from the investigated implementations of interval arithmetic. However when the commercial library is not available or a different platform is used, the freely available C++ interval library filib++ [LTG$^+$01] is preferable. Moreover it is editable contrary to the integrated interval arithmetic library for the SUN Forte Compiler, and therefore it is the most preferable for implementation of non standard interval arithmetic reviewed in the following sections.

An interval function can be constructed replacing the usual arithmetic operations by the interval arithmetic operations in the algorithm for calculation values of the function. An interval value of the function can be evaluated using the interval function with interval arguments. The resulting interval always encloses the range of real function values in the hyper-rectangular region defined by the vector of interval arguments:

$$\{f(X)\,|X \in \overline{\underline{X}}, \underline{X} \in \Re^n, \overline{X} \in \Re^n\} \subseteq \overline{\underline{f}}(\overline{\underline{X}}),$$

where $f : \Re^n \to \Re$, $\overline{\underline{f}} : [\Re, \Re]^n \to [\Re, \Re]$. Because of this property the interval value of the function can be used as the lower and upper bounds for the function in the region.

The bounds may be used in global optimization to detect the sub-regions of the feasible region which cannot contain a global minimizer. Such sub-regions may be discarded from further search. The first version of interval global optimization algorithm was oriented to minimization of a rational function by bisection of sub-domains [Ske74]. Interval methods for global optimization were further developed in [Moo77, Han78a, Han78b], where the interval Newton method and the test of strict monotonicity were introduced. A thorough description including theoretical as well as practical aspects can be found in [HW03] where the very efficient interval global optimization method involving monotonicity and non-convexity tests and the special interval Newton method is described. The method assumes that the objective function is twice continuously differentiable. The mathematical expressions of the functions should be available. If the monotonicity and non-convexity tests and interval Newton method are not used the method can minimize even noncontinuous functions, but then it is not so efficient.

A branch and bound technique is usually used to construct interval global optimization algorithms. An iteration of a classical branch and bound algorithm processes a node in the search tree representing an yet unexplored subspace of the solution space. Iterations have three main components: selection of the node to process, bound calculation, and branching. The rules of selection, branching and bounding differ from algorithm to algorithm. All interval global optimization branch and bound algorithms use the hyper-rectangular partitions and branching is usually performed bisecting the hyper-rectangle into two. The bounding rule describes how the bounds for a minimum are found. In interval global optimization algorithms bounds are estimated using interval arithmetic.

A disadvantage of standard interval arithmetic is the dependency problem [HW03]: when a given variable occurs more than once in interval computation, it is treated as a different variable in each occurrence. This causes widening of computed intervals and overestimation of the range of function values.

For example, for real arithmetic, definitions

$$x^2 + 2xy = (x + 2y)x = (x + y)^2 - y^2$$

are equal. However, if $\overline{\underline{x}} = [-1, 1]$ and $\overline{\underline{y}} = [-1, 1]$, intervals

$$\underline{\overline{x}}^2 + 2\underline{\overline{x}}\,\underline{\overline{y}} = [-2,3],\ (\underline{\overline{x}} + 2\underline{\overline{y}})\underline{\overline{x}} = [-3,3],\ (\underline{\overline{x}} + \underline{\overline{y}})^2 - \underline{\overline{y}}^2 = [-1,4]$$

overestimate the range of real function values $[-1,3]$ by 25–50%.

The tightness of bounds is a very important factor for efficiency of branch and bound based global optimization algorithms. An experimental model of interval arithmetic with controllable tightness of bounds to investigate the impact of bounds tightening in interval global optimization is proposed in [ŽŽ06]. The experimental results on efficiency of tightening bounds are presented for several test and practical problems. Experiments have shown that the relative tightness of bounds strongly influences efficiency of global optimization algorithms based on branch and bound approach combined with interval arithmetic.

Standard interval arithmetic provides guaranteed bounds but sometimes they are too pessimistic. Standard interval arithmetic is used in global optimization providing guaranteed solutions, but there are problems for which the time of optimization is too long. We hope that it is possible to construct global optimization algorithms using estimates of ranges of functions instead of interval bounds. In this chapter the methods to estimate ranges of functions using interval arithmetic and special inner interval arithmetic are reviewed.

3 Random Interval Arithmetic

Random interval arithmetic (RIA) proposed in [AL01] is obtained by choosing the standard or inner interval operations randomly with the same probability at each step of the computation. The inner interval arithmetic operations are defined as:

$$\underline{\overline{x}} +_{in} \underline{\overline{y}} = \left[\underline{x} + \overline{y} \vee \overline{x} + \underline{y}\right],$$

$$\underline{\overline{x}} -_{in} \underline{\overline{y}} = \left[\underline{x} - \underline{y} \vee \overline{x} - \overline{y}\right],$$

$$\underline{\overline{x}} \times_{in} \underline{\overline{y}} = \begin{cases} \left[\underline{x}\overline{y} \vee \overline{x}\underline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} > 0, \\ \left[\underline{x}\underline{y}, \underline{x}\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} \ni 0, \\ \left[\overline{x}\overline{y} \vee \underline{x}\underline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} < 0, \\ \left[\underline{x}\underline{y}, \overline{x}\underline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} > 0, \\ \left[\max(\underline{x}\overline{y}, \overline{x}\underline{y}), \min(\underline{x}\underline{y}, \overline{x}\overline{y})\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} \ni 0, \\ [\overline{x}\overline{y}, \underline{x}\overline{y}], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} < 0, \\ \left[\underline{x}\underline{y} \vee \overline{x}\overline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} > 0, \\ \left[\overline{x}\overline{y}, \overline{x}\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} \ni 0, \\ \left[\overline{x}\underline{y} \vee \underline{x}\overline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} < 0, \end{cases}$$

$$\underline{\overline{x}}/_{in}\underline{\overline{y}} = \begin{cases} \left[\underline{x}/\underline{y} \vee \overline{x}/\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} > 0, \\ \left[\overline{x}/\underline{y} \vee \underline{x}/\overline{y}\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} < 0, \\ [\underline{x}/\overline{y}, \overline{x}/\overline{y}], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} > 0, \\ \left[\overline{x}/\underline{y}, \underline{x}/\underline{y}\right], & \underline{\overline{x}} \ni 0, \underline{\overline{y}} < 0, \\ \left[\underline{x}/\overline{y} \vee \overline{x}/\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} > 0, \\ \left[\overline{x}/\overline{y} \vee \underline{x}/\underline{y}\right], & \underline{\overline{x}} < 0, \underline{\overline{y}} < 0, \end{cases}$$

where $[a \vee b] = [min(a, b), max(a, b)]$.

Reference [AL01] suggests estimating the range of function values using a number of sample intervals evaluated using RIA. The estimation is based on the assumptions that the distribution of the centres of the evaluated intervals is normal with a very small relative standard deviation and the distribution of the radii is normal but taking only positive values. The mean value of the centres $\mu_{centres}$, the mean value of the radii μ_{radii} and the standard deviations of the radii σ_{radii} of the intervals computed using RIA are used to evaluate an approximate range of the function

$$[\mu_{centres} \pm (\mu_{radii} + \alpha\sigma_{radii})] , \qquad (1)$$

where α is between 1 and 3 depending on the number of samples and the desired probability that the exact range is included in the estimated range. It is suggested in [AL01] that a compromise between efficiency and robustness can be obtained using $\alpha = 1.5$ and 30 samples. Experimental results presented in [AL01] for some functions over small intervals show that RIA provides tight estimates of the ranges of the considered function values with probability close to 1. However, in their experiments, the intervals of variables of the function considered were small. In the case of large intervals of variables, and particularly for multivariable functions, the obtained estimates for a range of function values frequently do not fully enclose the range of function values.

4 Balanced Random Interval Arithmetic

For the application of RIA to global optimization and modeling it is important to extend these ideas to the case of functions defined over large multidimensional regions. Balanced random interval arithmetic (BRIA) proposed in [ŽB04] extending the ideas of [AL01], is obtained by choosing the standard and inner interval operations at each step of the computation randomly with predefined probabilities of the standard and inner operations. A number of sample intervals are evaluated using BRIA. It is assumed that the distribution of centres of the evaluated balanced random intervals is normal and that the distribution of radii is folded normal [JK95], also known as absolute normal, because the radii cannot be negative. The range of values of the function in the defined region is estimated using the mean values (μ) and the standard deviations (σ) of centres and radii of the evaluated balanced random intervals:

$$[\mu_{centres} \pm (3.0\sigma_{centres} + \mu_{radii} + 3.0\sigma_{radii})] . \qquad (2)$$

If the estimated range exceeds the standard interval function in the same region, the evaluated interval is intersected with the standard interval while not reducing the probability that the estimated range contains all the values of function in the region.

The result of BRIA is equal to the result of standard interval arithmetic when the predefined probability of standard interval operations is equal to 1 and the probability of inner interval operations is equal to 0. The result of BRIA is equal to the result of inner interval arithmetic when the predefined probability of standard interval operations is equal to 0 and the probability of inner interval operations is equal to 1. BRIA provides wider or narrower ranges depending on the predefined probabilities. The values used for the predefined probabilities depend on the balance required between tightness of resulting intervals and the probability that resulting intervals contain all the values of the function.

The sample mean value and the sample standard deviation of centres distributed by normal distribution may be estimated using simple formulas

$$\mu_{centres} = \frac{1}{N}\sum_{i}^{N} centre_i,$$

$$\sigma_{centres} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(centre_i - \mu_{centres})^2},$$

where $centre_i = (\underline{f_i} + \overline{f_i})/2$. The sample standard deviation of centres distributed by normal distribution can be estimated using the second moment:

$$\sigma_{centres} = \sqrt{M_{2centres} - \mu^2_{centres}},$$

$$M_{2centres} = \frac{1}{N}\sum_{i=1}^{N} centres_i^2.$$

Second and fourth moments are needed to estimate the sample mean value and the sample standard deviation of radii distributed by a folded normal distribution:

$$\mu_{radii} = \sqrt{\sqrt{\frac{3M^2_{2radii} - M_{4radii}}{2}}},$$

$$\sigma_{radii} = \sqrt{m_2 - \sqrt{\frac{3M^2_{2radii} - M_{4radii}}{2}}},$$

$$M_{2radii} = \frac{1}{N}\sum_{i=1}^{N} radii_i^2,$$

$$M_{4radii} = \frac{1}{N}\sum_{i=1}^{N} radii_i^4,$$

where $radii_i = (\overline{f_i} - \underline{f_i})/2$.

The Kolmogorov-Smirnov test [Dar57, PFTW92] can be used to test the acceptability of the sample to the hypothetical distribution. The Kolmogorov-Smirnov statistic is the maximum value of the absolute difference between two cumulative distribution functions. A routine for evaluation of the significance level to disprove the null hypothesis that the distributions are the same, is implemented in [PFTW92]. Cumulative distribution function of the hypothetical distribution should be provided. The cumulative distribution function of a normal distribution is known:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) dt.$$

The cumulative distribution function of the folded normal distribution is:

$$P_f(x) = P(x) - P(-x), x \geq 0.$$

These functions can be defined using the complementary error function implemented in [PFTW92]:

$$P(x) = 1 - \frac{1}{2}\text{erfc}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right),$$

$$P_f(x) = -\frac{1}{2}\text{erfc}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) + \frac{1}{2}\text{erfc}\left(\frac{-x-\mu}{\sigma\sqrt{2}}\right), x \geq 0.$$

For example the balanced random interval values of a multidimensional scaling stress function with data from soft drinks testing [Mat96] were computed for a randomly generated subregion. The histograms of centres and radii of a sample of 1000 interval function values computed using BRIA are presented in Fig. 1. Values of the predefined probability (0.5 and 0.6) of the standard interval arithmetic operations are shown. The centres and radii of interval function values computed using standard and inner interval arithmetics are shown as vertical lines and denoted by 'inner' and 'standard'. The mean values of the centres and radii move towards the centre and radius of the standard interval when the probability of standard interval operations is increasing. The sample means and the sample standard deviations of the centres and radii of the generated random intervals are presented above the figures. The graphs of normal distributions for centres and the folded normal distributions for radii with the evaluated sample means and standard deviations are also shown in Fig. 1. The histograms of centres and radii of the random intervals are similar to the theoretical distributions. The Kolmogorov-Smirnov test was used to test the acceptability of the hypothetical distributions. P-values of tests, denoted PKS, are presented below the figures. The Kolmogorov-Smirnov test supports our observation: at the standard significance level 0.05 the theoretical hypotheses should be accepted.

When the predefined probability of the standard interval operations is 0.5, the standard or inner interval operations are chosen with the same probability.

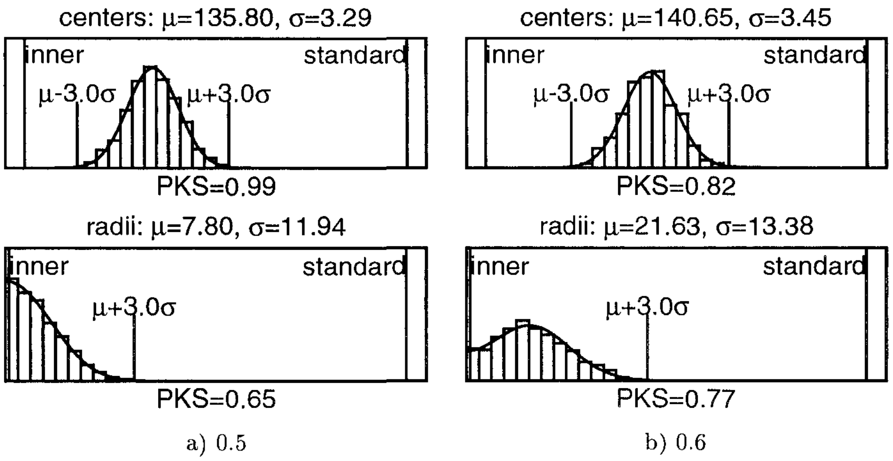


Fig. 1. The histograms of centres and radii of balanced random intervals with shown values of the predefined probability of the standard interval arithmetic operations for the multidimensional scaling function

Therefore, Fig. 1a may represent results of RIA as well. However, the standard deviation of centres is not small, therefore (2) is preferable to (1).

The disadvantage of BRIA (and RIA as well) is that computations with this arithmetic use more calculations than with standard interval arithmetic, because the number of random intervals that must be evaluated. When 30 samples are used, BRIA is more than 31 times more expensive than standard interval arithmetic. Another disadvantage is the assumption that distributions of centres and radii of the evaluated balanced random intervals are normal and folded normal respectively. The assumption is not true when computations involve a small number of arithmetic operations, what was shown in [ŽB03] for simple mathematical global optimization test functions.

5 Underestimating Interval Arithmetic

Kaucher arithmetic [Kau77, KNZ96] defining underestimates is useful to estimate how much standard interval bounds overestimate the range of values of function. A regularized version of Kaucher arithmetic proposed in [ŽŽ05] assumes regularity of the dependency between variables. The main difference from the underestimates to inner interval arithmetic concerns underestimating of the results of multiplication. In the underestimation assuming regularity of the dependency between variables, multiplication is defined as:

$$\underline{\overline{x}} \times_{under} \underline{\overline{y}} = \begin{cases} \left[\min(\underline{x}\overline{y}, \overline{x}\underline{y}), \mu(\underline{x}, \overline{x}, \underline{y}, \overline{y})\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} > 0 \text{ or } \underline{\overline{x}} < 0, \underline{\overline{y}} < 0, \\ \left[\mu(\underline{x}, \overline{x}, \overline{y}, \underline{y}), \max(\underline{x}\underline{y}, \overline{x}, \overline{y})\right], & \underline{\overline{x}} > 0, \underline{\overline{y}} < 0 \text{ or } \underline{\overline{x}} < 0, \underline{\overline{y}} > 0, \\ \left[\mu(\underline{x}, \overline{x}, \overline{y}, \underline{y}), \mu(\underline{x}, \overline{x}, \underline{y}, \overline{y})\right], & \text{otherwise}, \end{cases}$$

where

$$\mu(x_1, x_2, y_1, y_2) = \begin{cases} x_2 y_1, & \frac{(x_2-x_1)y_2 - x_1(y_2-y_1)}{2(x_2-x_1)(y_2-y_1)} > 1, \\ x_1 y_2, & \frac{(x_2-x_1)y_2 - x_1(y_2-y_1)}{2(x_2-x_1)(y_2-y_1)} < 0, \\ \frac{(x_2 y_2 - x_1 y_1)^2}{4(x_2-x_1)(y_2-y_1)}, & \text{otherwise.} \end{cases}$$

The exact range of function values is between the results of overestimating interval arithmetic and underestimating interval arithmetic.

6 Non-Random Methods for the Estimation of Ranges of Functions

Estimates of the ranges of function values estimated from the results of standard interval arithmetic and inner interval arithmetic are investigated in [Žil06]. There, balanced interval arithmetic (BIA) is defined as the weighted mean of the resulting standard and inner intervals of the function:

$$pc \times \underline{\overline{f}}\left(\underline{\overline{X}}\right) + (1 - pc) \times \underline{\overline{f}}_{in}\left(\underline{\overline{X}}\right), \tag{3}$$

where the predefined coefficient $0 \leq pc \leq 1$ defines the balance between standard and inner resulting intervals. Similarly, it is possible to balance standard and inner intervals in every interval operation defining balanced in every operation interval arithmetic (BIEOIA):

$$\underline{\overline{x}} \circ_b \underline{\overline{y}} = pc \times \left(\underline{\overline{x}} \circ \underline{\overline{y}}\right) + (1 - pc) \times \left(\underline{\overline{x}} \circ_{in} \underline{\overline{y}}\right). \tag{4}$$

The other way to estimate the ranges of function values investigated in [Žil06] was called scaled interval arithmetic (SIA), which is acquired from the standard interval by scaling its radius:

$$\left[\left(\underline{f}\left(\underline{\overline{X}}\right) + \overline{f}\left(\underline{\overline{X}}\right)\right)/2 \pm pc \times \left(\overline{f}\left(\underline{\overline{X}}\right) - \underline{f}\left(\underline{\overline{X}}\right)\right)/2\right]. \tag{5}$$

Similarly as with BIEOIA, it is possible to scale the radii of standard intervals in every interval operation defining scaled in every operation interval arithmetic (SIEOIA):

$$\underline{\overline{x}} \circ_s \underline{\overline{y}} = \left[\left(\underline{\underline{\overline{x}} \circ \underline{\overline{y}}} + \overline{\underline{\overline{x}} \circ \underline{\overline{y}}}\right)/2 \pm pc \times \left(\overline{\underline{\overline{x}} \circ \underline{\overline{y}}} - \underline{\underline{\overline{x}} \circ \underline{\overline{y}}}\right)/2\right]. \tag{6}$$

BIA, BIEOIA, SIA and SIEOIA provide wider or narrower ranges depending on the predefined coefficient. The values used for the predefined coefficient depend on the balance required between tightness of resulting intervals and the probability that resulting intervals contain all the values of the function. When the predefined coefficient is equal to 1, the results of BIA, BIEOIA, SIA and SIEOIA are equal to the results of standard interval arithmetic. When the predefined coefficient is equal to 0, the results of BIA and BIEOIA are equal to the results of inner interval arithmetic, and the results of SIA and SIEOIA are intervals with zero width and centres equal to the centres of resulting standard intervals.

The ranges of the values of the functions in random regions have been estimated using different ways proposed in [Žil06] with different values of the predefined coefficient and using BRIA with different values of predefined probability of standard interval arithmetic operations. Two criteria have been used in comparison of estimates of ranges: the success rate and the mean ratio of widths of estimated ranges and bounds evaluated using standard interval arithmetic.

The success rate shows which part of the ranges of the values of the given function over random regions is estimated successfully using the given way with given predefined coefficient or predefined probability. Successful estimation means estimation of ranges which enclose all values of the function in the region. When estimated ranges for the values of the function are used in global optimization, the success rate determines the reliability of the global optimization algorithm. The algorithm is more reliable when the success rate is higher.

The mean ratio of widths shows how estimated ranges are tighter than bounds evaluated using standard interval arithmetic. As estimated ranges can not be wider than bounds evaluated using standard interval arithmetic, the mean ratio of 1.0 means that the widths of estimated ranges are equal to the widths of bounds evaluated using standard interval arithmetic. When estimated ranges of the values of the function are used in global optimization, the mean ratio of widths determines the speed of the global optimization algorithm. The mean ratio of widths is smaller when estimated ranges are tighter and subregions are discarded earlier. So, the algorithm is faster when the mean ratio of widths is smaller.

The estimation of ranges is more reliable when the success rate grows earlier, therefore it is more reliable when the curve representing the success rate depending on the value of predefined probability or coefficient is higher. The ranges are tighter when the mean ratio of widths grows later, therefore the estimated ranges are more tighter when the curve representing the mean ratio of widths depending on the value of predefined probability or coefficient is lower. However both criteria depend on the predefined probability or coefficient and are related to each other. The relationship between the success rate and mean ratio of widths shown in [Žil06] is more informative. The estimation is more reliable with tighter ranges when the curve representing the relationship is higher. Moreover the value of the mean ratio of widths when the success rate reaches 1.0 shows what possible improvement against standard interval bounds may be reached using estimates of the ranges of the function values. If the value of the mean ratio of widths is 70%, improvement of 30% may be reached. The results of the experiments summarized in Table 1 show that ranges estimated using BIA compete with ranges estimated using BRIA. BIEOIA is a little bit less promising, SIA is even less promising, and SIEOIA is not promising. It is important to note that these ways require much less computations than BRIA.

Table 1. Possible improvement against standard interval bounds when using estimates of ranges for test functions of global optimization

test function	BRIA (2)	BIA (3)	BIEOIA (4)	SIA (5)	SIEOIA (6)
Multidimensional scaling	30%	30%	15%	10%	-
Separation	60%	60%	60%	60%	60%
Paviani	30%	40%	40%	40%	-
Goldstein and Price	40%	-	-	-	-
Six Hump Camel Back	15%	15%	-	-	-
Shekel 5, 7, 10	10%	10%	-	-	-
Levy 4, 5, 6, 7	5-15%	5-15%	5-15%	-	-

7 Conclusions

In this survey methods to estimate ranges of functions using standard interval arithmetic and inner interval arithmetic have been reviewed. Balanced random interval arithmetic proves to be promising but expensive. Computationally cheaper non-random balanced interval arithmetic is less tight but may be a good alternative. Although estimates are experimentally investigated on a several test functions for global optimization, there is a lack of results of applying of them to solve practical global optimization problems. This would be most promising direction of further research in the area.

Acknowledgement. The authors wish to acknowledge the financial support through the Royal Society/NATO Postdoctoral Fellowship for the research "Stochastic Interval Methods for Process Optimisation", NATO Reintegration Grant EAP.RIG.981300 and Lithuanian State Science and Studies Foundation.

References

[AL01] Alt, R., Lamotte, J.L.: Experiments on the evaluation of functional ranges using random interval arithmetic. Mathematics and Computers in Simulation, **56**(1), 17–34 (2001)

[Dar57] Darling, D.A.: The Kolmogorov-Smirnov, Cramér-von Mises tests. The Annals of Mathematical Statistics, **28**(4), 823–838 (1957)

[Han78a] Hansen, E.: A global convergent interval analytic method for computing and bounding real roots. BIT **18**, 415–424 (1978)

[Han78b] Hansen, E.: Global optimization using interval analysis – the multidimensional case. Numerische Mathematik **34**, 247–270 (1978)

[HW03] Hansen, E., Walster, G.: Global Optimization Using Interval Analysis. 2nd ed. Marcel Dekker, New York (2003)

[JK95] Johnson, N.L., Kotz, S.: Continuous Univariate Distributions. 2nd ed. John Wiley, New York (1994-1995)

[Kau77] Kaucher, E.: Über Eigenschaften und Anwendungsmöglichkeiten der erweiterten Intervall-rechnung des hyperbolische Fastkörpers über R^n. Computing, Suppl. **1**, 81–94 (1977)

[KNZ96] Kreinovich, V., Nesterov, V., Zheludeva, N.: Interval methods that are guaranteed to underestimate (and the resulting new justification of Kaucher arithmetic). Reliable Computing, **2**(2), 119–124 (1996)

[LTG$^+$01] Lerch, M., Tischler, G., von Gudenberg, J.W., Hofschuster, W., Krämer, W.: The Interval Library filib++ 2.0 – Design, Features and Sample Programs. Preprint 2001/4, Universität Wuppertal (2001)

[Mat96] Mathar, R.: A hybrid global optimization algorithm for multidimensional scaling. In: Klar, R., Opitz, O. (eds) Classification and Knowledge Organization. Springer, Berlin, 63–71 (1996)

[Moo66] Moore, R.E.: Interval Analysis. Prentice-Hall (1966)

[Moo77] Moore, R.E.: A test for existence of solutions to non-linear systems. SIAM Journal on Numerical Analysis **14**, 611–615 (1977)

[PFTW92] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. 2nd ed. Cambridge University Press (1992)

[Ske74] Skelboe, S.: Computation of rational interval functions. BIT **14**, 87–95 (1974)

[Sun01] SUN Microsystems: C++ Interval Arithmetic Programming Reference. Forte Developer 6 update 2 (Sun WorkShop 6 update 2). SUN Microsystems (2001)

[ŽB03] Žilinskas, J., Bogle, I.D.L.: Evaluation ranges of functions using balanced random interval arithmetic. Informatica, **14**(3), 403–416 (2003)

[ŽB04] Žilinskas, J., Bogle, I.D.L.: Balanced random interval arithmetic. Computers & Chemical Engineering, **28**(5), 839–851 (2004)

[Žil05] Žilinskas, J.: Comparison of packages for interval arithmetic. Informatica, **16**(1), 145–154 (2005)

[Žil06] Žilinskas, J.: Estimation of functional ranges using standard and inner interval arithmetic. Informatica, **17**(1), 125–136 (2006)

[ŽŽ05] Žilinskas, A., Žilinskas, J.: On underestimating in interval computations. BIT Numerical Mathematics, **45**(2), 415–427 (2005)

[ŽŽ06] Žilinskas, A., Žilinskas, J.: On efficiency of tightening bounds in interval global optimization. Lecture Notes in Computer Science, **3732**, 197–205 (2006)

Pseudo-Boolean Optimization in Case of an Unconnected Feasible Set

Alexander Antamoshkin and Igor Masich

Siberian State Aerospace University, Krasnoyarsk, Russia nii_suvpt@wave.krs.ru

1 Introduction

Unconstrained pseudo-Boolean optimization is an issue that studied enough now. Algorithms that have been designed and investigated in the area of unconstrained pseudo-Boolean optimization are applied successfully for solving various problems. Particularly, these are local optimization methods [AL97, AM04a, PS82] and stochastic and regular algorithms based on local search for special function classes [ASS90, BSV02, WW02]. Moreover, there is a number of algorithms for optimization of functions given in explicit form: Hammer's basic algorithm that was introduced in [HR68] and simplified in [BH02]; algorithms for optimization of quadratic functions [AFLS01, FH00, HS89], etc. Universal optimization methods are also used successfully: genetic algorithms, simulated annealing, tabu search [Gol89, Sch95].

If there are constraints on the binary variables, one of ways to take into account it as is well known is construction and optimization of an generalized penalty function. Shortcoming of this approach is existence of a large number of local optima of the generalized function that will be shown below. If an accessible region is a connected one then this issue can be partly eliminated, for example, by using local search with a stronger system of neighborhoods. Extension of search neighborhood reduces the number of local optima which locate mainly not far one from another in this case.

If an accessible region is unconnected then using penalty functions and unconstrained optimization methods get complicated because the accessible region is usually too small with respect to optimization space. That makes difficult searching feasible solution.

In this chapter some heuristic procedures of boundary point search are considered for a constrained pseudo-Boolean optimization problem. Experimental investigation of the algorithms are described, recommendations for their applying are given.

2 Basic Definitions

Consider some definitions that are necessary for describing optimization algorithm work [AM04b, ASS90].

- A *pseudo-Boolean function* is called a real function of binary variables: $f\colon B_2^n \to R^1$, where $B_2 = \{0,1\}$, $B_2^n = B_2 \times B_2 \times \cdots \times B_2$.
- Points $X^1, X^2 \in B_2^n$ are called *k-neighboring* points if they differ in k coordinate value, $k = \overline{1,n}$. 1-neighboring points are called simply neighboring.
- The set $O_k(X)$, $k = \overline{0,n}$, of all point of B_2^n, that are k-neighboring to a point X, is called a *k-th level* of the point X.
- A point set $W(X^0, X^l) = \{X^0, X^1, \ldots, X^l\} \subset B_2^n$ is called a *path* between points X^0 and X^l if for $\forall i = 1, \ldots, l$ the point X^i is a neighboring to X^{i-1}.
- A set $A \subset B_2^n$ is called a *connected set* if for $\forall X^0, X^l \in A$ the path $W(X^0, X^l) \subset A$ exists.
- A point $X^* \in B_2^n$, for which $f(X^*) < f(X)$, $\forall X \in O_1(X^*)$, is called a *local minimum* of pseudo-Boolean function f.
- A pseudo-Boolean function that has an unique local minimum is called an *unimodal* on B_2^n function.
- An unimodal function f is called *monotonic* on B_2^n if $\forall X^k \in O_k(X^*), k = \overline{1,n}$: $f(X^{k-1}) \le f(X^k), \forall X^{k-1} \in O_{k-1}(X^*) \cup O_1(X^k)$.

  Example: A polynomial of binary variables

$$\varphi(x_1, \ldots, x_n) = \sum_{j=1}^{m} a_j \prod_{i \in \beta_j} x_i,$$

where $\beta_j \subset 1, \ldots, n$, is an unimodal and monotonic pseudo-Boolean function for $a_j \ge 0$, $j = \overline{1,m}$.

3 Problem Statement

Consider the problem of the following form

$$C(X) \to \max_{X \in S \subset B_2^n}, \tag{1}$$

where $C(X)$ is a monotonically increasing from X^0 pseudo-Boolean function, $S \subset B_2^n$ is a certain subregion of the binary variable space; it is determined by a given constraint system, for example:

$$A_j(X) \le H_j, j = \overline{1,m}. \tag{2}$$

In the general case a set of feasible solutions S is a unconnected set.

4 Properties of a Set of Feasible Solutions of the Problem

- A point $Y \in A$ is a *boundary point* of the set A if there exist $X \in O_1(Y)$ for which $X \notin A$.
- A point $Y \in O_i(X^0) \cap A$ is called a *limiting point* of the set A with the basic point $X^0 \in A$ if for $\forall X \in O_1(Y) \cap O_{i+1}(X^0)$ $X \notin A$ holds.
- A constraint that determine a subregion of the binary variable space is called *active* if the optimal solution of the conditional problem do not coincide with the optimal solution of the appropriate problem without taking the constrain into account. Consider some properties of a feasible solution set [AM04b].
- If the object function is a monotonic unimodal function and the constraint is active then the optimal solution of the problem 1 is a point that belongs to the subset of limiting points of the feasible solution set S with the basic point X^0 in which the object function takes the lowest value:

$$C(X^0) = \min_{X \in B_2^n} C(X).$$

- Consider a problem 1 with a constraint 2. If the constraint function 2 is an unimodal function then the set S of feasible solutions of the problem 1 is a connected set.
- A number of limiting points of a connected feasible solution set for the problem 1 equals $s \leq s_{\max} = C_n^{[n/2]}$, where $[n/2]$ is the integer part of the value $n/2$. In case of $s = s_{\max}$ all limiting point belong to $O_{n/2}(X^0)$ if n is even and to $O_{(n-1)/2}(X^0)$ (or $O_{(n+1)/2}(X^0)$) if n is odd.

5 Transfer to Unconstrained Optimization

One of ways to take into account constraints in conditional problems is construction a generalized penalty function.

Consider the generalized function

$$F(X) = C(X) - r \times \sum_{j=1}^{m} \max\{0, A_j(X) - H_j\} \tag{3}$$

for a problem 1 with constraints 2.

In this case the following conclusion occurs [AM03].

Conclusion 1 *The limiting points of the set S of the feasible points of the primary problem 1-2 with the basic point X^0 are point of local maxima for generalized penalty function 3 with parameter r satisfying the condition*

$$r > \frac{C(Z) - C(X')}{\sum_{j=1}^{m} \max\{0, A_j(Z) - H_j\}}$$

for every limiting point $X' \in O_k(X^0)$ and every point $Z \in O_{k+1}(X_0) \cap O_1(X')$.

So, a problem 1-2 is identical to the problem

$$F(X) \to \max_{X \in B_2^n}, \tag{4}$$

in which a number of local maxima (in case of connectivity of the feasible solution set) $s \leq s_{\max} = C_n^{[n/2]}$. In general case, when feasible set is not connected, a number of local maxima theoretically can amount to 2^{n-1}.

The problem 4 is solved by known search methods: local search, genetic algorithms, simulated annealing, etc.

The essential shortcoming of this approach is loss of the monotonicity property for an object function $C(X)$. By addition of even simple (for example, linear) constraints the generalized function becomes a polymodal nonmonotonic function with exponential number of local maxima.

6 Heuristic Algorithms for Boundary Point Search

For any heuristic of boundary point search we will consider a pair of algorithms – primary and dual. A primary algorithm starts search from the feasible area and moves in a path of increasing of the objective function until it finds a limiting point of feasible area. Otherwise, a dual algorithm keeps search in the unfeasible area in a path of decreasing of the objective function until it finds some available solution.

Total scheme of primary search algorithm

1. Put $X_1 = X^0$, $i = 1$.
2. In accordance with a rule we choose $X_{i+1} \in O_i(X^0) \cap O_1(X_i) \cap S$. If there are no such point then go to 3; else $i = i + 1$ and repeat the step.
3. $X_{opt} = X_{i+1}$.

Total scheme of dual search algorithm

1. Put $X_1 \in O_n(X^0)$, $i = 1$.
2. In accordance with a rule we choose $X_{i+1} \in O_{n-i}(X^0) \cap O_1(X_i) \cap S$. If $X_{i+1} \in S$ then go to 3; else $i = i + 1$ and repeat the step.
3. $X_{opt} = X_{i+1}$.

From these schemes we can see that a primary algorithm finds a limiting point of the feasible area, but a dual algorithm finds a boundary point which may be not a limiting one. So a primary algorithm finds a better solution then dual in the most cases for problems with a connected set of feasible solutions. If we will use a primary algorithm for a problem with a unconnected feasible area then solution received in result may be far from the optimal because feasible and unfeasible solutions will rotate in a path of increasing of the objective function. For these cases a dual algorithm is more useful, because this rotation do not play any role for it. For improving the solution that given by the dual algorithm, it is recommended to apply the corresponding primary algorithm. Such improving is very significant in practice.

Boundary point search algorithms considered below differs by only a rule of choice of a next point in step 2 of the total schemes.

Search Rules

Rule 1. Random search of boundary points (RSB)

A point X_{i+1} is chosen by random way. Each point in the next step can be chosen with equal probabilities. For real-world problems these probabilities can be not equal but they are calculated in the basis of problem specific before search starts.

Rule 2. Greedy algorithm

A point X_{i+1} is chosen according to the condition

$$\lambda(X_{i+1}) = \max_{j} \lambda(X^j),$$

where $X^j \in O_i(X^0) \cap O_1(X_i) \cap S$ for a primary algorithm and $X^j \in O_{n-i}(X^0) \cap O_1(X_i)$ for a dual one.

The function $\lambda(X)$ is chosen from problem specific, for example:

the objective function $\lambda(X) = C(X)$,

specific value $\lambda(X) = C(X)/A(X)$ (for only constraint) and so on.

Rule 3. Adaptive random search of boundary points (ARSB)

A point X_{i+1} is chosen by random way in accordance with a probability vector

$$P^i = (p_1^i, p_1^i, \ldots, p_J^i),$$

where J is the number of points from which choice is made.

$$p_j^i = \frac{\lambda(X^j)}{\sum_{l=1}^{J} \lambda(X^l)}, j = \overline{1, J},$$

where $X^j \in O_i(X^0) \cap O_1(X_i) \cap S$ for a primary algorithm and $X^j \in O_{n-i}(X^0) \cap O_1(X_i)$ for a dual one. ARSB is a addition to the greedy algorithm.

Rule 4. Modificated random search of boundary points (MRSB)

A point X_{i+1} is chosen according to the condition

$$\lambda(X_{i+1}) = \max_{r} \lambda(X^r),$$

where X^r are points chosen accordance with the rule 1, $r = \overline{1, R}$; R is a algorithm parameter.

A greedy algorithm is regular algorithm, so it finds equivalent solutions under restart from a certain point. Other algorithms can be started several times and the best solution can be selected from found solutions. Run time of each algorithm start is constrained by

$$T \leq \frac{n(n+1)}{2},$$

but average run time of a greedy algorithm and ARSB is significant larger then for other because they look over all point of the next level in each step in distinction from RSB and MRSB which look over only one and R points correspondingly in each step in the dual scheme.

Further we consider applying the described algorithms for a real-world problem which has been solved by the authors for an aluminium plant in Krasnoyarsk. Formalization of the problem as a pseudo-Boolean optimization problem has been realized also by the authors. The obtained model is characterized by large dimension and an unconnected set of feasible solutions.

7 Optimization of Loading of Foundry Branches Capacities

Production of different kind is produces in foundry branches (FB). There is specialization in every FB by kind of production which can be produced by its foundry machines (FM). There is a quantity of orders for production. To each order there corresponds volume, a kind of production and term of performance. The kind of production is characterized by productivity for change (only 3 changes in a day). Replacement made on FM production demands its recustomizing borrowing one change. It is necessary to load thus FB capacities that orders were carried out all, production was made in regular more intervals in time, and the number of recustomizings of FM equipment was minimal.

Input data:

I is a number of days for planning;

J is a number of FB;

K_j is a number of FM in j-th FB, $j = \overline{1,J}$;

L is a number of orders for production that produces on FM (and corresponding number of production kinds);

V_l is productivity of l-th kind of production for change on FM, $l = \overline{1,L}$;

T_l is term of performance of l-th order (for production of l-th kind) on FM, $l = \overline{1,L}$;

W_l is volume of l-th order (for production of l-th kind) on FM,$l = \overline{1,L}$;

z_{jl} characterizes specialization of FB:

$$z_{jl} = \begin{cases} 1 & \text{FM of } j\text{-th FB can make production of } l\text{-th kind,} \\ 0 & \text{otherwise;} \end{cases}$$

α is the factor of rigidity of restriction on demand of uniformity of production on days, $0 < \alpha < 1$.

Variables:

For model construction introduce following binary variables:

$$Y = \{y_{ijkl}\} \in B_2^n,$$

$$X = \{x_{ijkl}\} \in B_2^n,$$

where $B_2 = \{0, 1\}$, $B_2^n = B_2 \times B_2 \times \cdots \times B_2$, is a set of binary variables.

$$y_{ijkl} = \begin{cases} 1 & \text{production of } l\text{-th kind is made in } i\text{-th day on } k\text{-th FM} \\ & \text{of } j\text{-th FB,} \\ 0 & \text{otherwise;} \end{cases}$$

$$x_{ijkl} = \begin{cases} 1 & \text{production of } l\text{-th kind is started to make in } i\text{-th day} \\ & \text{on } k\text{-th FM of } j\text{-th FB,} \\ 0 & \text{otherwise.} \end{cases}$$

Total dimension of a binary vector Y (and X) is

$$n = I \times L \times \sum_{j=1}^{J} K_j.$$

Remarks:

$$1.\ x_{ijkl} \le y_{ijkl} \forall i, j, k, l.$$

$$2.\ x_{ijkl} = y_{ijkl} \times (1 - y_{i-1,jkl}) \forall i, j, k, l (y_{0jkl} = 0 \forall i, j, k, l). \tag{5}$$

7.1 Optimization Model

1. *The objective function and the main constraints*

$$C(X) \to \min, \tag{6}$$

$$A_l^1(Y) \ge W_l, l = \overline{1, L}, \tag{7}$$

$$A_i^2(Y) \ge \alpha \times W^I, i = \overline{1, I}, \alpha \in (0, 1), W^I = \frac{\sum_{l=1}^{L} W_l}{I}, \tag{8}$$

where

$$C(X) = \sum_{i=1}^{I} \sum_{j=J}^{I} \sum_{k=1}^{K_j} \sum_{l=1}^{L} x_{ijkl} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K_j} \sum_{l=1}^{L} y_{ijkl} \times (1 - y_{i-1,jkl}),$$

$$A_l^1(Y) = \sum_{i=1}^{T_l}\sum_{j=1}^{J}\sum_{k=1}^{K_j} V_l \times y_{ijkl} \times (2 + y_{i-1,jkl}),$$

$$A_i^2(Y) = \sum_{j=1}^{J}\sum_{k=1}^{K_j}\sum_{l=1}^{L} V_l \times y_{ijkl} \times (2 + y_{i-1,jkl}),$$

$$y_{0jkl} = 0, \forall j, k, l.$$

2. *The additional constraints*

$$\sum_{l=1}^{L} y_{ijkl} \leq 1(\sum_{l=1}^{L} x_{ijkl}), i = \overline{1,I}, j = \overline{1,J}, k = \overline{1,K_j}, \tag{9}$$

$$y_{ijkl} \leq z_{jl}(x_{ijkl} \leq z_{jl}), i = \overline{1,I}, j = \overline{1,J}, k = \overline{1,K_j}, l = \overline{1,L}. \tag{10}$$

7.2 Model Properties

1. There are two spaces of binary variables (denote their by B^X and B^Y) corresponding vectors X and Y. For each point $Y \in B^Y$ a unique point $X \in B^X$ corresponds, components of which are determined by relation 5. Several points $Y \in B^Y$ (with different value of constraint function) can correspond to the point $X \in B^X$.
2. The objective function 6 is linear and unimodal monotonic in the space B^X with the minimum point $X^0 = (0, \ldots, 0)$. In the space B^Y the objective function is quadratic and unimodal nonmonotonic with the minimum point $Y^0 = (0, \ldots, 0)$.
3. The constraint function 7 and 8 in the space B^Y are quadratic and unimodal monotonic pseudo-Boolean functions with the minimum point $Y^0 = (0, \ldots, 0)$. In the space B^X the constraint functions unequivocally are not certain.
4. The feasible solution set in the spaces B^X and B^Y is limited from above by $I \times \sum_{j-1}^{L} K_j$-th level of the minimum point (X^0 and Y^0) according to the constraint 9. In the space B^Y this level corresponds to the case when production is produces on each BM in every day.
5. The feasible solution set is an unconnected set in general case (in the space B^Y).

Thus the problem solution is defined completely by the variables Y, but it does not hold for the variables X. But the objective function from X has good constructive properties so that optimum search on X is more efficient then on Y. As the constraint function 7, 8 from X are not defined, we should find values of these functions from the corresponding point Y. There are perhaps several such points

$$X \rightarrow Y_1, Y_2, \ldots, Y_H,$$

in some of them the solution may be feasible but in other not. As the constraint functions are monotonic here then for a certain X we should choose a such Y that belongs to the most possible level (with the most values of the functions):

$$Y = \arg \max_{Y_h, h=\overline{1,H}} \left(\sum_{i,j,k,l} y^h_{ijkl} \right),$$

$$\text{where} Y_h = (y_{1111}, \dots, y_{IJK_JL}).$$

One of algorithms of this transformation is presented below.
Algorithm 1 of transformation X to Y

```
1. Put $N_{jk} = 0$, $j = \overline{1,J}$, $k = \overline{1,K_j}$; $i = 1$.
2. For $j = \overline{1,J}$, $k = \overline{1,K_j}$, $l = \overline{1,L}$ do: if $x_{ijkl} = 1$ then $N_{jk} = l$.
3. For $j = \overline{1,J}$, $k = \overline{1,K_j}$, $l = \overline{1,L}$ do: if $N_{jk} = l$ then $y_{ijkl} = 1$ else $y_{ijkl} = 0$.
4. If $i < I$ then $i = i + 1$ and to 2.
```

At the same time the solution Y received from the found best vector X_{opt} by this way may corresponds to situation when a quantity of let out production is more higher then the requisite value (this does not contradict to the constructed model but this can influent on uniformity of capacities loading which is optimized on the next stage). So when the first stage of search has ended, we should define Y_{opt} by the rule

$$X_{opt} \to Y_1, Y_2, \dots, Y_H,$$

$$Y_{opt} = \arg \min_{Y_h:\ A^1_l(Y_h) \geq W_l, l=\overline{1,L}} \left(\sum_{i,j,k,l} y^h_{ijkl} \right).$$

In this case the transformation algorithm has some differences from the previous.
Algorithm 2 of transformation X to Y

```
1. Put $N_{jk} = 0$, $j = \overline{1,J}$, $k = \overline{1,K_j}$.
1a. Put $y_{ijkl} = 0$, $i = \overline{1,IJ}$, $j = \overline{1,J}$, $k = \overline{1,K_j}$, $l = \overline{1,L}$; $i = 1$.
2. For $j = \overline{1,J}$, $k = \overline{1,K_j}$, $l = \overline{1,L}$ do: if $x_{ijkl} = 1$ then $N_{jk} = l$.
3. For $j = \overline{1,J}$, $k = \overline{1,K_j}$, $l = \overline{1,L}$ do: if $N_{jk} = l$ and $A^1_l(Y) < W_l$ then $y_{ijkl} = 1$.
4. If $i < I$ then $i = i + 1$ and to 2.
```

Here the condition $A^1_l(Y) < W_l$ is added in the step 3, and the step 1a is added also for possibility of this calculation. There are no any needs in this transformation during the search. It is necessary only for determining result Y_{opt}.

7.3 Optimization Algorithms

The dual algorithms RSB, greedy and MRSB have been used for solving the problem. The algorithm ARSB has not been considered for this problem because of its excessive large run time by frequent start. One start of ARSB can very rarely give a solution which is better then the solution given by the greedy algorithm. The start point of search is the point of the unconstraint minimum of the objective function $X^0 = (0, \dots, 0)$. A found solution has been improved by the corresponding primary algorithm.

Moreover, the problem has been solved by the genetic algorithm (GA). To realize GA we have chosen a scheme that effective worked for multiple solving other combinatorial optimization problems.

Results of the experiments shows that the most effective algorithms (by precision and run time) from the considered ones for this problem are the greedy algorithm and MRSB. The other algorithms under hard constraints on the variables do not find any accessible solution at all. It is a sequel of problem specific: a large amount of different constraints and, as a result, a comparative small accessible region.

The average results of solving 10 problems of month planning capacity loading are presented in Table 1. The average values of input data:

$$I = 31, J = 3, K_1 = 12, K_2 = 9, K_3 = 7, L = 36, \alpha = 0.5,$$

$$V_l \in [40, 50], W_l \in [20, 25000].$$

Herewith the total dimension of the binary vector is $n = 31248$.

The number of algorithm starts L has been chosen so that the run time nearly equals to the run time of one start of the greedy algorithm. In this case the run time equals to $T \approx 8 \times 10^5$.

Table 1. Results of solving the problem

Algorithm	Number of starts L	Found solution C_{opt}
RSB	2000	Not found
Greedy	1	49
MRSB, $R = 1000$	12	47
MRSB, $R = 100$	60	52
MRSB, $R = 10$	200	Not found
GA [1]	–	Not found

MRSB is a more flexible procedure in compare with the greedy algorithm as the first one allows to select the parameters L and R that influence on

[1] GA parameters: tournament selection with the tournament value 5, population value 100, the largest number of generations 8000, mutation probability 0.0001, uniform crossover

algorithm run time and solution precision. The greedy algorithm does not allow that possibility and run time may be overmuch large under high dimensions. What about their efficiency, precision of the found solutions differs unessentially under nearly equivalent run time.

8 Conclusion

The algorithms of boundary point search shows high efficiency for solving the pseudo-Boolean optimization problem with unconnected accessible region. The most efficient algorithms for the considered problem are the dual algorithms MRSB and greedy.

References

[AFLS01] Allemand, K., Fukuda, K., Liebling, T.M., Steiner, E.: A polynomial case of unconstrained zero-one quadratic optimization. Math. Program., **Ser. A 91**, 49–52 (2001)

[AL97] Aarts, E.H.L., Lenstra, J.K.: Local Search in Combinatorial Optimization. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons Ltd, England (1997)

[AM03] Antamoshkin, A.N., Masich, I.S.: Greedy algorithm and local search for conditional pseudo-Boolean optimization. Electronic journal "Investigated in Russia", **177**, 2143–2149 (2003) http://zhurnal.ape.relarn.ru/articles/2003/177.pdf

[AM04a] Antamoshkin, A.N., Masich, I.S.: Comparative efficience of two schemes of local search for optimization of pseudo-Boolean functions. Electronic journal "Investigated in Russia", **51**, 544–546 (2004) http://zhurnal.ape.relarn.ru/articles/2004/051.pdf

[AM04b] Antamoshkin, A.N., Masich, I.S.: Unimprovable algorithm of conditional optimization of monotonic pseudo-Boolean functions. Electronic journal "Investigated in Russia", **64**, 703–708 (2004) http://zhurnal.ape.relarn.ru/articles/2004/064.pdf

[ASS90] Antamoshkin, A.N., Saraev, V.N., Semenkin, E.S.: Optimization of unimodal monotone pseudoboolean functions. Kybernetika, **26**(5), 432–441 (1990)

[BH02] Boros, E., Hammer, P.L.: Pseudo-Boolean optimization. Discrete Applied Mathematics, **123**(1-3), 155–225 (2002)

[BSV02] Björklund, H., Sandberg, S., Vorobjov, S.: Optimization on Completely Unimodal Hypercubes. Technical report 2002-018 (2002)

[FH00] Foldes, S., Hammer, P.L.: Monotone, Horn and quadratic pseudo-Boolean functions. Journal of Universal Computer Science, **6**(1), 97–104 (2000)

[Gol89] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading MA (1989)

[HR68] Hammer (Ivanescu), P.L., Rudeanu, S.: Boolean Methods in Operations Research and Related Areas. Springer-Verlag, Berlin Heidelberg New York (1968)

[HS89] Hammer, P.L., Simeone, B.: Quadratic functions of binary variables. In: Simeone, B. (ed) Combinatorial Optimization. Lecture Notes in Mathematics, vol. 1403. Springer, Heidelberg (1989)

[PS82] Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization. Prentice-Hall, Englewood Cliffs NJ (1982)

[Sch95] Schwefel, H.-P.: Evolution and Optimum Seeking. Whiley Publ., N.Y. (1995)

[WW02] Wegener, I., Witt, C.: On the Optimization of Monotone Polynomials by Simple Randomized Search Heuristics. Technical Report (2002)

Univariate Algorithms for Solving Global Optimization Problems with Multiextremal Non-differentiable Constraints*

Yaroslav D. Sergeyev[1,2,**], Falah M.H. Khalaf[3], and Dmitri E. Kvasov[2,4]

[1] DEIS, University of Calabria, Via P. Bucci, 42C, 87036 Rende (CS), Italy yaro@si.deis.unical.it
[2] Software Department, N.I. Lobatchevsky State University, Nizhni Novgorod, Russia
[3] Department of Mathematics, University of Calabria, Italy falah@mat.unical.it
[4] Department of Statistics, University of Rome "La Sapienza", Italy kvadim@si.deis.unical.it

Summary. In this chapter, Lipschitz univariate constrained global optimization problems where both the objective function and constraints can be multiextremal and non-differentiable are considered. The constrained problem is reduced to a discontinuous unconstrained problem by the index scheme without introducing additional parameters or variables. It is shown that the index approach proposed by R.G. Strongin for solving these problems in the framework of stochastic information algorithms can be successfully extended to geometric algorithms constructing non-differentiable discontinuous minorants for the reduced problem. A new geometric method using adaptive estimates of Lipschitz constants is described and its convergence conditions are established. Numerical experiments including comparison of the new algorithm with methods using penalty approach are presented.

Key words: Global optimization, multiextremal constraints, geometric algorithms, index scheme.

1 Introduction

Let us consider the one-dimensional global optimization problem of the finding of the value f^* and a point x^* such that

$$f^* = f(x^*) = \min\{f(x) : x \in [a,b],\ g_j(x) \le 0,\ 1 \le j \le m\} \qquad (1)$$

where $f(x)$ and $g_j(x)$, $1 \le j \le m$, are multiextremal Lipschtiz functions (particularly, this means that they can be non-differentiable). We suppose

* This research was supported by the following grants: FIRB RBNE01WBBB, FIRB RBAU01JYPN, and RFBR 04-01-00455-a.

** Corresponding author

also that the objective function $f(x)$ and the constraints can be partially defined. This means that a constraint $g_{j+1}(x)$ is defined only at subregions where $g_j(x) \leq 0$ and $f(x)$ is defined only over subregions of $[a,b]$ where all the constraints are satisfied.

It is not easy to find a traditional algorithm for solving problem (1). For example, the penalty approach requires that $f(x)$ and $g_i(x)$, $1 \leq i \leq m$, are defined over the whole search interval $[a,b]$. At first glance it seems that at the regions where a function is not defined it can be simply filled in with either a big number or the function value at the nearest feasible point. Unfortunately, in the context of Lipschitz algorithms, incorporating such ideas can lead to infinitely high Lipschitz constants, causing degeneration of the methods and non-applicability of the penalty approach.

It is worthy to notice that not only problem (1) but univariate global optimization problems in general (and in contrast to one-dimensional local optimization problems that have been very well studied in the past) continue to attract attention of many researchers (see [CZ99, HJ95, Lam99, LS95, MSB96, Pij72, Ser98, SFP01, Str78, WC96]). These happens at least for two reasons. First, there exists a large number of applications where it is necessary to solve such problems (see [Bro58, DGMS95, DGMS96, HJ95, PKS87, RWPD85, SDGM99, Str78]). Second, there exist numerous approaches (see, e.g., [FP96, HP95, HT96, Mla92, PR90, Pin96, SG01, Str78, SS00, TZ89, Zhi91]) enabling to generalize to the multidimensional case the methods proposed for solving univariate problems.

Let us return now to problem (1) and study its internal properties. We designate subdomains of the interval $[a,b]$ corresponding to the set of constraints from (1) as

$$Q_1 = [a,b], \quad Q_{j+1} = \{x \in Q_j : g_j(x) \leq 0\}, \quad 1 \leq j \leq m, \tag{2}$$

$$Q_1 \supseteq Q_2 \supseteq \dots \supseteq Q_m \supseteq Q_{m+1}.$$

We introduce the number M such that

$$Q_M \neq \varnothing, \quad Q_{M+1} = Q_{M+2} \dots = Q_{m+1} = \varnothing. \tag{3}$$

If the feasible region of problem (1) is not empty then $Q_{m+1} \neq \varnothing$ and $M = m+1$. In the opposite case M indicates the last subset Q_j from (2) such that $Q_j \neq \varnothing$. Note that since the constraints $g_j(x)$, $1 \leq j \leq m$, are multiextremal, the admissible region Q_{m+1} and regions Q_j, $1 \leq j \leq m$, can be collections of several disjoint subregions. We suppose hereafter that all of them consist of intervals of a finite length.

In order to unify the description process we shall use the designation $g_{m+1}(x) \triangleq f(x)$. Now we can write the Lipschitz conditions for the functions $g_j(x), 1 \leq j \leq m+1$,

$$| g_j(x') - g_j(x'') | \leq L_j \, | x' - x'' |, \quad x', x'' \in Q_j, \quad 1 \leq j \leq m+1, \tag{4}$$

$$0 < L_j < \infty, \quad 1 \leq j \leq m+1. \tag{5}$$

A promising approach called the *index scheme* has been proposed in [Str84] (see also [BS02, SM95, SM86, SS00]) in combination with information stochastic Bayesian algorithms for solving problem (1), (4), (5). An important advantage of the index scheme is that it does not introduce additional variables and/or parameters by opposition to classical approaches in [Ber96, Ber99, HP95, HT96, NW99]. It has been recently shown in [SFP01] that the index scheme can be also successfully used in combination with the Branch-and-Bound approach if the Lipschitz constants $L_j, 1 \leq j \leq m+1$, from (4), (5) are known a priori.

However, in practical applications the Lipschitz constants $L_j, 1 \leq j \leq m+1$, are very often unknown. Thus, the problem of their estimating arises inevitably. If there exists an additional information allowing us to obtain a priori fixed constants K_j, $1 \leq j \leq m+1$, such that

$$L_j < K_j < \infty, \quad 1 \leq j \leq m+1,$$

then the algorithm IBBA from [SFP01] can be used. In this chapter, we consider the case where there is no any additional information about the Lipschitz constants. A new global algorithm adaptively estimating the Lipschitz constants (GEA) during the search is proposed and studied both theoretically and numerically.

The rest of the chapter is organized as follows. The index scheme and a theoretical material required for describing the new method is given in Sect. 2. The new algorithm is described in Sect. 3. Finally, Sect. 4 contains computational results and a brief conclusion.

2 Theoretical Background

Let us describe the index scheme briefly. By using the designation (2), (3) we can rewrite problem (1), (4), (5) as the problem of finding a point x^*_M and the corresponding value g^*_M such that

$$g^*_M = g_M(x^*_M) = \min\{g_M(x) \; : x \in Q_M\}. \tag{6}$$

The values x^*_M, g^*_M coincide with the global solution of the problem (1) if $M = m+1$, i.e, when the original problem (1) is feasible. We associate with every point of the interval $[a, b]$ the *index*

$$\nu = \nu(x), \quad 1 \leq \nu \leq M,$$

which is defined by the conditions

$$g_j(x) \leq 0, \quad 1 \leq j \leq \nu - 1, \quad g_\nu(x) > 0, \tag{7}$$

where for $\nu = m+1$ the last inequality is omitted. We shall call *trial* the operation of evaluation of the functions $g_j(x)$, $1 \le j \le \nu(x)$, at a point x.

Thus, the index scheme considers constraints one at a time at every point where it has been decided to try to calculate the objective function $g_{m+1}(x)$. Each constraint $g_i(x)$ is evaluated at a point x only if all the inequalities

$$g_j(x) \le 0, \quad 1 \le j < i,$$

have been satisfied at this point. In its turn the objective function $g_{m+1}(x)$ is computed only for those points where all the constraints have been satisfied.

Let us introduce now an auxiliary function $\varphi(x)$ defined over the interval $[a,b]$ as follows

$$\varphi(x) = g_{\nu(x)} - \begin{cases} 0, \text{ if } & \nu(x) < m+1 \\ g^*_{m+1}, \text{ if } & \nu(x) = m+1 \end{cases} \tag{8}$$

where g^*_{m+1} is the solution to problem (1) and to problem (6) in the case $M = m+1$.

The following proposition (see [Str84, SM86, SS00]) describes important properties of the function $\varphi(x)$:

Proposition 1. *The function $\varphi(x)$ from (8) enjoys the following properties:*

i. if a point $x \in [a,b]$ has an index $\nu(x) < m+1$ then $\varphi(x) > 0$;
ii. if a point $x \in [a,b]$ has an index $\nu(x) = m+1$ then $\varphi(x) \ge 0$;
iii. if a point $x \in [a,b]$ has an index $\nu(x) = m+1$ and $g_{m+1}(x) = g^*_{m+1}$ then it follows $\varphi(x) = 0$;
iv. the function $\varphi(x)$ can have points of discontinuity at the points y such that for $x < y, x \in Q_j$, and for $x > y, x \in Q_i, i \ne j$, i.e., at the borders of the domains $Q_j, 1 < j \le m+1$.

Thus, the global minimizer of the original constrained problem (1) coincides with the solution x^* of the following unconstrained discontinuous problem

$$\varphi(x^*) = \min\{\varphi(x) : x \in [a,b]\}, \tag{9}$$

in the case $M = m+1$ and $g_{m+1}(x^*) = g^*_{m+1}$. Obviously, the value g^*_{m+1} used in the construction (8) is unknown; it is adaptively estimated during the work of methods using the index scheme.

Suppose now that k trials have been executed at some points

$$a = x_0 < x_1 < \ldots < x_i < \ldots < x_k = b \tag{10}$$

and the index $\nu_i = \nu(x_i)$, $0 \le i \le k$, have been calculated in accordance with (7). Naturally, since the value g^*_{m+1} from (8) is unknown, it is not possible to evaluate the function $\varphi(x)$ for the points x having the index $\nu(x) = m+1$.

In order to overcome this difficulty, we introduced the function $\varphi_k(x)$ which is evaluated at the points x_i and gives us the values $z_i = \varphi_k(x_i)$, $0 \le i \le k$, as follows

$$\varphi_k(x) = g_{\nu(x)}(x) - \begin{cases} 0, & \text{if } \nu(x) < m+1 \\ z_k^*, & \text{if } \nu(x) = m+1 \end{cases} \tag{11}$$

where the values

$$z_k^* = \min\{g_{m+1}(x_i) : 0 \le i \le k,\ \nu_i = m+1\}, \tag{12}$$

estimates g_{m+1}^* from (8). It can be seen from (8), (11), and (12) that

$$\varphi_k(x_i) = \varphi(x_i)$$

if $\nu(x_i) < m+1$ and

$$0 \le \varphi_k(x_i) \le \varphi(x_i)$$

if $\nu(x_i) = m+1$. In addition, the function $\varphi_k(x)$ is non-positive at points x having the index $\nu(x) = m+1$ and the value $g_{m+1}(x) \le z_k^*$, i.e.,

$$\varphi_k(x) \le 0, \quad x \in \{x : g_{m+1}(x) \le z_k^*\}. \tag{13}$$

The points x_i, $0 \le i \le k$, form subintervals

$$[x_{i-1}, x_i] \subset [a,b],\ 1 \le i \le k,$$

and there exist the following three types of them:

i. intervals $[x_{i-1}, x_i]$ such that $\nu_{i-1} = \nu_i$;
ii. intervals $[x_{i-1}, x_i]$ such that $\nu_{i-1} < \nu_i$;
iii. intervals $[x_{i-1}, x_i]$ such that $\nu_{i-1} > \nu_i$.

It has been shown in [SFP01] that if the Lipschitz constants from (4), (5) are known, then for the function $\varphi_k(x)$ from (11) over every subinterval $[x_{i-1}, x_i]$ of the interval $[a,b]$ it is possible to construct a *discontinuous index minorant function* $\psi_i(x)$ with the following properties

$$\psi_i(x) \le \varphi_k(x), \quad x \in \overline{Q}_{\overline{\nu_i}},$$

where

$$\overline{Q}_{\overline{\nu_i}} = [x_{i-1},\, x_i] \cap Q_{\overline{\nu_i}}, \qquad \overline{\nu_i} = \max\{\nu_{x_{i-1}},\, \nu_{x_i}\}.$$

Note that the introduced notion of the discontinuous index minorant function is weaker than the usual definition of a minorant function. In fact, nothing is required with regard to behavior of $\psi_i(x)$ over $[x_{i-1}, x_i] \setminus Q_{\overline{\nu_i}}$ and $\psi_i(x)$ can be greater than $\varphi_k(x)$ on this sub domain. This is done because over every subinterval $[x_{i-1}, x_i]$ we are interested only at the subregion $\overline{Q}_{\overline{\nu_i}}$ corresponding to the maximal index $\overline{\nu_i}$.

It has been also shown in [SFP01] that it is possible to find explicitly the value ψ_i^* defined as follows

$$\psi_i^* = \begin{cases} 0.5(z_{i-1} + z_i - L_{\nu_i}(x_i - x_{i-1})), & \nu_{i-1} = \nu_i, \\ z_i - L_{\nu_i}(x_i - x_{i-1} - z_{i-1}/L_{\nu_{i-1}}), & \nu_{i-1} < \nu_i, \\ z_{i-1} - L_{\nu_{i-1}}(x_i - x_{i-1} - z_i/L_{\nu_i}), & \nu_{i-1} > \nu_i, \end{cases} \tag{14}$$

$$\psi_i^* \le \min\{\psi_i(x) : x \in \overline{Q}_{\overline{v_i}}\}.$$

The algorithm IBBA from [SFP01] sequentially updates the function $\psi_i(x)$ during its work in such a way that the estimate z_k^* of the global minimum f^* of the original problem (1) is sequentially improved. At each iteration k of the algorithm the value f^* can be bounded as follows

$$f^* \in [\Psi^k + z_k^*, z_k^*],$$

where

$$\Psi^k = \min\{\psi_i^* : 1 \le i \le k\}.$$

However, the IBBA obtains these nice results by working with a priori known Lipschitz constants from (4), (5). In the next Section, we describe a new algorithm that adaptively estimates them during its work.

3 A New Geometric Index Algorithm Adaptively Estimating Lipschitz Constants

In the new algorithm GEA, we propose to substitute the unknown values, $L_j, 1 \le j \le m+1$, of the Lipschitz constants by their adaptively recalculated during each iteration k estimates $\Lambda_j = \Lambda_j(k), 1 \le j \le m+1$, using the information obtained from executing trials at the points x_i, $0 \le i \le k$, from (10). We calculate the estimate

$$\Lambda_j = \max\{\xi, \max\{\lambda_i : \nu_i = j,\ 0 \le i \le k\}\}, \qquad 1 \le j \le m+1, \tag{15}$$

where a value λ_i estimates the Lipschitz constant L_{ν_i} at a point x_i having the index ν_i and the parameter $\xi > 0$ is a small number reflecting our supposition that the objective function and constraints are not just constants over $[a,b]$, i.e., $L_j \ge \xi$, $1 \le j \le m+1$. The values λ_i are calculated as follows

$$\lambda_i = \begin{cases} \max\{\mid z_j - z_{j-1} \mid (x_j - x_{j-1})^{-1} : \ j = i, i+1\}, & \text{if } \nu_{i-1} = \nu_i = \nu_{i+1} \\ \max\{\mid z_i - z_{i-1} \mid (x_i - x_{i-1})^{-1},\ z_i(x_{i+1} - x_i)^{-1}\}, & \text{if } \nu_{i-1} = \nu_i < \nu_{i+1} \\ \max\{\mid z_{i+1} - z_i \mid (x_{i+1} - x_i)^{-1},\ z_i(x_i - x_{i-1})^{-1}\}, & \text{if } \nu_{i-1} > \nu_i = \nu_{i+1} \\ \max\{z_i(x_i - x_{i-1})^{-1},\ z_i(x_{i+1} - x_i)^{-1}\}, & \text{if } \nu_i < \nu_{i-1},\ \nu_i < \nu_{i+1} \\ z_i(x_i - x_{i-1})^{-1}, & \text{if } \nu_{i-1} > \nu_i > \nu_{i+1} \\ z_i(x_{i+1} - x_i)^{-1}, & \text{if } \nu_{i-1} < \nu_i < \nu_{i+1} \\ \mid z_i - z_{i-1} \mid (x_i - x_{i-1})^{-1}, & \text{if } \nu_{i-1} = \nu_i > \nu_{i+1} \\ \mid z_{i+1} - z_i \mid (x_{i+1} - x_i)^{-1}, & \text{if } \nu_{i-1} < \nu_i = \nu_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Naturally, when $i = 0$ or $i = k$, only one of the two expressions in the first four cases are defined and are used to calculate λ_i.

Thus, in (15), in order to estimate the value L_j we find all the points x_i having the index $\nu_i = j$ and take the maximum of the values λ_i corresponding to these points if it is greater than ξ, and ξ in the opposite case. The following theorem holds.

Theorem 1. *If the Lipschitz constants $L_j \geq \xi, 1 \leq j \leq m+1$, then the values Λ_j from (15) are underestimates of the Lipschitz constants $L_j, 1 \leq j \leq m+1$, and the following inequalities hold*

$$\xi \leq \Lambda_j \leq L_j, \qquad 1 \leq j \leq m+1.$$

Proof. This fact follows from (15), (16), and Proposition 1. □

We are ready now to describe the new algorithm GEA. Suppose that $k + 1, k \geq 1$, trails have been already executed in a way at points

$$x^0 = a, x^1 = b, x^2, x^3, ..., x^i, ...x^{k-1}, x^k \tag{17}$$

and their indexes and the value

$$M^k = \max\{\nu(x^i) : 0 \leq i \leq k\} \tag{18}$$

have been calculated. The value M^k defined in (18) estimates the maximal index M from (3) during the search. The choice of the point x^{k+1}, $k \geq 1$, where the next trial will be executed is determined by the rules presented below.

Step 1. The points $x^0, ..., x^k$ of the previous k iterations are renumbered by subscripts in order to form the sequence (10). Thus, two numerations are used during the work of the algorithm. The record x^i from (17) means that this point has been generated during the i-th iteration of the GEA. The record x_i indicates the place of the point in the row (10). Of course, the second enumeration is changed during every iteration.

Step 2. Recalculate the estimate z_k^* from (12) and associate with the points x_i from (10) the values $z_i = \varphi_k(x_i), 0 \leq i \leq k$, where the values $\varphi_k(x_i)$ are from (11).

Calculate the current estimates $\Lambda_j(k)$, $1 \leq j \leq m+1$, of the Lipschitz constants L_j, $1 \leq j \leq m+1$, as follows:

$$\Lambda_j = \Lambda_j(k) := \max\{\Lambda_j(k-1),\ \Lambda_j(k)\}, \quad 1 \leq j \leq m+1,$$

where $\Lambda_j(k)$, $1 \leq j \leq m+1$, at the right part of this formula are given by (15) and $\Lambda_j(k-1)$, $1 \leq j \leq m+1$, are the estimates of the Lipschitz constants L_j, $1 \leq j \leq m+1$, calculated at the previous iteration $k-1$.

Step 3. For each interval $[x_{i-1}, x_i]$, $1 \le i \le k$, calculate the *characteristic* of the interval

$$R_i = \begin{cases} 0.5\,[z_{i-1} + z_i - r\Lambda_{\nu_i}(x_i - x_{i-1})]\,, & \nu_{i-1} = \nu_i, \\ z_i - r\Lambda_{\nu_i}(x_i - x_{i-1} - z_{i-1}/r\Lambda_{\nu_{i-1}}), & \nu_{i-1} < \nu_i, \\ z_{i-1} - r\Lambda_{\nu_{i-1}}(x_i - x_{i-1} - z_i/r\Lambda_{\nu_i}), & \nu_{i-1} > \nu_i, \end{cases} \qquad (19)$$

where $r > 1$ is the reliability parameter of the method.

Step 4. Find an interval t corresponding to the minimal characteristic, i.e.,

$$t = \min\{\arg\min\{R_i : 1 \le i \le k\}\}. \qquad (20)$$

Step 5. If for the interval $[x_{t-1}, x_t]$, where t is from (20), the stoping rule

$$x_t - x_{t-1} \le \epsilon \qquad (21)$$

is satisfied for a preset accuracy $\epsilon > 0$, then **Stop** – the required accuracy has been reached. In the opposite case go to Step 6.

Step 6. Execute the $(k+1)$-th trial at the point

$$x^{k+1} = \begin{cases} 0.5[x_{t-1} + x_t + (z_{t-1} - z_t)/(r\Lambda_{\nu_t})], & \text{if } \nu_{t-1} = \nu_t \\ 0.5(x_{t-1} + x_t), & \text{if } \nu_{t-1} \ne \nu_t, \end{cases} \qquad (22)$$

and evaluate its index $\nu(x^{k+1})$.

Step 7. If $\nu(x^{k+1}) > M^k$, then perform two additional trials at the points

$$x^{k+2} = 0.5(x_{t-1} + x^{k+1}), \qquad (23)$$

$$x^{k+3} = 0.5(x^{k+1} + x_t), \qquad (24)$$

calculate their indexes, set $k = k + 3$, and go to Step 8. If $\nu(x^{k+1}) < M^k$ and there is only one point x_T with the maximal index M^k, i.e., $\nu_T = M^k$, then execute two additional trials at the points

$$x^{k+2} = 0.5(x_{T-1} + x_T), \qquad (25)$$

$$x^{k+3} = 0.5(x_T + x_{T+1}), \qquad (26)$$

if $0 < T < k$, calculate their indexes, set $k = k + 3$, and go to Step 8. If $T = 0$ then the trial is executed only at the point (26). Analogously, if $T = k$ then the trial is executed only at the point (25). In these two cases calculate the index of the additional point, set $k = k + 2$ and go to Step 8. In all the remaining cases set $k = k + 1$ and go to Step 8.

Step 8. Calculate M^k and go to Step 1.

The introduced algorithm constructs an auxiliary function similar to the discontinuous minorant $\psi_i(x)$. The difference consists of the substitution of the unknown Lipscitz constants L_j by their adaptively recalculated during each iteration k estimates $\Lambda_j = \Lambda_j(k), 1 \le j \le m+1$, multiplied by the reliability parameter $r > 1$. Thus, the characteristics R_i estimate the values ψ_i^* from (14). Before proving a theorem establishing sufficient global convergence conditions of the new method we need the following lemma.

Lemma 1. *Let $\bar{x}$ be a limit point of the sequence $\{x^k\}$ generated by the algorithm GEA proposed above and $i = i(k)$ be the number of an interval $[x_{i-1}, x_i]$ containing this point in the course of the k-th iteration. Then,*

$$\lim_{k\to\infty} x_{i(k)} - x_{i(k)-1} = 0. \tag{27}$$

Proof. Since the point $\bar{x}$ is a limit point of the trial sequence $\{x^k\}$ generated by the GEA, there exists an infinite number of iterations k such that the interval $[x_{i(k)-1}, x_{i(k)}]$ containing this point will be subdivided in the course of this iteration into two subintervals

$$[x_{i-1}, y], \quad [y, x_i].$$

The point $y = x^{k+n}$ is determined by formulae (22)–(26) and n can vary from 1 to 3. Thus, in order to prove (27) it is sufficient to show that the following contracting estimate

$$\max\{x_i - y, y - x_{i-1}\} \le \alpha(x_i - x_{i-1}), \qquad 0.5 \le \alpha < 1, \tag{28}$$

holds for all the cases (22)–(26).

In the cases (23)–(26) and (22) with $\nu_t \ne \nu_{t-1}$ we obtain (28) immediately by taking $\alpha = 0.5$. In the case (22) with $\nu_t = \nu_{t-1}$ we can write that

$$x_i - x^{k+1} = x_i - 0.5[x_{i-1} + x_i + (z_{i-1} - z_i)/(r\Lambda_i)] \le$$

$$0.5[(x_i - x_{i-1}) + \mid z_{i-1} - z_i \mid /(r\Lambda_i)] \le 0.5(1 + r^{-1})(x_i - x_{i-1})$$

taking into account that we have $\nu_i = \nu_{i-1}$ and due to (15), (16) it follows

$$\mid z_i - z_{i-1} \mid \le \Lambda_i(x_i - x_{i-1}).$$

An analogous estimate can be obtained for the difference $x^{k+1} - x_{i-1}$. Thus, (28) is proved for the case (22) with $\nu_t = \nu_{t-1}$ also because we can take $\alpha = (r+1)/(2r)$. It is clearly satisfies the condition $0.5 \le \alpha < 1$ since $r > 1$. This last observation concludes the proof. □

Let us now formulate and prove a theorem describing sufficient conditions of global convergence of the algorithm GEA.

Theorem 2. *Let the feasible region $Q_{m+1} \ne \varnothing$ consist of intervals having finite lengths, x^* be any solution to problem (1), and $j = j(k)$ be the number of an interval $[x_{j-1}, x_j]$ containing this point during the k-th iteration. Then, if for $k \ge k^*$ the following conditions*

$$r\Lambda_{\nu_{j-1}} > C_{j-1}, \qquad r\Lambda_{\nu_j} > C_j, \tag{29}$$

$$C_{j-1} = z_{j-1}/(x^* - x_{j-1}), \qquad C_j = z_j/(x_j - x^*). \tag{30}$$

take place, then the point x^* will be a limit point of the trial sequence $\{x^k\}$ generated by the GEA.

Proof. Let us return to the interval $[x_{i-1}, x_i]$ from Lemma 1 containing a limit point $\bar{x}$ of the sequence $\{x^k\}$. Since it contains the limit point, it follows from (19), Proposition 1, and Lemma 1 that

$$\lim_{k\to\infty} R_{i(k)} = \lim_{k\to\infty} \varphi_k(\bar{x}) \geq 0 \tag{31}$$

because for all k the values $z_i \geq 0, 1 \leq i \leq k$.

Let us estimate now the characteristic $R(j(k))$ of the interval $[x_{j-1}, x_j]$ the global minimizer x^* belongs to during the k-th iteration. It follows from (30) that

$$z_{j-1} = C_{j-1}(x^* - x_{j-1}), \qquad z_j = C_j(x_j - x^*). \tag{32}$$

Consider first the case $\nu_{j-1} = \nu_j$. From (32) we obtain

$$z_j + z_{j-1} \leq \max\{C_{j-1}, C_j\}(x_j - x_{j-1}).$$

From this estimate, (19), and (29) we can derive that

$$R_j = 0.5[z_{j-1} + z_j - r\Lambda_{\nu_j}(x_j - x_{j-1})] \leq$$
$$0.5[\max\{C_{j-1}, C_j\}(x_j - x_{j-1}) - r\Lambda_{\nu_j}(x_j - x_{j-1})] < 0. \tag{33}$$

In the case $\nu_{j-1} < \nu_j$ we obtain from (19), (29), and (32) that

$$R_j = z_j - r\Lambda_{\nu_j}(x_j - x_{j-1} - z_{j-1}/r\Lambda_{\nu_{j-1}}) =$$
$$C_j(x_j - x^*) - r\Lambda_{\nu_j}(x_j - x_{j-1}) + \Lambda_{\nu_j} z_{j-1}/\Lambda_{\nu_{j-1}} =$$
$$C_j(x_j - x^*) - r\Lambda_{\nu_j}(x_j - x_{j-1}) + C_{j-1}(x^* - x_{j-1})\Lambda_{\nu_j}/\Lambda_{\nu_{j-1}} <$$
$$C_j(x_j - x^*) - r\Lambda_{\nu_j}(x_j - x_{j-1}) + r\Lambda_{\nu_j}(x^* - x_{j-1}) =$$
$$(C_j - r\Lambda_{\nu_j})(x_j - x^*) < 0. \tag{34}$$

By a complete analogy we can obtain the estimate $R_j < 0$ in the case $\nu_{j-1} > \nu_j$.

Assume now, that x^* is not a limit point of the sequence $\{x^k\}$. Then, there exists a number Q such that for all $k \geq Q$ the interval $[x_{j-1}, x_j], j = j(k)$, is not changed, i.e., new trial points will not fall into this interval.

Consider again the interval $[x_{i-1}, x_i]$ from Lemma 1 containing the limit point $\bar{x}$. It follows from (31), (33), and (34) that there exists a number N such that

$$R(i(k)) > R(j(k))$$

for all $k \geq k^* = \max\{Q, N\}$. This means that starting from k^* the characteristic of the interval $[x_{i-1}, x_i], i = i(k), k \geq k^*$, is not minimal and due to Step 4 of the GEA, the interval $[x_{i-1}, x_i]$ will not be chosen for the further subdivision until a trial will not fall into the interval $[x_{j-1}, x_j]$. Hence, a trial will fall into the interval $[x_{j-1}, x_j]$ because $\bar{x}$ is a limit point and infinitely many points should fall in it. Thus, our assumption that x^* is not a limit point was false and we have proved the theorem. □

4 Numerical Comparison

The new algorithm has been numerically compared with the following methods:

- The method proposed by Pijavskii (see [HJ95, Pij72]) combined with the penalty approach used to reduce the constrained problem to an unconstrained one; this method is indicated hereafter as PEN. The Lipschitz constant of the obtained unconstrained problem is supposed to be known as it is required by Pijavskii algorithm.
- A method (indicated hereafter as GPEN) where a piece-wise linear auxiliary function is constructed similarly to Pijavskii algorithm and the penalty approach is used to reduce the constrained problem to an unconstrained one. The Lipschitz constant of the obtained unconstrained problem is assumed to be unknown and is estimated adaptively during the search by the value

$$K_k = r \cdot \max\{\xi, \max\{\mid z_i - z_{i-1} \mid (x_i - x_{i-1})^{-1} : 1 \le i \le k\}\},$$

  where $x_i, 0 \le i \le k$, are trial points, $r > 1$ is the reliability parameter of the method, and ξ is a small positive constant (having the same sense as in (15)).
- The method IBBA from [SFP01] using the index scheme in combination with the Branch-and-Bound approach and the known Lipschitz constants $L_j, 1 \le j \le m+1$, from (4), (5).

Ten non-differentiable test problems introduced in [FSP02] have been used in all the experiments (since there were several misprints in the original paper [FSP02], the accurately verified formulae have been applied, which are available at http://wwwinfo.deis.unical.it/~yaro/constraints.html). In this set of tests, problems 1–3 have one constraint, problems 4–7 two constraints, and problems 8–10 three constrains. The same accuracy $\epsilon = 10^{-4}(b-a)$ with b and a from (1) and the same value $\xi = 10^{-6}$ from (15) have been used in all the experiments for all the methods.

In Table 1 and in Table 2, results for the PEN, taken from [FSP02], and for the GPEN with the parameter $r = 1.2$ are presented, respectively. The constrained problems were reduced to the unconstrained ones as follows

$$f_{P^*}(x) = f(x) + P^* \max\{0, g_1(x), g_2(x), \dots, g_{N_v}(x)\}. \qquad (35)$$

The coefficient P^* has been computed by the following rules:

1. P^* has been chosen equal to 15 for all the problems and it has been checked if the found solution (XPEN, FPEN) for each problem belongs or not to the feasible subregions;
2. if it does not belong to the feasible subregions, the coefficient P^* has been iteratively increased by 10 starting from 20 until a feasible solution has been found. Particularly, this means that a feasible solution has not been found in Table 1 for the problem 5 when P^* was equal to 15.

Table 1. Numerical results obtained by the PEN

Problem	XPEN	FPEN	P^*	Trials	Evaluations
1	1.258104	4.174415	15	247	494
2	1.959536	−0.079023	15	241	482
3	9.400529	−4.400529	15	917	1834
4	0.332786	3.346203	15	273	819
5	0.869955	0.741685	20	671	2013
6	3.769843	0.166667	15	909	2727
7	5.201133	0.903518	15	199	597
8	8.028603	4.051839	15	365	1460
9	0.950192	2.648041	15	1183	4732
10	0.799887	1.000726	15	135	540
Average	–	–	–	514.0	1569.8

Table 2. Numerical result obtained by the GPEN with $r = 1.2$

Problem	XGPEN	FGPEN	P^*	Trials	Evaluations
1	1.258027	4.174495	15	307	614
2	1.959400	−0.078897	15	252	504
3	9.400288	−4.400288	15	1131	2262
4	0.333012	3.346197	15	68	204
5	0.870147	0.741996	20	830	2490
6	3.769644	0.166667	15	1142	3426
7	5.201371	0.904226	15	155	465
8	8.028173	4.053683	15	387	1548
9	0.950099	2.648041	15	810	3240
10	0.800071	1.000427	15	146	584
Average	–	–	–	522.8	1533.7

It should be noticed that in Tables 1 and 2 the column "Evaluations" shows the total number of evaluations of the objective function $f(x)$ and all the constraints. Thus, it is equal to

$$(N_v + 1) \times N_{trials},$$

where N_v is the number of constraints and N_{trials} is the number of the trials executed by the PEN and by the GPEN for each problem, respectively.

Results obtained by the IBBA, taken from [SFP01], and by the new method GEA with the parameter $r = 1.2$ are summarized in Table 3 and in Table 4, respectively. Columns in Tables 3 and 4 have the following meaning:

–the columns XIBBA, FIBBA and XGEA, FGEA represent the estimate to the global solution $(x^*, f(x^*))$ found by the IBBA and by the GEA for each problem, respectively;

Table 3. Numerical results obtained by the IBBA

Problem	XIBBA	FIBBA	N_{g_1}	N_{g_2}	N_{g_3}	N_f	Trials	Eval.
1	1.258310	4.174200	23	–	–	28	51	79
2	1.959676	−0.079152	18	–	–	16	34	50
3	9.400685	−4.400685	171	–	–	19	190	209
4	0.332868	3.346198	136	15	–	84	235	418
5	0.869951	0.741679	168	91	–	24	283	422
6	3.769772	0.166667	16	16	–	597	629	1839
7	5.201208	0.903122	63	18	–	39	120	216
8	8.028350	4.050069	29	11	3	21	64	144
9	0.950325	2.648041	8	86	57	183	334	1083
10	0.799964	1.000233	42	3	17	13	75	151
Average	–	–	–	–	–	–	201.5	461.1

Table 4. Numerical result obtained by the GEA with $r = 1.2$

Problem	XGEA	FGEA	N_{g_1}	N_{g_2}	N_{g_3}	N_f	Trials	Eval.
1	1.257961	4.174564	23	–	–	17	40	57
2	1.959660	−0.079138	19	–	–	19	38	57
3	9.400450	−4.400450	71	–	–	12	83	95
4	0.332824	3.346200	137	16	–	91	244	442
5	0.870149	0.742000	147	68	–	18	233	337
6	3.769387	0.166667	14	13	–	168	195	544
7*	5.201611	0.905851	46	27	–	45	118	235
8	8.028361	4.049852	9	15	4	23	51	143
9	0.950117	2.648041	11	35	1	113	160	536
10	0.800044	1.000266	12	1	15	16	44	123
Average	–	–	–	–	–	–	120.6	256.9

–the columns N_{g_1}, N_{g_2}, and N_{g_3} represent the number of trials where the constraint g_i, $1 \leq i \leq 3$, was the last evaluated constraint;

–the column "Eval." is the total number of evaluations of the objective function and the constraints. This quantity is equal to:

–$N_{g_1} + 2 \times N_f$, for problems with one constraint;

–$N_{g_1} + 2 \times N_{g_2} + 3 \times N_f$, for problems with two constraints;

–$N_{g_1} + 2 \times N_{g_2} + 3 \times N_{g_3} + 4 \times N_f$, for problems with three constraints.

The symbol '*' in Table 4 indicates that $r = 1.2$ was not sufficient to find the global minimizer of problem 7. The results for this problem in Table 4 are related to the value $r = 1.6$ which was sufficient to locate the solution.

Finally, Table 5 represents the speed up obtained by the GEA in comparison with the other methods used in the experiments.

In order to illustrate performance of the methods graphically, in Figs. 1–4 we show the dynamic diagrams of the search executed by the PEN, the GPEN, the IBBA, and the GEA, correspondingly, for problem 5 from [FSP02] that

Table 5. Speed up obtained by the GEA with $r = 1.2$ in comparison with the other methods used in the experiments

Problem	Trials			Evaluations		
	$\frac{\text{PEN}}{\text{GEA}}$	$\frac{\text{GPEN}}{\text{GEA}}$	$\frac{\text{IBBA}}{\text{GEA}}$	$\frac{\text{PEN}}{\text{GEA}}$	$\frac{\text{GPEN}}{\text{GEA}}$	$\frac{\text{IBBA}}{\text{GEA}}$
1	6.18	7.68	1.28	8.67	10.77	1.39
2	6.34	6.63	0.89	8.46	8.84	0.88
3	11.05	13.63	2.29	19.31	23.81	2.20
4	1.12	0.28	0.96	1.85	0.46	0.95
5	2.88	3.56	1.21	5.97	7.39	1.25
6	4.66	5.86	3.23	5.01	6.36	3.38
7*	1.69	1.31	1.02	2.54	1.98	0.92
8	7.16	7.59	1.25	10.21	10.83	1.01
9	7.40	5.06	2.09	8.83	6.04	2.02
10	3.07	3.32	1.70	4.39	4.75	1.23
Average	5.16	5.49	1.59	7.52	8.12	1.52

can be seen in the upper subplots of Figs. 3, 4:

$$\begin{aligned} &\min_{x\in[0,4]} \quad f(x) = \tfrac{1}{4}\left(\left|x-\tfrac{3}{2}\right| - |\sin(10x)| + 3\right) \\ &\text{subject to} \\ &\qquad g_1(x) = \exp\left(-\left|\sin\left(\tfrac{5}{2}\sin\left(\tfrac{11}{5}x\right)\right)\right|\right) - \tfrac{1}{2} + \tfrac{1}{100}x^2 \le 0 \\ &\qquad g_2(x) = \tfrac{8}{25} - \exp(-x)\,|\sin(3\pi x)| \le 0 \end{aligned} \tag{36}$$

The problem has two constraints leading to three disjoint feasible subregions shown by three continuous bold lines in Figs. 3, 4. The global minimizer is located at the point $x^* = 0.86992$.

Figures 1, 2 present at their upper subplots the unconstrained problem (35) obtained from (36). The line of symbols '+' located under the graph of the function (35) in these Figs. shows points at which trials have been executed by the corresponding methods. Finally, the lower subplots show dynamics of the search. The PEN has executed 671 trials and the number of evaluations was equal to $671 \times 3 = 2013$. The number of trials executed by the GPEN is equal to 830 and the number of evaluations is equal to $830 \times 3 = 2490$.

In Fig. 3, showing performance of the IBBA, the first line (from up to down) of symbols '+', located under the graph of problem (36), represents the points where the first constraint has not been satisfied (number of such trials is equal to 168). Thus, due to the decision rule of the IBBA, the second constraint has not been evaluated at these points.

The second line of symbols '+' represents the points where the first constraint has been satisfied but the second constraint has been not (number of such trials is equal to 91). At these points both constraints have been evaluated but the objective function has been not. The last line represents the

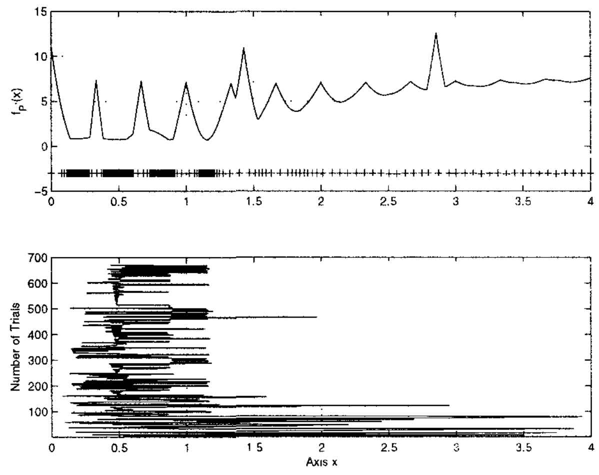


Fig. 1. Solving by the PEN the unconstrained problem (35) obtained from the constrained problem (36)

points where both the constraints have been satisfied (number of such trials is 24) and, therefore, the objective function has been evaluated too. The total number of evaluations is equal to $168+91\times2+24\times3 = 422$. These evaluations have been executed during $168+91+24 = 283$ trials.

Analogously, in Fig. 4, demostrating performance of the GEA, the first line of symbols '+' indicates 147 trial points where the first constraint has not been satisfied. The second line represents 68 points where the first constraint has been satisfied but the second constraint has been not. The last line shows 18 points where both the constraints have been satisfied and the objective function has been evaluated. The total number of evaluations is equal to $147 + 68 \times 2 + 18 \times 3 = 337$. These evaluations have been executed during $147 + 68 + 18 = 233$ trials.

References

[Ber96] Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific, Belmont, MA (1996)

[Ber99] Bertsekas, D.P.: Nonlinear Programming. 2nd ed. Athena Scientific, Belmont, MA (1999)

[Bro58] Brooks, S.H.: Discussion of random methods for locating surface maxima. Operation Research, **6**, 244–251 (1958)

[BS02] Barkalov, K.A., Strongin, R.G.: A global optimization technique with an adaptive order of checking for constraints. Comput. Math. Math. Phys., **42**, 1289–1300 (2002)

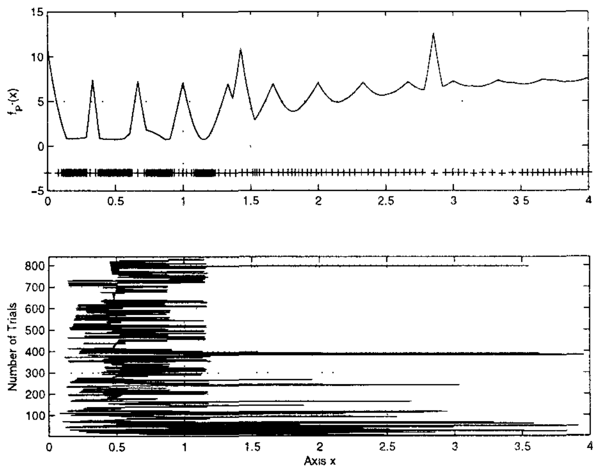


Fig. 2. Solving by the GPEN with $r = 1.2$ the unconstrained problem (35) obtained from the constrained problem (36)

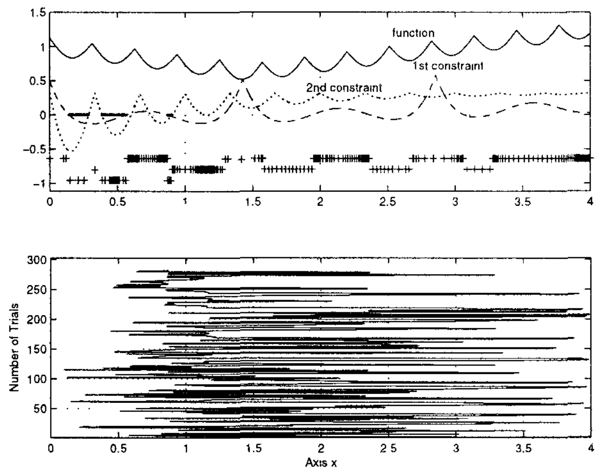


Fig. 3. Solving by the IBBA the constrained problem (36)

[CZ99] Calvin, J., Žilinskas, A.: On the convergence of the P-algorithm for one-dimensional global optimization of smooth functions. J. Optim. Theory Appl., **102**, 479–495 (1999)

[DGMS95] Daponte, P., Grimaldi, D., Molinaro, A., Sergeyev, Ya.D.: An algorithm for finding the zero-crossing of time signals with Lipschitzean derivatives. Measurement, **16**, 37–49 (1995)

[DGMS96] Daponte, P., Grimaldi, D., Molinaro, A., Sergeyev, Ya.D.: Fast detection of the first zero-crossing in a measurement signal set. Measurement, **19**,

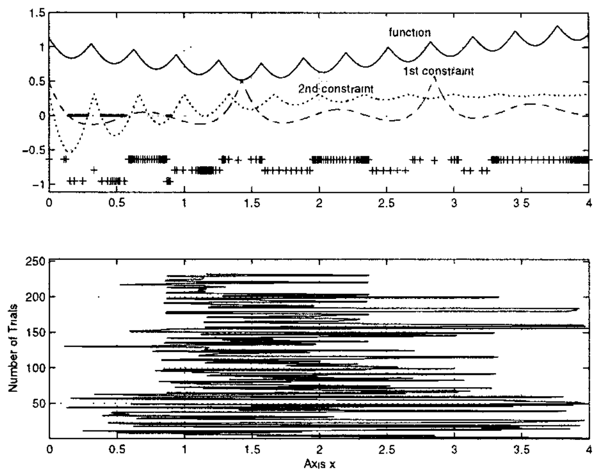


Fig. 4. Solving by the GEA with $r = 1.2$ the constrained problem (36)

29–39 (1996)

[FP96] Floudas, C.A., Pardalos, P.M.: State of the Art in Global Optimization. Kluwer Academic Publishers, Dordrecht (1996)

[FSP02] Famularo, D., Sergeyev, Ya.D., Pugliese, P.: Test problems for Lipschitz univariate global optimization with multiextremal constraints. In: Dzemyda, G., Šaltenis, V., Žilinskas, A. (eds) Stochastic and Global Optimization. Kluwer Academic Publishers, Dordrecht, 93–110 (2002)

[HJ95] Hansen, P., Jaumard, B.: Lipshitz optimization. In: Horst, R., Pardalos, P.M. (eds) Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht, 407–493 (1995)

[HP95] Horst, R., Pardalos, P.M.: Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht (1995)

[HT96] Horst, R., Tuy, H.: Global Optimization – Deterministic Approaches. 3rd ed. Springer-Verlag, Berlin (1996)

[Lam99] Lamar, B.W.: A method for converting a class of univariate functions into d.c. functions. J. Global Optim., **15**, 55–71 (1999)

[LS95] Locatelli, M., Schoen, F.: An adaptive stochastic global optimisation algorithm for one-dimensional functions. Ann. Oper. Res., **58**, 263–278 (1995)

[Mla92] Mladineo, R.: Convergence rates of a global optimization algorithm. Math. Program., **54**, 223–232 (1992)

[MSB96] MacLagan, D., Sturge, T., Baritompa, W.P.: Equivalent methods for global optimization. In: Floudas, C.A., Pardalos, P.M. (eds) State of the Art in Global Optimization. Kluwer Academic Publishers, Dordrecht, 201–212 (1996)

[NW99] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research, Springer Verlag (1999)

[Pij72] Pijavskii, S.A.: An algorithm for finding the absolute extremum of a function. USSR Comput. Math. Math. Phys., **12** 57–67 (1972)

[Pin96] Pintér, J.D.: Global Optimization in Action. Kluwer Academic Publisher, Dordrecth (1996)

[PKS87] Patwardhan, A.A., Karim, M.N., Shah, R.: Controller tuning by a least-squares method. AIChE J., **33**, 1735–1737 (1987)

[PR90] Pardalos, P.M., Rosen, J.B. (eds): Computational Methods in Global Optimization. Annals of Operations Research, **25** (1990)

[RWPD85] Ralston, P.A.S., Watson, K.R., Patwardhan, A.A., Deshpande, P.B.: A computer algorithm for optimized control. Industrial and Engineering Chemistry, Product Research and Development, **24**, 1132 (1985)

[SDGM99] Sergeyev, Ya.D., Daponte, P., Grimaldi, D., Molinaro, A.: Two methods for solving optimization problems arising in electronic measurements and electrical engineering. SIAM J. Optim., **10**, 1–21 (1999)

[Ser98] Sergeyev, Ya.D.: Global one-dimensional optimization using smooth auxiliary functions. Math. Program., **81**, 127–146 (1998)

[Ser99] Sergeyev, Ya.D.: On convergence of "Divide the Best" global optimization algorithms. Optimization, **44**, 303–325 (1999)

[SFP01] Sergeyev, Ya.D., Famularo, D., Pugliese, P.: Index branch-and-bound algorithm for Lipschitz univariate global optimization with multiextremal constraints. J. Global Optim., **21**, 317–341 (2001)

[SG01] Sergeyev, Ya.D., Grishagin, V.A.: Parallel asynchronous global search and the nested optimization scheme. J. Comput. Anal. Appl., **3(2)**, 123–145 (2001)

[SM86] Strongin, R.G., Markin, D.L.: Minimization of multiextremal functions with nonconvex constraints. Cybernetics, **22**, 486–493 (1986)

[SM95] Sergeyev, Ya.D., Markin, D.L.: An algorithm for solving global optimization problems with nonlinear constraints. J. Global Optim., **7**, 407–419 (1995)

[SS00] Strongin, R.G., Sergeyev, Ya.D.: Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers, Dordrecht (2000)

[Str78] Strongin, R.G.: Numerical Methods on Multiextremal Problems. Nauka, Moscow (1978) in Russian

[Str84] Strongin, R.G.: Numerical methods for multiextremal nonlinear programming problems with nonconvex constraints. In: Demyanov, V.F., Pallaschke, D. (eds) Lecture Notes in Economics and Mathematical Systems 255, Proceedings 1984. Springer-Verlag. IIASA, Laxenburg/Austria, 278–282 (1984)

[TZ89] Törn, A., Žilinskas, A.: Global Optimization. Springer-Verlag, Lecture Notes in Computer Science, **350** (1989)

[WC96] Wang, X., Chang, T.S.: An improved univariate global optimization algorithm with improved linear bounding functions. J. Global Optim., **8**, 393–411 (1996)

[Zhi91] Zhigljavsky, A.A.: Theory of Global Random Search. Kluwer Academic Publishers, Dordrecht (1991)

Packing up to 200 Equal Circles in a Square

Péter Gábor Szabó[1] and Eckard Specht[2]

[1] Department of Applied Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary pszabo@inf.u-szeged.hu
[2] Department of Experimental Physics, University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany specht@hydra.nat.uni-magdeburg.de

1 Introduction

The Hungarian mathematician Farkas Bolyai (1775–1856) published in his principal work ('Tentamen', 1832–33 [Bol04]) a dense regular packing of equal circles in an equilateral triangle (see Fig. 1). He defined an infinite packing series and investigated the limit of *vacuitas* (in Latin, the gap in the triangle outside the circles). It is interesting that these packings are not always optimal in spite of the fact that they are based on hexagonal grid packings. Bolyai probably was the first author in the mathematical literature who studied the density of a series of packing circles in a bounded shape.

The problem of finding the densest packing of n equal and non-overlapping circles has been studied for several shapes of the bounding region (e.g. in a rectangle, triangle and circle [Mel97]). This chapter focuses only on the 'Packing of Equal Circles in a Square'-problem (PECS problem), however, the developed stochastic optimization algorithm can be used for other shapes as well.

The Hungarian mathematicians Dezső Lázár and László Fejes Tóth have already investigated the PECS problem before 1940. This problem first appeared in the literature in 1960, when Leo Moser [Mos60] guessed the optimal arrangement of 8 circles. J. Schaer and A. Meir [SM65] proved this conjecture and J. Schaer [Sch65] solved the $n = 9$ case, too. C. de Groot et al. [GPW90] solved the $n = 10$ case after many authors published new and improved packings. R. Peikert et al. [PWMG92] found and proved optimal packings from $n = 10$ to $n = 20$ using a computer aided method. Based on theoretical tools only, G. Wengerodt published proofs for $n = 14, 16$, and 25 [Wen83, Wen87a, Wen87b], and with K. Kirchner for $n = 36$ [KW87]. However, there are gaps in both of proofs for $n = 25$ and 36 according to the review MR1453444 in Mathematical Reviews. In the last decades, several deterministic [NO99, LR02, Mar03, Mar04, MC05] and stochastic [NO97, BDGL00, CGSC01] methods were published for this problem.

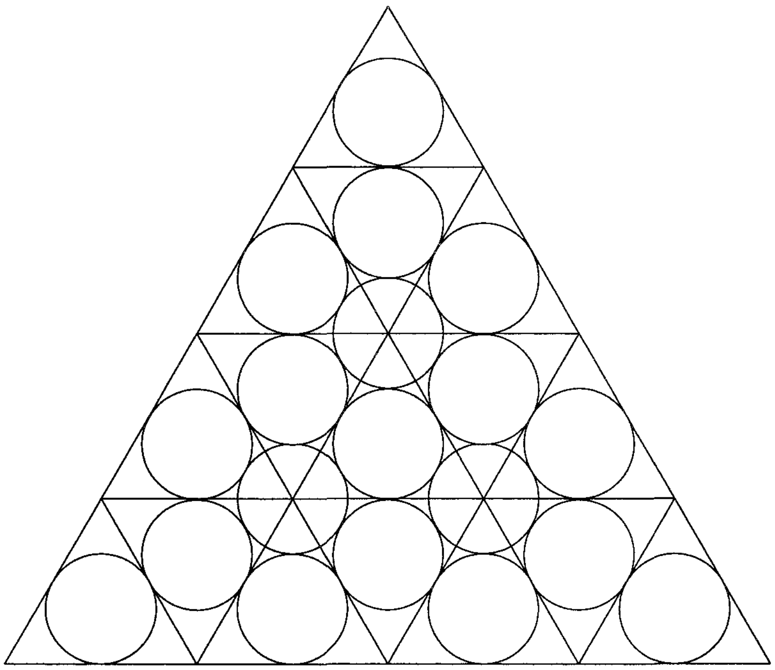


Fig. 1. The example of Bolyai for packing 19 equal circles in an equilateral triangle

Proven optimal packings are known up to $n = 30$ [PWMG92, NO99, Mar03, Mar04, MC05]. Approximate packings (i.e. packings determined by computer aided numerical computations without a rigorous proof) were reported in the literature for up to $n = 100$ [NO97, BDGL00, CGSC01, LR02]. At the same time, some other related results (e.g. patterns, bounds and some properties of the optimal solutions) were published as well [GL96, NOS99, Sza00, SCCG01]. A more detailed history of the PECS problem can be found in [PWMG92, Mel97, SCCG01, SMC05].

In this chapter we propose many new approximate packings. These packings are interesting for discrete geometric investigations, because they suggest new possible structures. Note that similar structures occur in several cases, so they define pattern classes [GL96, NO97, NOS99, Sza00]. The good approximate packings are important for the reliable computer aided methods to speed up the localization of optimal packings and to prove the optimality [Mar03, NO99, PWMG92].

The chapter is organized as follows: Sect. 2 presents definitions and mathematical models of the PECS problem as a global optimization problem. In Sect. 3, a theoretical lower bound is given for every $n \geq 2$. We summarize some earlier deterministic and stochastic optimization approaches in Sect. 4. In the following Sect. 5, we propose a modified billiard simulation method and in Sect. 6 numerical values, figures and density plots of all optimal and approximate packings up to $n = 200$ are given. It has been verified by interval arithmetic based computations that the numerical results represent in fact existing packings.

2 Definitions and Models

First of all, we give a formal definition of the PECS problem.

Definition 2.1 Let us denote the number of the circles by $n \geq 2$. A *packing of circles with radius r_n in the unit square* is $P = (p_1, \ldots, p_n) \in P_{r_n}$, where $P_{r_n} = \{((x_1, y_1), \ldots, (x_n, y_n)) \in [0,1]^{2n} \mid (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r_n^2; x_i, y_i \in [r_n, 1 - r_n] \ (1 \leq i < j \leq n)\}$. P is an *optimal packing of circles*, if $P \in P_{\overline{r}_n}$, where $\overline{r}_n \max\limits_{P_{r_n} \neq \emptyset} r_n$.

Consequently, this definition leads to the following

Problem (P): Determine the optimal packing of circles for $n \geq 2$ in the unit square.

From another point of view, we may consider only the centers of the circles, so that we obtain the following problem: Locate n points in the unit square in such a way that the minimal distance between any two of them be maximal.

Definition 2.2 Let us denote the number of the points by $n \geq 2$. A *point arrangement with a minimal distance m_n in the unit square* is $A = (a_1, \ldots, a_n) \in A_{m_n}$, where $A_{m_n} = \{((x_1, y_1), \ldots, (x_n, y_n)) \in [0,1]^{2n} \mid (x_i - x_j)^2 + (y_i - y_j)^2 \geq m_n^2 \ (1 \leq i < j \leq n)\}$. A is an *optimal point arrangement*, if $A \in A_{\overline{m}_n}$, where $\overline{m}_n \max\limits_{A_{m_n} \neq \emptyset} m_n$.

This definition leads to

Problem (A): Determine the optimal point arrangements in the unit square for $n \geq 2$.

Problems (P) and (A) are known to be equivalent [Sza00]. The following relation holds between the radius $\overline{r}_n$ of the optimal packing and the distance $\overline{m}_n$ of the optimal point arrangement:

$$\overline{r}_n = \frac{\overline{m}_n}{2(\overline{m}_n + 1)}. \tag{1}$$

The following problem settings are also equivalent with problems (P) and (A), respectively:

- Find the smallest square of side $\overline{\rho}_n$ that contains n equal and non-overlapping circles with a radius of 1.
- Determine the smallest square of side $\overline{\sigma}_n$ that contains n points with mutual distances of at least 1.

Furthermore, it can be proved that

$$\overline{p}_n \overline{r}_n = 1 \quad \text{and} \quad \overline{\sigma}_n \overline{m}_n = 1. \tag{2}$$

The following definition summarizes some terms used in this chapter.
Definition 2.3 We say, that

- two circles are in *contact* in a packing if the distance between their centers is $2r_n$,
- a circle is *free* (a *rattler*) if it can be moved inside the square by a positive distance without getting in contact or overlapping another one,
- a circle is *fixed* if it isn't a free circle,
- the *density* of a packing in the unit square is $d_n = nr_n^2\pi$.

2.1 The PECS Problem as a Global Optimization Problem

The PECS problem is on the one hand a geometrical problem and on the other hand a continuous global optimization problem [TZ89]. Problem (A) can be written shortly as a $2n$ dimensional optimization problem in the following form:

$$\max_{s_k \in [0,1]^2,\ 1 \le k \le n} \ \min_{1 \le i < j \le n} \|s_i - s_j\| .$$

This problem can be considered as
a) *a continuous nonlinear constrained global optimization problem*:

$$\max_{x_i, y_i} \ t$$

subject to

$$\sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \ge t \quad (1 \le i < j \le n)$$

$$0 \le x_i, y_i \le 1, \quad 1 \le i \le n.$$

b) *a max-min optimization problem*:

$$\max_{x_i, y_i} \ \min_{1 \le i < j \le n} s_{ij}$$

subject to

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = s_{ij} \quad (1 \le i < j \le n)$$

$$0 \le x_i, y_i \le 1, \quad 1 \le i \le n.$$

c) *a DC programming problem* [6]:
A DC (difference of convex functions) programming problem is a mathematical programming problem, where the objective function can be given as a difference of two convex functions. The objective function of the PECS problem can be stated as the difference of the convex functions g and h:

$$g(z) = 2\sum_{j=1}^{2n} z_j^2, \text{ and}$$

$$h(z) = \max\left\{\left(2\sum_{j\in J\setminus J_{ik}} z_j^2 + (z_i+z_k)^2 + (z_{n+i}+z_{n+k})^2\right) : 1 \le i < k \le n\right\},$$

where $J = \{1,\dots,2n\}$, $z = (x_1,\dots,x_n,y_1,\dots,y_n)$, $J_{ik} = \{i,k,n+i,n+k\}$.
d) and finally as *an all-quadratic optimization problem.*

The general form of an all-quadratic optimization problem is

$$\min \quad \left[x^{\mathrm{T}}Q^0x + (d^0)^{\mathrm{T}}x\right]$$

subject to

$$x^{\mathrm{T}}Q^lx + (d^l)^{\mathrm{T}}x + c^l \le 0 \qquad l = 1,\dots,p$$
$$x \in P,$$

where Q^l $(l = 0,\dots,p)$ are real $(n+1)\times(n+1)$ matrices, d^l $(l = 0,\dots,p)$ are real $(n+1)$-dimensional vectors, c^l $(l = 1,\dots,p)$ are real numbers, p is the number of constraints, and P is a polyhedron. Solving the general case of an all-quadratic optimization problem is knew to be NP-hard.

The PECS problem with the following values is a special all-quadratic optimization problem with a linear objective function:

$$Q^0 = \mathbf{0}, \quad x^{\mathrm{T}} = (x_0,x_1,\dots,x_{2n}), \quad (d^0)^{\mathrm{T}} = (-1,0,\dots,0),$$
$$(d^l)^{\mathrm{T}} = \underline{0}, \;\; c^l = 0, \;\; p = \frac{n(n-1)}{2}, \;\; P = [0,\sqrt{2}]\times[0,1]^{2n},$$

and for all

$$1 \le i,j \le 2n+1,$$
$$1 \le l' < l'' \le n.$$

$$[Q^l]_{ij} = Q_{ij}^{l'l''} = \begin{cases} -1, & \text{if } i = j = \begin{cases} 2l', \\ 2l'', \\ 2l'+1, \\ 2l''+1, \end{cases} \\ 1, & \text{if } i = j = 1, \\ & \quad i = 2l''+1 \text{ and } j = 2l'+1, \\ & \quad i = 2l'' \quad\;\; \text{and } j = 2l', \\ & \quad i = 2l'+1 \text{ and } j = 2l''+1, \\ & \quad i = 2l' \quad\;\; \text{and } j = 2l'', \\ 0, & \quad\text{otherwise.} \end{cases}$$

In this model, x_0 is the minimal distance between the points. The coordinates of the i^{th} point $(1 \le i \le n)$ are (x_{2i-1}, x_{2i}).

The investigations show that those approaches are effective that use not only optimization models, but some geometrical properties of the problem, too. Before we study the most efficient earlier approaches, we give some theoretical lower bounds for $\overline{m}_n$.

3 Theoretical Lower Bound for $\overline{m}_n$

In this section we show a proof for the statement that the known $\sqrt{\frac{2}{\sqrt{3}n}}$ asymptotic formula is a lower bound of $\overline{m}_n$.

Proposition 1. *For every integer $n \ge 2$*

$$\sqrt{\frac{2}{\sqrt{3}n}} < \overline{m}_n.$$

Proof: Proposition 1 is equivalent with the following statement:

There can be at least $\frac{2}{\sqrt{3}}\sigma^2$ points located in a square of side σ such that the distances between the pairs of points are at least 1.

The proof is constructive, and it is based on the hexagonal grid packing. Let us divide the square into stripes of width $\frac{\sqrt{3}}{2}$. This division determines $\left\lfloor \frac{2\sigma}{\sqrt{3}} \right\rfloor + 1$ parallel zones. The first level is one side of the square. On this level $\lfloor\sigma\rfloor + 1$ points can be located, where the first point is in the edge of the square and the distance between the points is 1. On the second level the distance between the side of the square and the first point is $\frac{1}{2}$. On this level one can place $\lfloor \sigma - \frac{1}{2} \rfloor + 1$ points. If the number of the level is an odd number, then there are always $\lfloor\sigma\rfloor + 1$ points on this level and for an even number there are $\lfloor \sigma - \frac{1}{2} \rfloor + 1$ points. If $\left\lfloor \frac{2\sigma}{\sqrt{3}} \right\rfloor + 1$ is even, then there are $\frac{\left\lfloor \frac{2\sigma}{\sqrt{3}} \right\rfloor + 1}{2} \left(\lfloor\sigma\rfloor + \lfloor \sigma - \frac{1}{2} \rfloor + 2 \right)$ points in the square. If it is odd, then there are $\frac{\left\lfloor \frac{2\sigma}{\sqrt{3}} \right\rfloor}{2} \left(\lfloor \sigma - \frac{1}{2} \rfloor + \lfloor\sigma\rfloor + 2 \right) + \lfloor\sigma\rfloor + 1$ points. In both cases it is easy to see that the stated inequality holds, because $2\sigma < \lfloor \sigma - \frac{1}{2} \rfloor + \lfloor\sigma\rfloor + 2$, $\frac{2}{\sqrt{3}}\sigma - 1 < \left\lfloor \frac{2\sigma}{\sqrt{3}} \right\rfloor$ and have $\sigma - 1 < \lfloor\sigma\rfloor$, therefore

$$\frac{\frac{2\sigma}{\sqrt{3}} - 1 + 1}{2} 2\sigma < n \implies \frac{2}{\sqrt{3}}\sigma^2 < n \quad \text{(even case)},$$

and

$$\frac{\frac{2\sigma}{\sqrt{3}} - 1}{2} 2\sigma + \sigma - 1 + 1 < n \implies \frac{2}{\sqrt{3}}\sigma^2 < n \quad \text{(odd case)}. \qquad \square$$

The theoretical lower bounds of $\overline{m}_n$ were well improved by computer aided methods. These approaches usually fall into two classes: deterministic and stochastic methods. A typical way to find approximate packings in the latter case is to apply stochastic global optimization algorithms.

4 Some Earlier Approaches for Finding Approximate Packings

In this section we give an overview of some earlier methods to find approximate packings. Several strategies were used (e.g. nonlinear programming solvers, and the Cabri-Geometry software). Here we summarize some useful earlier approaches to find approximate packings for higher n values.

4.1 Energy Function Minimization

The PECS problem as a mathematical programming problem can be formalized in the following way:

$$\max \min_{1\le i<j\le n} ||s_i - s_j||$$

subject to

$$s_i \in [0,1]^2, \quad 1 \le i \le n.$$

By virtue of $\min\limits_{1\le i<j\le n} ||s_i - s_j|| = \lim\limits_{m\to -\infty} \left(\sum\limits_{1\le i<j\le n} ||s_i - s_j||^m \right)^{\frac{1}{m}}$, the problem is relaxed as

$$\min_{s_i\in[0,1]^2,\ 1\le i\le n} \sum_{1\le i<j\le n} \frac{1}{||s_i - s_j||^m}.$$

This objective function can be interpreted as a potential or energy function. A physical analogon of this approach is to regard the points as electrical charges (all positive or all negative) which are repulsing eachother. If the minimal distance between the charged particles increases, the corresponding value of the potential function decreases. K. J. Nurmela and P. R. J. Östergård [NO97] used a similar energy function with large values of m:

$$\sum_{1\le i<j\le n} \left(\frac{\lambda}{||s_i - s_j||^2} \right)^m.$$

Introducing $x_i = \sin(x_i^{'})$ and $y_i = \sin(y_i^{'})$, it transforms into an unconstrained optimization problem in variables $x_i^{'}, y_i^{'}$, where the coordinates of the centers of the circles fulfill the constraints $-1 \le x_i \le 1$, $-1 \le y_i \le 1$. They published candidate packings up to 50 circles using a combination of the Goldstein-Armijo backtracking linear search and the Newton method for the optimization.

4.2 Billiard Simulation

Let us consider a random arrangement of the points. Draw equal circles around the points without overlapping. Each circle can be considered as a ball with an initial radius, moving direction, and speed. Start the balls and increase slowly the common radius of them. The swing of each ball during the process will be less and less. The algorithm stops when the packing or a substructure of the packing becomes rigid for even. Using billiard simulation, R. L. Graham and B. D. Lubachevsky [GL96] reported several approximate packings for up to 50 circles and for some values beyond.

4.3 A Perturbation Method

D. W. Boll et al. [BDGL00] used a stochastic algorithm which gave improved packings for $n = 32, 37, 48$, and 50. A brief outline of their method is

```
1. Step: consider $n$ random points in the unit square and define $s = 0.25$,
2. Step: for each point
      a) perturb the place of the center by $s$ in North, South, East
         or West direction,
      b) if during the movement the distance between the point and
         its nearest neighbour becomes greater, update the new loca-
         tion of the point,
3. Step repeat Step 2 while movable points exist,
4. Step $s := s/1.5$, if $s > 10^{-10}$, and continue with Step 2.
```

Using the previous simple algorithm good candidate packings can be found after some millions of iterations. They have found unpublished approximate packings up to $n = 200$.

4.4 TAMSASS-PECS

The TAMSASS-PECS (Threshold Accepting Modified Single Agent Stochastic Search for Packing Equal Circles in a Square) method is based on the Threshold Accepting global optimization technique and a modified SASS local optimization algorithm. The algorithm starts with a pseudorandom initial packing with certain set, standard deviation and threshold level. The algorithm tries to improve the current solution by an iterative procedure. At every step it tries to find a better position of the actual point using a local search. The stopping criterion is based on the value of the standard deviation, which is decreased at every iteration. The framework of the method is the Threshold Accepting approach. It is a close alternative of the Simulated Annealing algorithms. It accepts every move that leads to a new approximate solution not much worse than the current one and rejects other moves. Using TAMSASS-PECS L. G. Casado et al. [CGSC01] reported approximate packings up to $n = 100$ and improved some earlier packings.

4.5 A Deterministic Approach Based on LP-Relaxation

The PECS problem can be regarded as an all-quadratic optimization problem, i.e. an optimization problem with not necessarily convex quadratic constraints. The hardness is due to the large number of constraints. This approach provides a rectangular subdivision branch-and-bound algorithm. To provide an upper bound at each node of the branch-and-bound tree, M. Locatelli and U. Raber used the special structure of the constraints and gave an LP-relaxation [LR02]. They have found candidate packings up to 39 circles proving the optimality theoretically within a given accuracy, except when $n = 36$ and 37. Moreover, a new solution for $n = 37$ has been detected, but not yet proved to be optimal within the given tolerance.

5 The MBS Algorithm

We will call our approach MBS (Modified Billiard Simulation). The basic idea is as follows: Distribute randomly n points inside the unit square and blow them up as increasing circles in a uniform manner. This can be done by incrementing the radii gradually from an initial value, which is a safe lower bound. In early stages of the process, when the distance between the small circles is much greater than their size and no collisions occur, there is no need to change their positions. As the circles grow, we have to deal with collisions between them, and with those between the circles and the boundaries. The iteration stops when the improvement is too small or the number of iterations is larger than a given number.

The efficiency of the MBS algorithm comes from a significant reduction of computational costs. The basic idea is as follows: It is not necessary to calculate and store the distance between two circles if they are too far from each other and will never meet. For the numerical calculation the program uses two matrices CCD and CED. Matrix CCD stores the adjacency between the objects themselves, and the matrix CED holds this between the objects and the sides of the square. At start, all matrix elements are set to NEAR which implies that only such pairs of circles will be checked during the calculation. When after thousands of collosions a mutual distance of a pair is great enough, then the value is set to FAR which means that this contact will never occur in later iterations. As long as the program runs the cost of the subroutine which determines the contacts will become less and less.

It is useful to consider not only random arrangements for the initial packing but hexagonal or other regular lattice packings too. Sometimes the relationship between the number of the circles and the structure of packing can predict a good initial configuration. The code and the found packings can be downloaded from the Packomania web-site: `http://www.packomania.com/`.

6 Approximate Packings up to n=200

In this section the found packings are reported up to $n = 200$ using the MBS algorithm. The packings were found by numerical computations. An important part of the investigations is to prvoide a guaranty that the arrangements with the given structures really (in mathematical sense) exist. A possible way for proving is based on a system of equations which corresponds to the packings. The solution of the system of equations (if it exists) prove rigorously the existence of the packing. To solve the system of equations is easy if we can guess the solution. Sometimes the structures of the packings help us to guess the exact coordinates of the circles (e.g. for grid packings). For the present situation any kind of reliable tools can be used (e.g. the software package INTBIS) to solve the system of equations.

The numerical value of m_n could be incorrect due to rounding, so we checked the calculations by interval arithmetic computations, too. Based on these computations the new m_n value is the minimum of the lower bounds of the mutual interval distances between the points. In a formula it is

$$m_n = \min_{1 \le i < j \le n} \underline{D_{ij}},$$

where

$$D_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2},$$

$$X_i = [x_i - \epsilon, x_i + \epsilon], \quad Y_i = [y_i - \epsilon, y_i + \epsilon],$$
$$X_j = [x_j - \epsilon, x_j + \epsilon], \quad Y_j = [y_j - \epsilon, y_j + \epsilon].$$

and (x_i, y_i) is the coordinates of the ith point $(1 \le i \le n)$. Underlining means the lower bound. The capital letters denote intervals. The accuracy of the calculations was 10^{-14}. A fully interval arithmetic based method to validate the solutions was proposed by M.Cs. Markót [Mar03, Mar04, MC05].

Figure 2 shows the density plots of packings for up to 200 circles. It is interesting to observe the local maxima of the graphs. Usually the related packings are grid or near-grid packings.

Tables 1–3 use bold face for the improved packings compared to other, previous results, and italic style denotes that there are better known values, but the coordinates have not been published yet. In 66 cases we have found similar values to the packings reported in the literature. In 24 cases the results are worse than Dave Boll et al.'s packings. The number of the improved lower bounds of the packings is 110. In these tables we summarized the numerical results.

The figures of the packings are available in PostScript files including the coordinates of the circles at the mentioned web-site. Figure 3 represent two nice approximate packings with free circles.

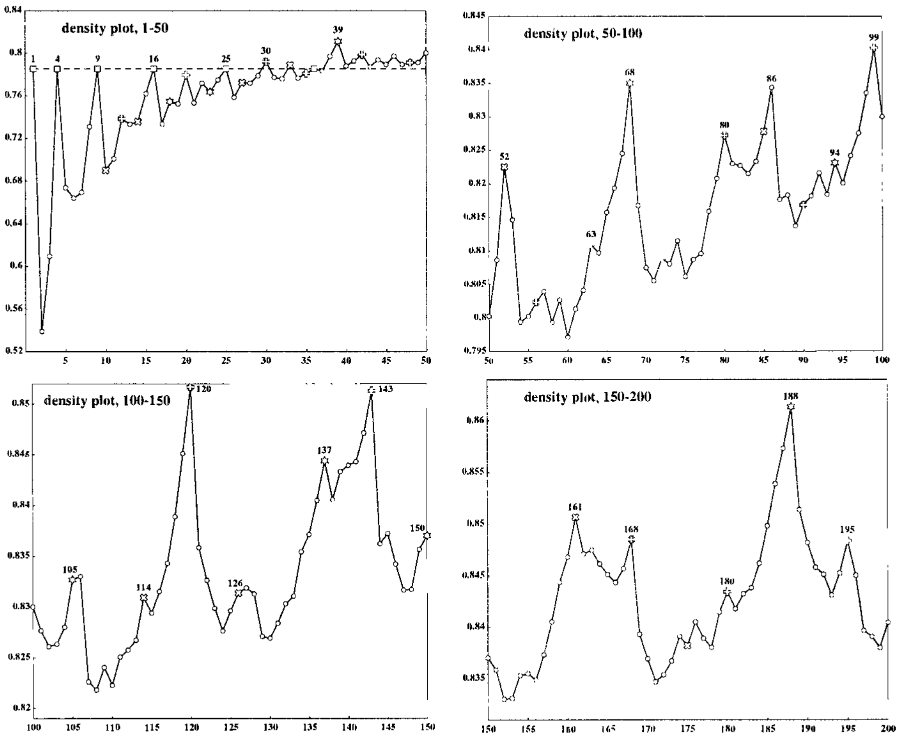
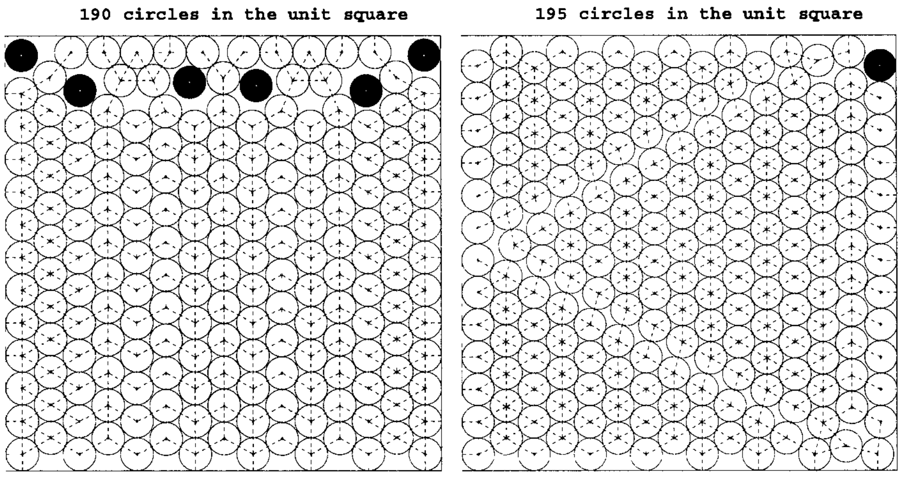


Fig. 2. Density plots for up to 200 circles


190 circles in the unit square
195 circles in the unit square

```
radius   = 0.037695999979   density  = 0.848191685066      radius   = 0.037213368627   density  = 0.848364435455
distance = 0.081539419899   contacts = 421                 distance = 0.080411503065   contacts = 514
```

Fig. 3. Approximate packings for 190 and 195 circles

Table 1. The numerical results for $n = 2 - 36$ circles

n	exact r_n	exact m_n	approx. m_n	d_n
2	$\frac{1}{2}(2-\sqrt{2})$	$\sqrt{2}$	1.4142135624	0.5390120845
3	$\frac{1}{2}(8-5\sqrt{2}+4\sqrt{3}-3\sqrt{6})$	$\sqrt{6}-\sqrt{2}$	1.0352761804	0.6096448087
4	$\frac{1}{4}$	1	1.0000000000	0.7853981634
5	$\frac{1}{2}(-1+\sqrt{2})$	$\frac{1}{2}m_2$	0.7071067812	0.6737651056
6	$\frac{1}{46}(-13+6\sqrt{13})$	$\frac{1}{6}\sqrt{13}$	0.6009252126	0.6639569095
7	$\frac{1}{13}(4-\sqrt{3})$	$4-2\sqrt{3}$	0.5358983849	0.6693108268
8	$\frac{1}{4}(1+\sqrt{2}-\sqrt{3})$	$\frac{1}{2}m_3$	0.5176380902	0.7309638253
9	$\frac{1}{6}$	$\frac{1}{2}$	0.5000000000	0.7853981634
10	–	–	0.4212795440	0.6900357853
11	(see below)	(see below)	0.3982073102	0.7007415778
12	$\frac{1}{382}(-34+15\sqrt{34})$	$\frac{1}{15}\sqrt{34}$	0.3887301263	0.7384682239
13	–	–	0.3660960077	0.7332646949
14	$\frac{1}{33}(6-\sqrt{3})$	$\frac{2}{13}(4-\sqrt{3})$	0.3489152604	0.7356792555
15	$\frac{1}{2}r_3$	$2\,r_8$	0.3410813774	0.7620560109
16	$\frac{1}{8}$	$\frac{1}{3}$	0.3333333333	0.7853981634
17	–	–	0.3061539853	0.7335502633
18	$\frac{1}{262}(-13+12\sqrt{13})$	$\frac{1}{2}m_6$	0.3004626063	0.7546533579
19	–	–	0.2895419920	0.7523078967
20	$\frac{1}{482}(65-8\sqrt{2})$	$\frac{1}{16}(6-\sqrt{2})$	0.2866116524	0.7794936869
21	–	–	0.2718122554	0.7533577029
22	–	–	0.2679584016	0.7716801121
23	$\frac{1}{2}(-7-5\sqrt{2}+4\sqrt{3}+3\sqrt{6})$	$\frac{1}{4}m_3$	0.2588190451	0.7636310321
24	$\frac{1}{92}(21-5\sqrt{2}+3\sqrt{3}-4\sqrt{6})$	r_3	0.2543330950	0.7749632598
25	$\frac{1}{10}$	$\frac{1}{4}$	0.2500000000	0.7853981634
26	–	–	0.2387347572	0.7584690905
27	$\frac{1}{3022}(-89+40\sqrt{89})$	$\frac{1}{40}\sqrt{89}$	0.2358495283	0.7723114565
28	–	–	0.2305354936	0.7718541114
29	–	–	0.2268829007	0.7789062418
30	$\frac{1}{1202}(126-5\sqrt{10})$	$\frac{1}{75}(20-\sqrt{10})$	0.2245029645	0.7920190265
31	–	–	0.2175472916	0.7772974787
32	–	–	0.2131341934	0.7757618736
33	–	–	0.2113283841	0.7888523039
34	$\frac{1}{2}m_8$	$2r_{23}$	0.2056046467	0.7766490643
35	(see below)	$2\,r_{24}$	0.2027636009	0.7812272130
36	$\frac{1}{12}$	$\frac{1}{5}$	0.2000000000	0.7853981634

$$r_{11} = \frac{\left(176 - 9\sqrt{2} - 14\sqrt{3} - 13\sqrt{6} - 2\sqrt{-16523 + 12545\sqrt{2} - 9919\sqrt{3} + 6587\sqrt{6}}\right)}{568}$$

$$m_{11} = \frac{1}{4}\left(-4 - 3\sqrt{2} + 2\sqrt{3} + 3\sqrt{6} + \left(4 + \sqrt{2} - 2\sqrt{3} - \sqrt{6}\right)\sqrt{1 + 2\sqrt{2}}\right)$$

$$r_{35} = \frac{1}{772}(112 - 17\sqrt{2} + 8\sqrt{3} - 15\sqrt{6})$$

Table 2. The numerical results for $n = 37 - 124$ circles

n	approx. m_n	d_n	n	approx. m_n	d_n
37	0.1964291842	0.7833027206	**81**	0.1283368559	0.8230005836
38	0.1953423041	0.7970425568	**82**	0.1274269118	0.8227146966
39	0.1943650631	0.8111790272	**83**	0.1264543531	0.8215014743
40	0.1881755220	0.7879795188	**84**	0.1257627022	0.8233399269
41	0.1860995118	0.7927238993	85	0.1253084290	0.8278015474
42	0.1842770721	0.7986842786	*86*	0.1250425340	0.8343840262
43	0.1801911354	0.7872641664	**87**	0.1228265796	0.8176519577
44	0.1786392456	0.7938428078	**88**	0.1220983545	0.8183334905
45	0.1757163141	0.7894442684	**89**	0.1209431296	0.8137202694
46	0.1744593608	0.7971871319	90	0.1204480494	0.8168616061
47	0.1712705638	0.7892938820	**91**	0.1198126022	0.8181738104
48	0.1694054293	0.7911440338	**92**	0.1193622666	0.8216190446
49	0.1673860768	0.7912169895	**93**	0.1183841706	0.8184234765
50	0.1665265773	0.8002721839	**94**	0.1180563847	0.8231316147
51	0.1656183743	0.8086569481	**95**	0.1171194312	0.8201101448
52	0.1653862379	0.8225308420	96	0.1167579991	0.8241689679
53	0.1626480663	0.8146424042	**97**	0.1163574839	0.8276442131
54	0.1591395163	0.7994076230	**98**	0.1165351364	0.8335521941
55	0.1575557475	0.8002712972	**99**	0.1160181348	0.8402999370
56	0.1561565004	0.8023517295	**100**	0.1145801945	0.8300157794
57	0.1547474069	0.8039656738	**101**	0.1137678096	0.8276766917
58	0.1526925313	0.7993307243	**102**	0.1130265581	0.8261170420
59	0.1515619183	0.8026893318	**103**	0.1124324277	0.8263480419
60	0.1495056540	0.7971391564	**104**	0.1119552283	0.8280134249
61	0.1485441266	0.8013741245	**105**	0.1117113035	0.8327015567
62	0.1474526798	0.8041134778	**106**	0.1111479979	0.8330195984
63	0.1468193136	0, 8109737806	*107*	0.1098006344	0.8226089313
64	0.1453677544	0.8096850385	108	0.1091766948	0.8218111889
65	0.1446990147	0.8157400181	109	0.1087836926	0.8240438322
66	0.1438039660	0.8193554530	110	0.1081056069	0.8222742690
67	0.1430855758	0.8245156558	**111**	0.1077672183	0.8250669455
68	0.1429094775	0.8350205982	**112**	0.1072831265	0.8257591285
69	0.1399481818	0.8167765737	*113*	0.1068253164	0.8267201679
70	0.1379067766	0.8075060206	**114**	0.1066053963	0.8309359404
71	0.1366129972	0.8055769545	**115**	0.1059840023	0.8294126663
72	0.1358499279	0.8089083021	**116**	0.1056270416	0.8315355075
73	0.1347098276	0.8080563922	**117**	0.1053200675	0.8342993237
74	0.1339986726	0.8115167751	**118**	0.1051451721	0.8389032937
75	0.1324888136	0.8061980168	**119**	0.1050811970	0.8450812908
76	0.1317300376	0.8086999566	**120**	0.1050454468	0.8516581650
77	0.1308410780	0.8095910198	**121**	0.1034896517	0.8358581333
78	0.1304607726	0.8158933303	*122*	0.1028022705	0.8328448925
79	0.1299652027	0.8208069227	**123**	0.1021525019	0.8298691626
80	0.1296133854	0.8272178885	**124**	0.1015480773	0.8276525614

Table 3. The numerical results for $n = 125 - 200$ circles

n	approx. m_n	d_n	n	approx. m_n	d_n
125	0.1012318916	0.8296158463	**163**	0.0885685524	0.8474713225
126	0.1009062359	0.8313727312	**164**	0.0881985853	0.8461369376
127	0.1005031141	0.8318980160	**165**	0.0878493719	0.8451107320
128	0.1000307339	0.8312987414	**166**	0.0875180593	0.8443458471
129	0.0993263118	0.8270941821	**167**	0.0873073510	0.8456746787
130	0.0988937215	0.8269119921	**168**	0.0871824504	0.8485011578
131	0.0986288136	0.8292143817	**169**	0.0863882436	0.8392971739
132	0.0982888835	0.8303084177	**170**	0.0859789843	0.8369135174
133	0.0979321354	0.8310764541	*171*	0.0855822297	0.8346948524
134	0.0978107863	0.8354360146	*172*	0.0853489718	0.8353646965
135	0.0975218520	0.8371459461	**173**	0.0851552703	0.8367106190
136	0.0973414248	0.8405056543	**174**	0.0850212754	0.8391079841
137	0.0971962814	0.8443861367	**175**	0.0847072864	0.8381936262
138	0.0965684977	0.8405593584	**176**	0.0845723809	0.8405094143
139	0.0963607140	0.8433304280	**177**	0.0842258678	0.8389085248
140	0.0960210816	0.8439433382	**178**	0.0839212683	0.8380279161
141	0.0956706024	0.8443178921	**179**	0.0838518837	0.8414507200
142	0.0954741493	0.8471212060	**180**	0.0837056682	0.8434307533
143	0.0953634647	0.8512820167	**181**	0.0833661059	0.8417768920
144	0.0940769937	0.8362256095	*182*	0.0831953206	0.8432289693
145	0.0937839766	0.8372440265	*183*	0.0829799252	0.8438130489
146	0.0932472250	0.8342146320	*184*	0.0828621097	0.8462006652
147	0.0927445712	0.8316620732	*185*	0.0828104531	0.8498202051
148	0.0924063478	0.8317384492	**186**	0.0827835097	0.8539004243
149	0.0923037281	0.8356565148	**187**	0.0827219524	0.8573124997
150	0.0920498844	0.8370331951	**188**	0.0826931189	0.8613421953
151	0.0916459689	0.8358530518	**189**	0.0819385675	0.8513795535
152	0.0911461597	0.8329987760	**190**	0.0815394198	0.8481916850
153	0.0908262169	0.8330913575	**191**	0.0811842378	0.8457992069
154	0.0906340269	0.8352857436	**192**	0.0809200493	0.8451158427
155	0.0903257963	0.8354733542	*193*	0.0805888840	0.8430949394
156	0.0899733630	0.8348541653	*194*	0.0804752001	0.8452518615
157	0.0898047212	0.8373181432	**195**	0.0804115030	0.8483644354
158	0.0896806181	0.8405154516	**196**	0.0800186870	0.8450185742
159	0.0895956726	0.8443652173	*197*	0.0795255746	0.8396607811
160	0.0894304663	0.8468018971	*198*	0.0792782634	0.8390666657
161	0.0893487418	0.8506653983	*199*	0.0790085605	0.8379950934
162	0.0888450720	0.8471092098	**200**	0.0789188489	0.8404343621

Summary

The chapter gives a short overview of some earlier models and computer aided methods for the packing of equal circles in a square problem. Theoretical lower bounds are given for every n and based on a modified billiard simulation algorithm several approximate packings are reported up to $n = 200$.

Acknowledgement. The authors are thankful to Tibor Csendes (University of Szeged, Hungary) for his suggestions and comments on this chapter.

References

[BDGL00] Boll, D.W, Donovan, J., Graham, R.L., Lubachevsky, B.D.: Improving dense packings of equal disks in a square. The Electronic Journal of Combinatorics, **7**, #R46 (2000)

[Bol04] Bolyai, W.: Tentamen Juventutem Studiosam in Elementa Matheseos Purae, Elementaris Ac Sublimioris, Methodo Intuitiva, Evidentiaque Huic Propria, Introducendi (in Latin, 1832-33). 2nd ed. Volume **2**, 119–122 (1904)

[CGSC01] Casado, L.G., García, I., Szabó, P.G., Csendes, T.: Packing equal circles in a square II. — New results for up to 100 circles using the TAMSASS-PECS stochastic algorithm. In: Giannessi, F., Pardalos, P., Rapcsák, T. (eds) Optimization Theory: Recent Developments from Mátraháza. Kluwer Academic Publishers, Dordrecht Boston London 207–224 (2001)

[GL96] Graham, R.L., Lubachevsky, B.D.: Repeated patterns of dense packings of equal circles in a square. The Electronic Journal of Combinatorics, **3**, #R16 (1996)

[GPW90] de Groot, C., Peikert, R., Würtz, D.: The optimal packing of ten equal circles in a square. In: IPS Research Report, Eidgenössiche Technische Hochschule, Zürich, No. 90-12, August (1990)

[HT99] Horst, R., Thoai, N.V.: D.C. programming: overview. Journal of Optimization Theory and Applications, **103**, 1–43 (1999)

[KW87] Kirchner, K., Wengerodt, G.: Die dichteste Packung von 36 Kreisen in einem Quadrat. Beiträge zur Algebra und Geometrie, **25**, 147–159 (1987)

[LR02] Locatelli, M., Raber, U.: Packing equal circles in a square: a deterministic global optimization approach. Discrete Applied Mathematics, **122**, 139–166 (2002)

[Mar03] Markót, M.Cs.: Reliable Global Optimization Methods for Constrained Problems and Their Application for Solving Circle Packing Problems (in Hungarian). Ph.D. Thesis, University of Szeged, Szeged (2003)

[Mar04] Markót, M.Cs.: Optimal packing of 28 equal circles in a unit square — the first reliable solution. Numerical Algorithms, **37**, 253–261 (2004)

[MC05] Markót, M.Cs., Csendes T.: A new verified optimization technique for the "packing circles in a unit square" problems. SIAM Journal on Optimization, **16**, 193–219 (2005)

[Mel97] Melissen, H.: Packing and Covering with Circles. Ph.D. Thesis, University of Utrecht, Utrecht (1997)

[Mos60] Moser, L.: Problem 24. Canadian Mathematical Bulletin, **8**, 78 (1960)

[NO97] Nurmela, K.J., Östergård, P.R.J.: Packing up to 50 equal circles in a square. Discrete & Computational Geometry, **18**, 111–120 (1997)

[NO99] Nurmela, K.J., Östergård, P.R.J.: More optimal packings of equal circles in a square. Discrete & Computational Geometry, **22**, 439–457 (1999)

[NOS99] Nurmela, K.J., Östergård, P.R.J., aus dem Spring, R.: Asymptotic behaviour of optimal circle packings in a square. Canadian Mathematical Bulletin, **42**, 380–385 (1999)

[PWMG92] Peikert, R., Würtz, D., Monagan, M., de Groot, C.: Packing circles in a square: a review and new results. In: Kall, P. (ed) System Modelling and Optimization. Vol. 180 of Lecture Notes in Control and Information Sciences, Springer–Verlag, Berlin 45–54 (1992)

[SCCG01] Szabó, P.G., Csendes, T., Casado, L.G., García, I.: Packing equal circles in a square I. — Problem setting and bounds for optimal solutions. In: Giannessi, F., Pardalos, P., Rapcsák, T. (eds) Optimization Theory: Recent Developments from Mátraháza. Kluwer Academic Publishers, Dordrecht Boston London, 191–206 (2001)

[Sch65] Schaer, J.: The densest packing of nine circles in a square. Canadian Mathematical Bulletin, **8**, 273–277 (1965)

[SM65] Schaer, J., Meir, A.: On a geometric extremum problem. Canadian Mathematical Bulletin, **8**, 21–27 (1965)

[SMC05] Szabó, P.G.,Markót M.Cs., Csendes, T.: Global optimization in geometry — circle packing into the square. In: Audet, C., Hansen, P., Savard, G. (eds) Essays and Surveys in Global Optimization, GERAD 25th Anniversary Series, 7, Springer, New York 233–265 (2005)

[Sza00] Szabó, P.G.: Some new structures for the "equal circles packing in a square" problem. Central European Journal of Operations Research, **8**, 79–91 (2000)

[TZ89] Törn, A., Zilinskas, A.: Global Optimization. Lecture Notes in Computer Science, No 350, Springer Verlag, Heidelberg (1989)

[Wen83] Wengerodt, G.: Die dichteste Packung von 16 Kreisen in einem Quadrat. Beiträge zur Algebra und Geometrie, **16**, 173–190 (1983)

[Wen87a] Wengerodt, G.: Die dichteste Packung von 14 Kreisen in einem Quadrat. Beiträge zur Algebra und Geometrie, **25**, 25–46 (1987)

[Wen87b] Wengerodt, G.: Die dichteste Packung von 25 Kreisen in einem Quadrat. Ann. Univ. Sci. Budapest Eötvös Sect. Math., **30**, 3–15 (1987)

Global Optimization of Network Length and Simulation of Film Evolution

Vydūnas Šaltenis

Institute of Mathematics and Informatics, Akademijos 4, Vilnius 08663, Lithuania
saltenis@ktl.mii.lt

Summary. An idealized thin film when subjected to some constraints acquires length-minimizing properties. The length-minimizing curve of the film may achieve a configuration close to the Steiner minimal tree in the Euclidean plane. The Steiner problem asks for the shortest network that spans a given set of fixed points in the Euclidean plane. The main idea is to use the mathematical model for an idealized wet film, connecting the fixed points with some liquid inside the film. Gradually decreasing the interior area, the film may achieve the globally optimal solution. A system of equations and an algorithm for simulating wet film evolution are presented here. Computational experiments and tests show the abilities of global optimization. The investigation of a simple case illustrates how the film evolution leads up to the global optimum.

Key words: global optimization, Steiner problem, unconventional computing, wet film, simulation.

1 Introduction

An idealized soap film when subjected to some constraints acquires length-minimizing properties. Courant and Robbins [CR41] paid attention to the fact that the length-minimizing curve of a thin film may achieve a configuration close to the Steiner minimal tree in the Euclidean plane (also see [Mor92]).

Minimizing the network length is one of the oldest optimization problems. In the seventeenth century the following problem was suggested: find a point P that minimizes the sum of distances from P to each of the three given points in the plane. Fermat, Torricelli, and Cavalieri independently derived solutions to this problem. P may be inside a triangle formed by the points; the angles formed by the lines connecting P to each of the points are all 120°. P may also be one of the vertices, and the angle formed by connecting P to the other vertices is greater than or equal to 120°.

Later the problem was extended by allowing the addition of an arbitrary number of points. Courant and Robbins [CR41] popularized this problem, which became known as the Steiner problem.

The Steiner problem asks for the shortest network that spans a given set of n fixed points in the Euclidean plane. Any network solving the Steiner problem must be a tree, which is called a Steiner Minimal Tree (SMT). It may contain vertices different from the points to be connected. These points are called Steiner points.

The publications on the Steiner problem are extremely numerous. We will mention only surveys [HRW92, IT94, Cie98, PS02]. Melzak [Mel61] was the first to establish the basic properties of the SMT (see also [GP66]):

- the degree of each vertex is at most three;
- the degree of each Steiner point equals three;
- any two edges incident to a Steiner point meet at an angle of $120°$;
- there are at most $n - 2$ Steiner points;
- SMT has at most $2n - 3$ edges;
- if there is a Steiner point in the tree, then at least one of them has two given points as neighbors;
- the SMT is a Minimal Spanning Tree (MST) for the summary set of given fixed points and of Steiner points.

The MST is usually stated as follows. Given such a weighted (connected) graph, one would then wish to select for construction a set of communication links that would connect all the vertices and have the minimal total cost of a given graph (see for example [GH85]).

Any network with these properties is defined as a Steiner tree. However, the properties are insufficient for the length of the Steiner tree to be minimal. The number of different topologies grows exponentially as n increases. It has been shown that the Euclidean Steiner problem is NP hard [GGJ77]. There can be no polynomial-time algorithm for generating minimal Steiner trees.

The problem can be reduced to a continuous but extremely multimodal optimization problem of how to define coordinates of $N \leq n$ additional points that minimize the sum of the length of all arcs. When the coordinates of all additional points are fixed, the topology can be easily found using the well-known algorithms for solving the MST problem.

The Steiner problem can be seen in nature. Place pins to represent fixed points into a flat surface and place a flat surface on the other ends of the pins. Dip this apparatus into a soap solution and remove. The soap film formed between the pins (fixed points) will minimize the overall surface area (see Fig. 1). Since the distance between the surfaces is constant, the film will also minimize the total length forming Steiner points in the process.

Idealized infinitely thin soap films of this sort are called dry [BM98]. We can get not only one stable dry film configuration of different total length for the same fixed points. For example, the configuration in Fig. 1, which

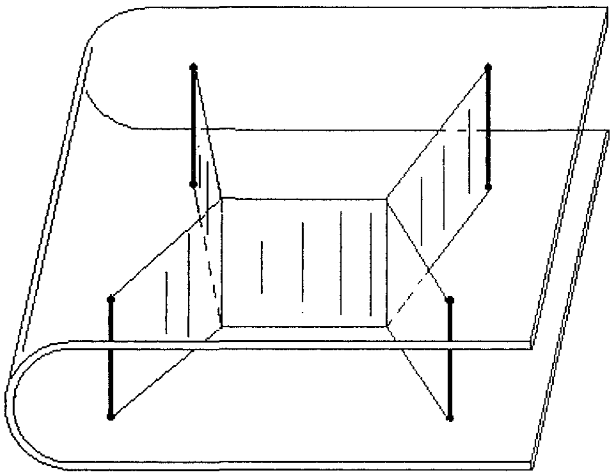


Fig. 1. Illustration of a dry soap film forming Steiner points

may be denoted as horizontal double Y, is not unique; the vertical double Y configuration is also stable.

The main idea is to use the mathematical model for an idealized wet soap film, connecting the fixed points with some liquid inside the film. The wet film tries to shrink to the minimum length, subject to a constraint. The constraint can be either that the liquid has a fixed interior area, or that the liquid is under a fixed pressure.

We can find an attempt to simulate the evolution of an idealized soap film to solve the Steiner problem in [Jak65]. However, the results of these computational experiments were not published.

The goal of this chapter is to investigate the global optimality properties of a wet film evolution model introduced by the author ([Sal99, Sal00]). The film model and investigation results may be used as one of possible unconventional computing models [Ada02].

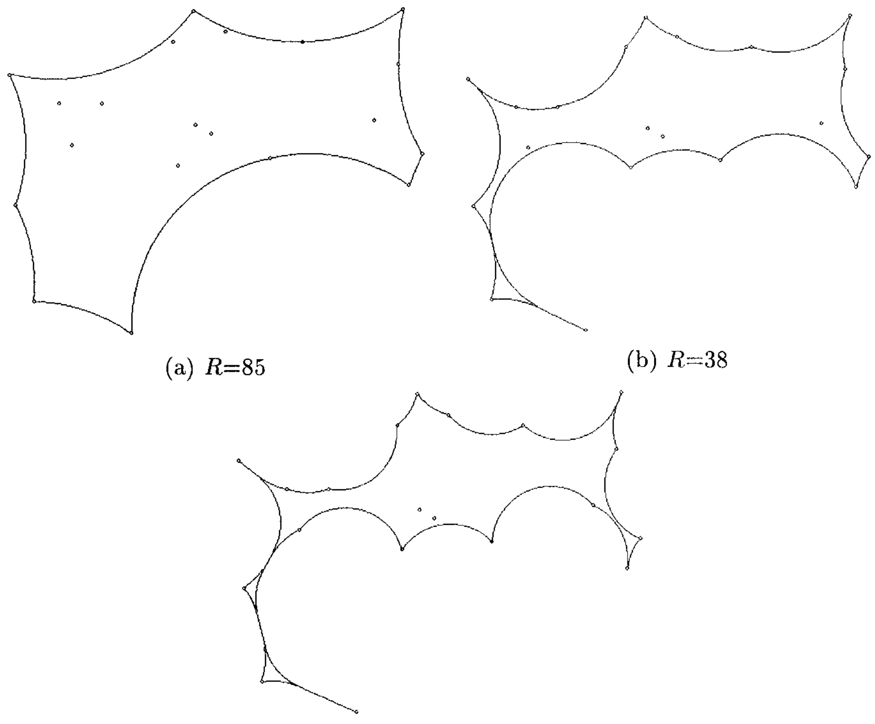
2 Simulation of Wet Film Evolution

The theorems in [HM96] and [BM98] state that a length-minimizing curve of a wet film enclosing the region of a fixed area to be composed of:

- circular arcs of equal positive outward curvature and
- straight-line two-ply segments.

The radius R of arcs depends on the external pressure p:

$$R = 1/p.$$



(a) R=85

(b) R=38

(c) R=29

Fig. 2. Three first stages of wet film evolution ($n = 20$)

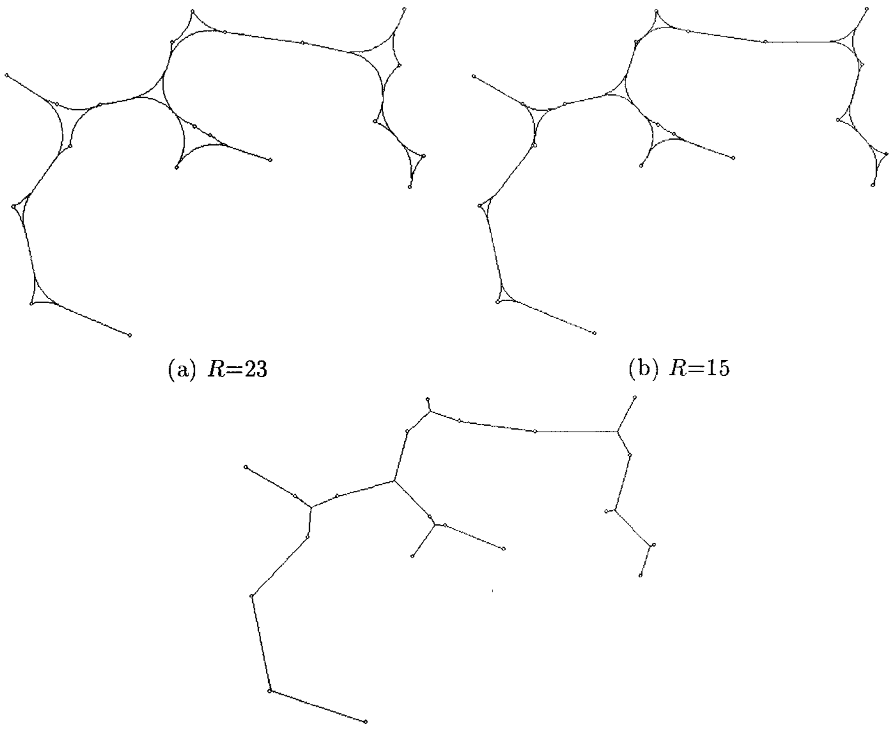
Figures 2 and 3 illustrate some intermediate evolution stages and the shapes of a wet film connecting twenty fixed points for various radii R and decreasing the interior area. The illustrations are useful to substantiate the idea, that by gradually changing radius R (or equivalently changing the pressure) of the length-minimizing curve, we are able to achieve a configuration of the length-minimizing curve close to the SMT.

In the beginning (Fig. 2a), the curve includes only 11 fixed points. Figure 2b shows the stage, when some fixed not connected points are touched by the curve, and one pair of the opposite arcs touched each other. Figure 2c shows the stage when the next pair of the opposite arcs touched each other.

At further stages (Fig. 3a and 3b), the interior area and radius R decrease forming the shape of the SMT. The last stage (Fig. 3c) only corrects the Steiner point coordinates.

2.1 Definitions

Points to be connected will be referred to as *fixed points* P_i ($i = 1, ..., n$).



(a) R=23 (b) R=15

(c) R=0.1

Fig. 3. Three last stages of wet film evolution ($n = 20$)

We also define the *corners* C_j ($j = 1, ..., m$) of the length-minimizing curve. They are *wet* if their coordinates are the same as the coordinates of the respective fixed points (Fig. 4a); otherwise, they are *dry* (Fig. 4b, c). The maximal number of corners may amount to $3(n-2)$.

Each corner has its *base*. A base may be:

- a fixed point, directly connected by means of a straight-line segment to the corner (the fixed point P_{jB} in Fig. 4b);
- another corner, directly connected by means of a straight-line segment to the corner (the corner C_{jB} in Fig. 4c).

The wet corner and its base have the same coordinates.

Each corner C_j has two centers of its arcs: O_{j1} and O_{j2} (Fig. 4).

2.2 A System of Equations

Let us denote a Euclidean distance between two points A_i and A_j as $d(A_i, A_j)$. Let radius R be given.

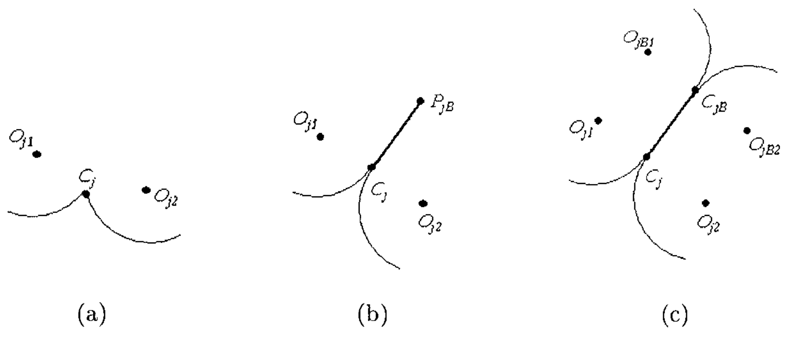


Fig. 4. Characteristic points of the length-minimizing curve

Then the coordinates of centers and corners of the length-minimizing curve satisfy the following system of nonlinear equations.

1. For each wet corner C_j the base P_{jB} of which has the same coordinates, we have the following two equations (see Fig. 4a):

$$d(P_{jB}, O_{j1}) = R, \tag{1}$$
$$d(P_{jB}, O_{j2}) = R. \tag{2}$$

2. For each dry corner C_j the base of which is a fixed point P_{jB}, we have the following equation (see Fig. 4b):

$$d(P_{jB}, O_{j1}) = d(P_{jB}, O_{j2}). \tag{3}$$

3. For each dry corner C_j whose base is another corner C_{jB}, we have the following equation (see Fig. 4c):

$$d(C_{jB}, O_{j1}) = d(C_{jB}, O_{j2}). \tag{4}$$

4. For each dry corner C_j we have the following equation (see Figs. 4b, c):

$$d(O_{j1}, O_{j2}) = 2R. \tag{5}$$

The coordinates of dry corners C_{jB} : $x(C_{jB})$ and $y(C_{jB})$ may be simply calculated from the coordinates of two corresponding centers O_{jB1}, O_{jB2} (the corners are located in the middle of centers):

$$C_{jB}) = (x(O_{jB1}) + x(O_{jB2}))/2,$$
$$C_{jB}) = (y(O_{jB1}) + y(O_{jB2}))/2.$$

The unknowns of the system are the coordinates of centers O. The number of the unknowns is equal to the number of equations because each center belongs to two corners (see Fig. 4).

2.3 Initial Stage of Simulation

The simulation of film evolution begins with extremely high values of radius R. Usually, the initial values of R are defined two or three times higher than the longest distance between the fixed points. Only a part of the fixed points belonging to a convex hull of fixed points is included into the length-minimizing curve.

At the initial stage all corners are wet, therefore the system consists only of equations of type (1) and (2). In this case, the unknowns may be simply solved in an explicit way, because the system may be divided into the equation pairs and the arcs have positive outward curvature.

The values of the unknowns obtained are used as initial approximations for the next step of simulation with changed radius R.

2.4 Steps of Simulation

The steepest descent method [BF93] is used for solving the system of nonlinear equations in the general case. The results of simulation of the previous step are used as initial approximations. The changes of radius ΔR from step to step are small; therefore the number of iterations for a sufficient accuracy is extremely small.

At each step of changing R we check the occurrence of the next events:

- a wet corner became dry;
- a dry corner became wet;
- a pair of the opposite arcs touched each other;
- a pair of the opposite arcs separated from each other;
- a fixed point, not connected so far, was touched by the length-minimizing curve;
- a fixed point, connected to the length-minimizing curve so far, separated.

The respective corrections must be made in the equation system depending on the event. For example, when a wet corner becomes dry, its two equations of type (1, 2) must be changed to type (3, 5).

When a pair of the opposite arcs touches each other, two new dry corners, similar to that in Fig. 4c, emerge. Four new equations of type (4, 5) must be included into the equation system.

When a fixed point, not connected so far, is touched by the length-minimizing curve, a new wet corner is born. A pair of new equations of type (1, 2) must be included.

The simulation process stops when the value of radius R achieves low values. When this global convergence condition is satisfied the curve of a wet film is composed practically of straight-line segments formative the Steiner problem solution.

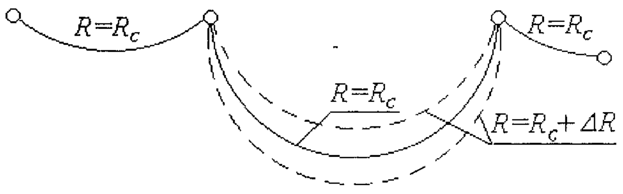


Fig. 5. A fragment of two wet corners under the dead-point situation

3 Dead-Point Situations in the Simulation Process

The results of simulation at each step of reducing radius R are used as the initial approximations for the next step of simulation. However, for some allocation of fixed points at specific stages of simulation, some dead-point situations may occur. In such a case the solution of equations of type (1) – (5) cannot be found for the values of radius R less than some limit value R_c.

The dead-point situations make the continuous evolution of a wet film to the zero interior area impossible, reducing radius R step by step. But in fact there always exists a continuous further evolution of the film under this situation. After achieving the critical radius, we must pass to the stage of temporarily increasing the radius R. It does not mean the return to the same past trajectory of the film coordinates.

A special paper of the author [Sal00] is dedicated to the dead-point situations.

We illustrate only the simplest case of a fragment of two wet corners (see Fig. 5). We can see that decreasing of radius is impossible from the critical value $R = R_c$ in the case where the arc angle is equal to 180^0. However, a continuous evolution of the curve is possible by temporarily increasing the radius with arc angles greater than 180^0.

4 Computational Experiments and Tests

4.1 Tests

Some numerical comparison with global optimization of Steiner points was performed. The data files of Steiner test problems, exactly solved in [Bea92], were used to verify that the wet film evolution really approaches the globally optimal solution. The data for test problems solved in case $n = 10$ are presented in Table 1, and their respective exact solutions obtained are illustrated in Fig. 6. All test problems were solved exactly.

Table 1. Data of test problems ($n = 10$)

Point	Test No 1		Test No 2		Test No 3	
No	x	y	x	y	X	y
1	.8183892	.4929768	.1470158	.6131368	.9819494	.9247995
2	.4060003	.6464021	.1251936	.2125058	.7895648	.6555445
3	.6673119	.3007983	.7411042	.2036110	.6774995	.1526794
4	.3283856	.4356078	.8044316	.4252316	.4818264	.5863090
5	.7079213	.7496262	.2116787	.2721307	.1042028	.3883077
6	.3988946	.9371529	.6624317	.7419167	.0373098	.7508847
7	.1729512	.0135599	.8043960	.5698598	.9812760	.1422699
8	.4489141	.0562685	.6684968	.2824909	.7802728	.2131439
9	.7066958	.1568812	.4043257	.1895910	.7190162	.1872961
10	.6843121	.5391302	.4978816	.2117592	.5176013	.1996263

Point	Test No 4		Test No 5	
No	x	y	x	Y
1	.4811719	.7890001	.2645109	.7072475
2	.7307729	.2483504	.7940569	.2692202
3	.6416435	.3074510	.0027461	.2156894
4	.7247301	.6898011	.8038497	.0668658
5	.2185930	.6933254	.5500578	.4690731
6	.2054522	.6914300	.2657439	.2869247
7	.2818990	.0855741	.2719144	.3460835
8	.0223121	.8674689	.9310821	.4768678
9	.6497331	.7051559	.3302928	.4342739

4.2 Special Test Example

A special test example was constructed by allocating five fixed points so as "to set a trap" for the optimal film evolution. The coordinates of fixed points are presented in Fig. 7a. We can see that four fixed points are located on the corners of the rectangle that is slightly flattened. The fifth fixed point is located in the interior area, and at the initial stage of film evolution the film does not touch it.

The optimal solution is illustrated in Fig. 7f, and the film evolution really gives this exact solution with a vertical double Y configuration. However, at the initial stages, when only four points are included in the curve (Fig. 7a,b), evolution tendencies are directed to the horizontal double Y configuration (the distance between the top and bottom arcs is shorter than between the left and right arcs). But later on the fifth point stops the evolution of the top and bottom arcs (Fig. 7c), the left and right arcs move more rapidly and the film evolution corrects their movement in the optimal way (Fig. 7d, e, f).

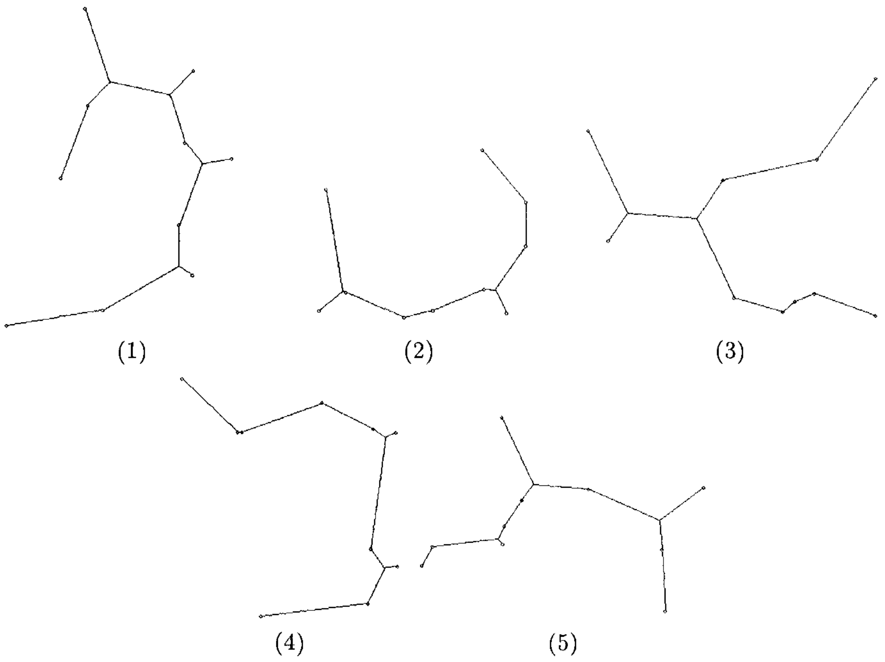


Fig. 6. Exact solutions of five Steiner test problems obtained at the end of the film evolution ($n = 10$)

5 Why the Film Evolution Leads up to the Global Minimum

At the beginning of evolution, when the values of radius R and interior area are extremely high and all corners are dry, the state of the film is unique. At the end of evolution, when the interior area is near to zero, the curve (if $n > 3$) has more than one stable state. Each of these states corresponds to one of the locally minimal trees. How does the evolution of the curve lead up to the global minimum?

Let us analyze the following simple example. Consider the four nodes of a rectangle of width equal to two and height equal to $2\Delta y$. The height is fractionally lesser (in our example $\Delta y = 0.9$). For these fixed points there are two locally minimal trees: one with a vertical double Y configuration, and another with a horizontal double Y configuration. Consequently, because of $\Delta y < 1$ we do have an SMT only in the last case. The length-minimizing curve in this case also has two corresponding stable configurations if radius R is relatively small.

The wet film formed by the film arcs has a rectangular symmetry. Therefore we may investigate only one quadrant of the curve (Fig. 8). We see one of the

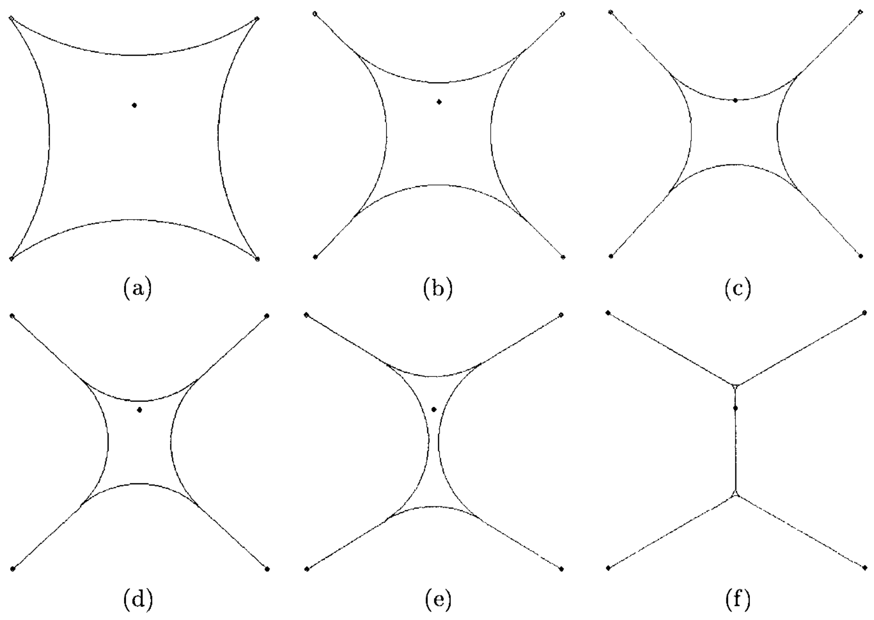


Fig. 7. Six stages of wet film evolution in a specially constructed example

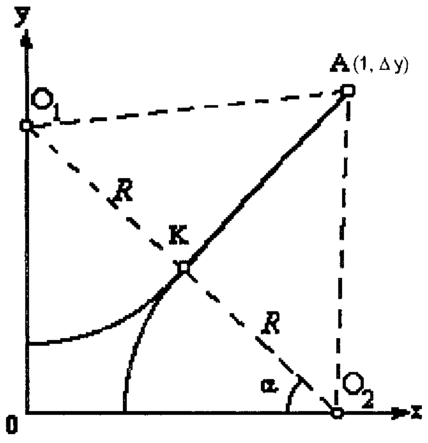


Fig. 8. One quadrant of the film fragment of four dry corners

fixed points A with the coordinates $(1,\ \Delta y)$, its dry corner K, and the centers of two arcs: O_1 and O_2.

The summary length L and the interior area S of one quadrant film for angle α values $30° < \alpha < 60°$ are respectively equal to:

$$L = 2\sqrt{(1 - R\cos\alpha)^2 + (\Delta y - R\sin\alpha)^2} + \pi R/2 \tag{6}$$

and

$$S = R^2(\sin 2\alpha - \pi/4), \tag{7}$$

where $R = \frac{\cos\alpha - \Delta y \sin\alpha}{\cos 2\alpha}$.

The angle α changes during the beginning of the film evolution. If its value achieves an angle of 30°, then the top and bottom arcs touch each other; and as it achieves an angle of 60°, the left and right arcs touch each other. After these events the values of angle α remain constant and equal to 30° for the horizontal double Y configuration or to 60° for the vertical double Y configuration in the next steps of evolution. Formula 6 must be rewritten as

$$L = 1 + \sqrt{3}\Delta y + \left(\pi/2 - \sqrt{3}\right) R \mathrm{for} \alpha = \pi/6 \tag{8}$$

or

$$L = \sqrt{3} + \Delta y + \left(\pi/2 - \sqrt{3}\right) R \mathrm{for} \alpha = \pi/3. \tag{9}$$

The dependency of the summary length L and interior area S during the wet film evolution may be calculated for all stable configurations, using (6–9). The dependency is illustrated in Fig. 8. The bottom graph of Fig. 9 corresponds to the horizontal double Y configuration, which is globally optimal. The summary length L at the end of evolution ($R = 0$) may be calculated by (8) and is equal to 2.5588

The top graph corresponds to the vertical double Y configuration, which is not globally optimal. The summary length L at the end of evolution is greater than in the case of the horizontal double Y configuration. It may be calculated by (9) and is equal to 2.6320.

We can see that the continuous evolution of the wet film, by gradually decreasing the interior area from the values 0.16, has its continuation only for the globally optimal configuration. Locally optimal configuration starts only from the interior area values 0.0038, and is, in some sense, isolated. There is no possibility to obtain them at the end of film evolution.

6 Conclusions

The investigations show that simulation of wet film evolution is an appealing tool for solving the Steiner problem. The mathematical model of wet film evolution consists of the system of equations; some equations must be changed at certain stages of simulation.

Computational experiments and tests exhibit the abilities of a wet film to achieve the global optimal configurations. This phenomenon may be explained for simple cases by means of the dependency of the summary length and interior area.

The approach is an example of the successful use of a natural metaphor to design an optimization algorithm. Evidently the approach can be used in solving other optimization problems related with the shortest length, for example, the Traveling Salesman Problem.

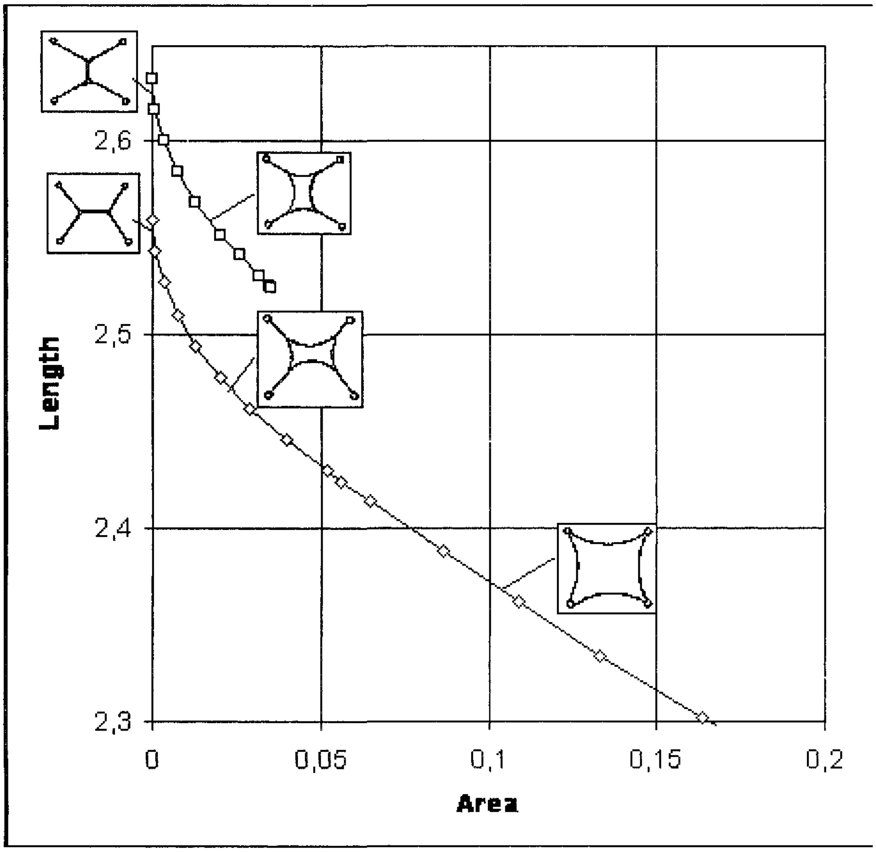


Fig. 9. Dependency of the summary length and interior area during the wet film evolution for all stable configurations

References

[Ada02] Adamatzky, A.: Computing in Nonlinear Media and Automata Collectives. Institute of Physics Publishing, Bristol and Philadelphia (2002)

[Bea92] Beasley, J.E.: A heuristic for Euclidean and rectilinear Steiner problems. European Journal of Operational Research, **58**, 284–292 (1992)

[BF93] Burden, R.L., Faires, J.D.: Numerical Analysis. PWS Publishing Company, Boston (1993)

[BM98] Brakke, K., Morgan, F.: Instability of the wet X soap film. The Journal of Geometric Analysis, **5**, 749–768 (1998)

[Cie98] Cieslik, D.: Steiner Minimal Trees. Kluwer Academic Publishers (1998)

[CR41] Courant, R., Robbins, H.: What is Mathematics? Oxford University Press, London, NY, Toronto (1941)

[GGJ77] Garey, M.R., Graham, R.L., Johnson, D.S.: The complexity of computing Steiner minimal trees. SIAM Journal on Applied Mathematics, **32**, 835–859 (1977)

[GH85] Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. Annals of the History of Computing, **7**, 43–57 (1985)

[GP66] Gilbert, E.N., Pollak, H.O.: Steiner minimal trees. SIAM Journal on Applied Mathematics, **16**, 1–29 (1966)

[HM96] Hass, J., Morgan, F.: Geodesics and soap bubbles in surfaces. Mathematische Zeitschrift, **223**, 185–196 (1996)

[HRW92] Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. North-Holland (1992)

[IT94] Ivanov, A.O., Tuzhilin, A.A.: Minimal Networks – the Steiner Problem and its Generalizations. CRC Press, Boca Raton (1994)

[Jak65] Jakutaviciene, D.A.: The algorithm for solving the problem of minimal linkage of points in a plane. In: Samonastrajivajuscijiesja systiemy. Institute of Automatics and Telemechanics, Nauka, Moscow, 284–286 (1965) (in Russian).

[Mel61] Melzak, Z.A.: On the problem of Steiner. Canadian Mathematical Bulletin, **4**, 143–148 (1961)

[Mor92] Morgan, F.: Minimal surfaces, crystals, shortest networks, and undergraduate research. Mathematical Intelligencer, **14**, 37–44 (1992)

[PS02] Promel, H.J., Steger, A.: The Steiner Tree Problem. Vieweg (2002)

[Sal99] Saltenis, V.: Simulation of wet film evolution and the Euclidean Steiner problem. Informatica, **10**(4), 457–466 (1999)

[Sal00] Saltenis, V.: Investigation of dead-point situations in the simulation of wet film evolution. Informatica, **11**(4), 469–478 (2000)

A Probabilistic Hybrid Differential Evolution Algorithm

Montaz M. Ali

School of Computational and Applied Mathematics, Witwatersrand University, Johannesburg, South Africa mali@cam.wits.ac.za

Summary. In this chapter we propose a hybrid point generation scheme in the differential evolution (DE) algorithm. In particular, we propose a DE algorithm that uses a probabilistic combination of the point generation by the β-distribution and the point generation by mutation. Numerical results suggest that the resulting algorithm is superior to the original version both in terms of the number of function evaluations and cpu times.

Key words: Global optimization, population set, β-distribution, continuous variable, probabilistic adaption.

1 Introduction

The global optimization problem in this chapter follows the form:

$$\text{minimize}\, f(x) \text{ subject to } x \in \Omega \subset R^n, \tag{1}$$

where $f(x) : \Omega \mapsto R$ is a continuous real-valued function. The domain Ω is defined by specifying upper (u^j) and lower (l^j) limits of each dimension j. Therefore, for any $x = (x^1, x^2, \cdots, x^j \cdots, x^n) \in \Omega$, x^j is bounded, i.e. $l^j \leq x^j \leq u^j$. We denote a global optimal solution x^*, with its corresponding global optimal function value $f(x^*)$ or f^* for a short hand notation.

The DE algorithm [SP97] is a population set-based algorithm [AT04] and is purely heuristic. All population set based algorithms use a population set S. The initial set

$$S = \{x_1, x_2, \cdots, x_N\} \tag{2}$$

consists of N uniform random points in Ω. A contraction process is then used to drive these points to the vicinity of a global minimizer. The contraction process involves replacing bad point(s) in S with better point(s), per iteration. DE attempts to replace all points in S by new points at each iteration. It progresses in an epoch or era base. During each epoch, N new function values are evaluated on N trial points. Trial points are generated using mutation and crossover.

In this chapter, we propose a DE algorithm that generates the trial points by (probabilistically) combining the mutation scheme and β-distribution.

This chapter is divided into six sections. In the next section, a brief description of DE is given. In Sect. 3, the motivation for trial point generation using the β-distribution is presented. In Sect. 4, the new algorithms are presented. Section 5 presents numerical results and finally, Sect. 6 contains the concluding remarks.

2 A Brief Description of DE

The DE algorithm attempts to replace each point in S with a new better point. Therefore, in each iteration, N competitions are held to determine the members of S for the next iteration. The i-th ($i = 1, 2, \cdots, N$) competition is held to replace x_i in S. Considering x_i as the target point, a trial point y_i is found from two points (parents), the point x_i, i.e., the target point, and the primary trial point $\hat{x}_i$ (hereafter trial point) determined by the mutation operation. In its mutation phase, DE randomly selects three distinct points $x_{p(1)}, x_{p(2)}$ and $x_{p(3)}$ from the current set S. None of these points should coincide with the current target point x_i. The weighted difference of any two points is then added to the third point which can be mathematically described as:

$$\hat{x}_i = x_{p(1)} + F(x_{p(2)} - x_{p(3)}) \ , \tag{3}$$

where $F > 0$ is a scaling factor, and $x_{p(1)}$ is known as the base vector. If the point $\hat{x}_i \notin \Omega$ then the mutation operation is repeated. We denote the point generation using mutation by M_μ. The secondary trial point y_i is found from its parents x_i and $\hat{x}_i$ using the following crossover rule :

$$y_i^j = \begin{cases} \hat{x}_i^j \ \text{if}\, R^j \le C_R \,\text{or}\, j = I_i \\ x_i^j \ \text{if}\, R^j > C_R \,\text{and}\, j \neq I_i \ , \end{cases} \tag{4}$$

where I_i is an integer randomly chosen with replacement from the set I, i.e., $I_i \in I = \{1, 2, \cdots, n\}$; the superscript j represents the j-th component of respective vectors; $R^j \in (0, 1)$, drawn uniformly for each j ($j = 1, 2, \cdots, n$). The ultimate aim of the crossover rule (4) is to obtain the secondary trial vector y_i with components coming from the components of the target vector x_i and mutated vector $\hat{x}_i$. This is ensured by introducing the parameter C_R and the set I. The targeting process continues until all members of S are considered. After all N secondary trial points y_i have been generated, acceptance is applied. In the acceptance phase, the function value at the secondary trial point, $f(y_i)$, is compared to $f(x_i)$, the value at the target point. If $f(y_i) < f(x_i)$ then y_i replaces x_i in S, otherwise, S retains the original x_i. Reproduction (mutation and crossover) and acceptance continue until some stopping conditions are met.

It can be seen from (3) that mutation (M_μ) is the main point generation mechanism of DE. This operation calculates the coordinates of new points. The crossover operation (4) chooses the coordinates of a (secondary) trial point from the known coordinates of two points using a distribution controlled by C_R. We intend to modify the scheme M_μ. To this end, we use the β-distribution.

3 The Self-Adjusting β-Density

The β-distribution $[l, u]$ has probability density given by

$$f_Z(x) = \frac{\Gamma(a+b)(x-l)^{(a-1)}(u-x)^{(b-1)}}{\Gamma(a)\Gamma(b)(u-l)^{(a+b-1)}}, \ a, b > 0, \tag{5}$$

where a and b are the parameters and Γ is the gamma function. The case where $l = 0$ and $u = 1$ is called the *standard β distribution.* The mean, variance, and the coefficient of skewness of the *standard β distribution* are respectively given by

$$\mu = a/(a+b), \ s^2 = ab/(a+b)^2(a+b+1), \ sk = 2\frac{(b-a)}{(a+b+2)}\sqrt{\frac{a+b+1}{ab}}. \tag{6}$$

Clearly, the values of a and b determine the shape of the density function. If l and u are known then the parameters a and b are estimated as follows:

$$a = \mu\left(\frac{\mu(1-\mu)}{s^2} - 1\right), \ b = (1-\mu)\left(\frac{\mu(1-\mu)}{s^2} - 1\right), \tag{7}$$

where μ is the sample mean and s^2 is the sample variance. If l and u are not 0 and 1, respectively, then μ is replaced with $\frac{\mu-l}{u-l}$ and s^2 with $\frac{s^2}{(u-l)^2}$ in (7).

We now show how the β-distribution can be used as mutation in DE instead of M_μ. We call this scheme the first β-distribution scheme. For each target point x_i the mutation, M_μ, of DE calculates the (mutant) trial point $\hat{x}_i$, $i = 1, 2, \cdots, N$. We would like to generate the trial vector $\hat{x}_i$, coordinate by coordinate, corresponding to the target x_i, using the β-distribution instead of M_μ. Let the j-th coordinate be our coordinate of interest. Since we are interested in the j-th component $\hat{x}_i^j \in [l^j, u^j]$ of a trial point $\hat{x}_i$, we let a^j and b^j represent the parameters of the β-distribution in the j-th coordinate, $j = 1, 2, \cdots n$. We also denote the mean by $\underline{x}^j$ and the standard deviation by s^j of the β-distribution in $[l^j, u^j]$. It can be seen from (7) that the parameters a^j and b^j can be calculated from the given mean and standard deviation. If we denote a randomly generated value of this coordinate by $\hat{x}_i^j$ then $\hat{x}_i^j$ must lie between the upper and lower limits, u^j and l^j, respectively of the j-th coordinate, $\hat{x}_i^j = k^j z^j + l^j$, where $k^j = u^j - l^j$ and z^j is a random realization from a $\beta(a^j, b^j)$ distribution. We define $\theta^j = (\underline{x}^j - l^j)/k^j$ and

$A^j = \left[(k^j)^2\theta^j(1-\theta^j)/(s^j)^2\right] - 1$, where θ^j and s^j/k^j are the given mean and standard deviation of the β random variable Z^j in $[0,1]$ with realization z^j respectively. One can use these to write $a^j = A^j\theta^j$ and $b^j = A^j(1-\theta^j)$ as in (7).

The above procedure of generating the trial point $\hat{x}_i$ in DE has been suggested in a recent study [AF06]. However, an empirical based formula is used for the standard deviation of the β-distribution. It has been shown in [AF06] that at the later stages of DE the β-distribution performs very poorly. After reaching the vicinity of the global minimum the β-distribution takes many iterations to satisfy the stopping condition. This is due to the average improvement of the function values in S being very small although the β-distribution replaces the target points, per iteration, significantly.

In our implementation, we replace the empirical based formula with the following sample based standard deviation in the first β-distribution scheme and combine it with a second β-distribution scheme. An estimate of the standard deviation s^j would be the sample standard deviation, σ^j, of the j-th components of the N points in the current set S, giving

$$\sigma = (\sigma^1, \cdots, \sigma^j, \cdots, \sigma^n). \tag{8}$$

Clearly, $||\sigma|| \to 0$ as the iteration counter $k \to \infty$.

In our second β-distribution scheme we calculate the parameters of the β distribution using the coefficient of skewness and mean. The values of a^j and b^j can be calculated for known skewness and mean. One can then generate the j-th component, $\hat{x}_i^j$, of the trial vector $\hat{x}_i$, $i = 1, 2, \cdots, N$. Let sk^j be the coefficient of skewness of the β distribution; then both a^j and b^j satisfy the following equations:

$$\theta^j = a^j/(a^j + b^j), \tag{9}$$

$$sk^j = 2\frac{(b^j - a^j)}{(a^j + b^j + 2)}\sqrt{\frac{a^j + b^j + 1}{a^j b^j}}, \tag{10}$$

where sk^j can take both positive and negative values. Notice that sk^j is zero for $a^j = b^j$ when the β distribution is symmetrical to its mean. Clearly, the solution of (9) and (10) is not trivial due to the non-linearities in (10). From (9) we can write

$$b^j = a^j(1-\theta^j)/\theta^j. \tag{11}$$

Substituting b^j in (10) we get the following equation in a^j:

$$F(a^j) = \frac{2(1-2\theta^j)}{(1-\theta^j)^{\frac{1}{2}}} \frac{(a^j + \theta^j)^{\frac{1}{2}}}{(a^j + 2\theta^j)} - sk^j = 0. \tag{12}$$

The value of a^j can now be found by solving (12) using Newton method for known sk^j and θ^j, and the value of b^j can be found from (11) for any given a^j. We only use an approximate solution to (12). For example, we stop the

Newton's method if the absolute difference between any two consecutive solutions is less than 0.01. However, if this condition does not satisfy in 10 iterations of the Newton method we take the solution of the 10-th iteration. Our experience of the numerical solution of (12) using Newton's method suggested occasional numerical instabilities. When this occurred we obtained a^j randomly in $[1, 5000]$.

Notice that the mean can be embedded in the second beta random scheme in $[l^j, u^j]$ but not the the standard deviation. As a result, at the early stages of DE when the points in S are scattered in Ω the trial point generated by this β-distribution will be local to the given mean, due to the standard deviation being too low. On the other hand, when the points in S form a cluster around the global minimum the standard deviation determined by the second scheme will be too high as compared to the sample standard deviation. To overcome the above difficulties as well as to guarantee the use of M_μ at the later stages of DE the following switching mechanism is used.

If the standard deviation, s^j, of the second scheme is less than or equal to a fraction of the sample standard deviation, σ^j, the first scheme is used. In particular, if

$$s^j = \left(\frac{a^j b^j}{(a^j + b^j)^2(a^j + b^j + 1)}\right)^{\frac{1}{2}} \leq \epsilon\sigma^j \tag{13}$$

then we use the first scheme otherwise we use the second. We denote this combined scheme by M_β. The combined scheme M_β will overcome the difficulty when the standard deviation of the second scheme is too low compared to the range of the distribution. When the points in S are dense then (13) will not be satisfied which means that the second scheme will be used by M_β. However, trial points generated by the second scheme, when the points in S form a cluster around the global minimum, will be unsuccessful. This knowledge is then used to switch from M_β to M_μ at the later stages of the algorithm. A probabilistic scheme is suggested for this purpose.

Besides the standard deviation, mean of the β-distribution in both the schemes has to be provided. There is no specific way one can choose the mean of the β distribution in $[l^j, u^j]$. We choose the mean to reflect the goodness of the points in S. In particular, we choose the mean $\underline{x} = (\underline{x}^1, \cdots, \underline{x}^j, \cdots, \underline{x}^n)$ as the best of the two uniform random points, y_1 and y_2, in S. Therefore, the mean $\underline{x}$ is such that

$$\underline{x} = \arg\min_{y_1, y_2} \{f(y_1), f(y_2)\}. \tag{14}$$

The calculated parameters a^j and b^j using both β-distribution schemes could take on negative values, which are not permitted, or values between 0 and 1, for which the β-distribution is U-shaped. We therefore restrict them to values in the range $[1, \infty)$, replacing them by 1 if they fall outside it. The values of a^j and b^j, therefore, lie in $[1, L]$, where L is a large number and $L < \infty$. Notice that in the limiting case $a^j = b^j = 1$, the β-distribution is a uniform distribution. In our implementation, however, if the calculated values of a^j and b^j were less than or equal to 1 then we used

$$\hat{x}_i^j = 0.5(y_1^j + y_2^j), \tag{15}$$

to obtain the corresponding j-th component of $\hat{x}_i$. The points y_1 and y_2 are random points in S used in calculating the mean $\underline{x}$ in (14). We use the algorithm of Cheng [Che78] for generating the β-variates. The second β-distribution scheme requires the coefficient of skewness and the mean for the j-th component of the trial vector $\hat{x}_i$. Since we restrict the parameters a^j and b^j in $[1, L]$ the skewness lies in $[-2, 2]$. Therefore, we select $sk^j \in [-2, 2]$.

In the next section, we suggest an algorithm that probabilistically combines the schemes M_μ and M_β. The algorithm generates trial points using some probability distribution on $\{M_\mu, M_\beta\}$. We refer to this version of DE as the probabilistic hybrid DE algorithm (PHDE). The main feature of PHDE is that it probabilistically adapts a trial points generation scheme that solves a given problem in a robust manner. Full details of the new algorithm are given below.

4 The Probabilistic Hybrid DE Algorithm (PHDE)

The PHDE algorithm generates trial points $\hat{x}_i$ $(i = 1, 2, \cdots, N)$ using some probability distribution over the set $\{M_\mu, M_\beta\}$. That is, trial points are either generated using M_μ or M_β at each iteration. Initially equal probabilities (0.5) are assigned to both scheme M_μ and M_β and these probabilities are updated according to some rules based on reward (for being successful) and penalty (for being unsuccessful). This probabilistic adaptation in the algorithm guides PHDE in deciding on which mutation scheme to use most in generating points for any given problem. This allows the algorithm to bias the the rule that solves a given problem in a most efficient and robust manner. The probabilistic scheme penalizes (rewards) a mutation scheme for not making (making) good progress. We combine the scheme M_μ with M_β so that mutated points are either generated with some probability α_k using M_μ or with probability $\gamma_k = 1 - \alpha_k$ using M_β. A scheme is selected with some probability (e.g. M_μ is first selected with probability 0.5) to create mutated points, and this probability is updated after each iteration. If $\hat{x}_i$ are generated, say using M_μ, and a higher number of these points are found to be producing much better secondary trial points compared to the current N points in S then the probability for using the scheme M_μ is increased (reward). We use the following scheme for increasing the probability for M_μ:

$$\alpha_k = \alpha_{k-1} + \frac{1}{2}\alpha_{k-1}(1 - \alpha_{k-1}), \tag{16}$$

and γ_k is obtained using $\gamma_k = 1 - \alpha_k$. If, however, the secondary trial points are not better in comparison to the current N points in S then the probability of M_μ is decreased (penalty) using

$$\alpha_k = \alpha_{k-1} - \frac{1}{2}\alpha_{k-1}(1 - \alpha_{k-1}). \tag{17}$$

The motivation for the use formulae (16) and (17) in probabilistic adaptation can be found in [NPL90] where they have been used in the context of combinatorial optimization using learning automata.

To determine whether to reward or to penalize a mutation scheme we use the following empirical scheme. We count the number of replacements (nr) in S in an iteration. If nr is greater than or equal to the nearest integer to $0.6N$ then we reward the mutation scheme used. If it is less than or equal to $0.3N$ than we penalize the scheme. If, however, nr falls between $0.3N$ to $0.6N$ then we neither reward or penalize the scheme. This adaptive process tends to let the algorithm decide for itself which scheme to use most in generating trial points for any given problem so that it solves the problem in a much more efficient way. This is done by increasing the value of α_k whenever M_μ gives more favourable points, thus reducing the probability of using M_β. On the other hand, if M_β produces a high nr then α_k is reduced. We force $\alpha_k, k \geq 1$, to lie in $[0.05, 0.95]$. For example, if α_k goes below 0.05, we set $\alpha_k = 0.05$ by clipping. This is done in order to avoid the algorithm switching entirely to one scheme. Thus α_k and γ_k lie in $(0, 1)$ to allow the algorithm to be able to switch from one scheme to the other. The algorithm for PHDE is described below.

The PHDE Algorithm

Step 1 **Determine the initial set**

$$S = \{x_1, x_2, \cdots, x_N\}$$

where the points x_i, $i = 1, 2, \cdots, N$, are sampled randomly in Ω; evaluate $f(x)$ at each x_i. Take $N \gg n$, n being the dimension of the function $f(x)$. Set iteration counter $k = 0$. Set $\alpha_k = 0.5$.

Step 2 **Determine the best and the worst point in** S: Determine the points $x_{\max}, x_{\min}$ and their function values $f_{\max}, f_{\min}$ such that

$$f_{\max} = \max_{x \in S} f(x) \quad \text{and} \quad f_{\min} = \min_{x \in S} f(x).$$

If the stopping condition is satisfied, then stop.

Step 3 **Generate points to replace points in** S. If $\omega \leq \alpha_k$, where ω is a random number in [0,1], then go to Step 3a else go to Step 3b.

Step 3a Mutation rule M_μ: For each $x_i \in S$ $(i = 1, 2, \cdots, N)$ determine $\hat{x}_i$ using (3). Go to Step 3c.

Step 3b Mutation using M_β: For each $x_i \in S$ $(i = 1, 2, \cdots, N)$ determine $\hat{x}_i$ using M_β. Randomly select two points from S (which may include x_i, the target point) and find the best point, say $\underline{x}$ of the two as in (14). Obtain $\hat{x}_i^j$ $(j = 1, 2, \cdots, n)$ using β-distribution with mean $\underline{x}^j$ and skewness sk^j. Go to Step 3c.

Step 3c Crossover : Calculate the secondary trial vector y_i corresponding to the target x_i from x_i and $\hat{x}_i$ using the crossover rule (4).

Step 4 **Replace points in** S. Set nr=0. Select each trial vector y_i for the $(k+1)$-th iteration using the acceptance criterion : If $f(y_i) < f(x_i)$ then set nr=nr+1 and replace $x_i \in S$ with y_i otherwise retain x_i. If this Step is reached from Step 3a then go to Step 4a else go to Step 4b.

Step 4a If nr$\geq 0.6N$ then $\alpha_k = \alpha_{k-1} + \frac{1}{2}\alpha_{k-1}(1-\alpha_{k-1})$. If nr$\leq 0.3N$ then $\alpha_k = \alpha_{k-1} - \frac{1}{2}\alpha_{k-1}(1-\alpha_{k-1})$. Set $k = k+1$ and go to Step 2.

Step 4b If nr$\geq 0.6N$ then $\alpha_k = \alpha_{k-1} - \frac{1}{2}\alpha_{k-1}(1-\alpha_{k-1})$. If nr$\leq 0.3N$ then $\alpha_k = \alpha_{k-1} + \frac{1}{2}\alpha_{k-1}(1-\alpha_{k-1})$. Set $k = k+1$ and go to Step 2.

Remarks

1. It can be seen from the above Algorithm that when $\alpha_k = 1$ we have the DE algorithm. However, when $\alpha_k = 0$ we have a DE algorithm that uses M_β as mutation instead of M_μ.
2. The skewness in Step 3b is assigned using the following rule. At the first call of M_β, the second β-distribution scheme, we select sk^j, for each j, uniformly in $[-2,2]$. In subsequent calls, we assign values of sk^j depending upon the success of M_β in the previous call. For example, if the number of target points replaced by M_β is less than 30% of the population size N then we assigned sk^j uniformly in $\{-1,1\}$ in order to generate trial points locally, at the current iteration. On the other hand, we assigned sk^j uniformly in $\{-2,2\}$ for exploration.

5 Numerical Results

In this section we judge the performance of the new algorithm using a collection of 27 test problems. These problems range from 3 to 20 in dimension and have a variety of inherent difficulty. All the problems have continuous variables. A detailed description of each test problem (P) in the collection can be found in [AKZ05].

We compare the results obtained by the new algorithms with those of the DE algorithm. The algorithms were run 100 times on each of the 27 test problems to Determine the success rate (sr) (or percentages of success) of each algorithm. There were 2700 runs in total. We calculated the average number of function evaluations (fe) and cpu times (cpu) for those runs for which the global minima were found. We used

$$\left|f_{\max} - f_{\min}\right| \leq 10^{-4} \tag{18}$$

as our stopping condition. We used sr, fe and cpu as the criteria for comparison. A solution to the problem need not be the global minimum f^* exactly, but may be any value less than f^*_ε,

$$f^*_\varepsilon = f^* + 9 \times 10^{-4}. \tag{19}$$

A success was counted when the final solution satisfies (19).

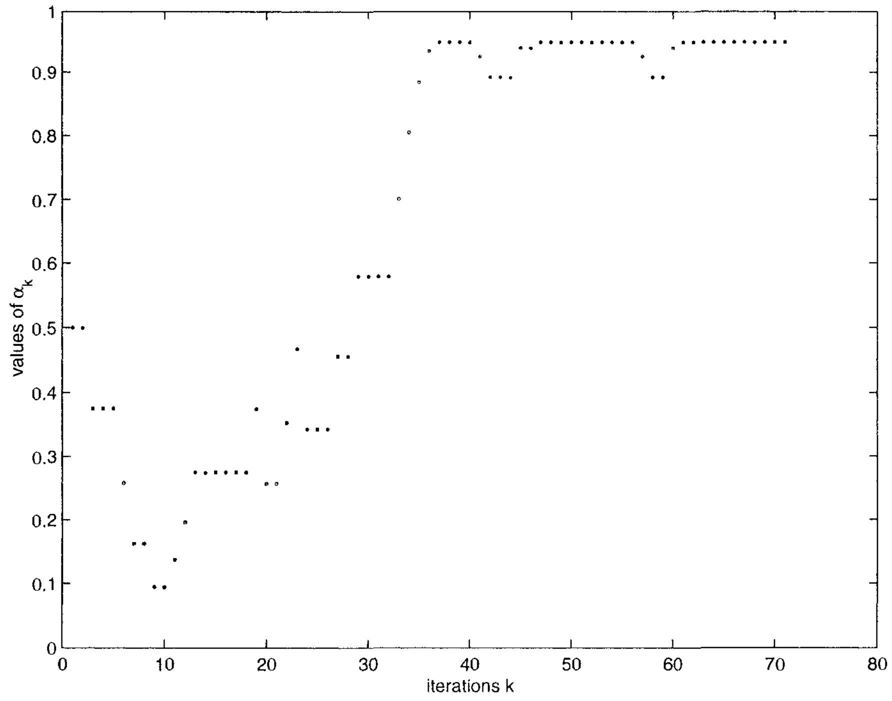


Fig. 1. Adaptation of PHDE on H6

5.1 Parameter Value Selection

We first discuss the parameters that are common to both algorithms. For example, the size N of the population set S and the scaling parameter F in (3) and the parameter C_R in its (4). We took the value of N to be $10n$ where n is the dimension of the problem. These are heuristic choices. For example the value of N can always be increased for obtaining the global minimum with higher probability. However, the higher the value of N, the higher the number of fe is. We have used $C_R = F = 0.5$, see reference [Kae05] for this choice. A parameter associate with PHDE is ϵ in (13). The value $\epsilon = 0.25$ was found to be suitable after numerical testings. We do not claim these values to be the optimal for any given problem in general but they are good values to choose.

5.2 Comparison

We present the results obtained by the algorithms in in Table 1. In Table 1, the first column contains the problem name [AKZ05] with its dimension given in brackets. The last row contains the total results. A comparison using total results shows that DE is much inferior to PHDE in terms of fe. For example, PHDE is superior to DE by about 64% in terms of fe. Although, the total

Table 1. Comparison of PHDE and DE using 27 test problems

	PHDE			DE		
$P(n)$	fe	sr	cpu	fe	sr	cpu
ACK(10)	37796	100	1.52	206978	100	6.72
EM(10)	248600	32	13.58	538741	96	14.86
EXP(10)	7332	100	0.36	16366	100	0.03
GW(10)	40322	100	1.48	264494	100	4.77
GRP(3)	4119	100	0.51	4004	100	0.24
H3(3)	1185	100	0.04	1474	100	0.01
H6(6)	5771	100	0.19	9180	100	0.13
HV(3)	4735	100	0.08	4602	100	0.03
KL(4)	1720	100	0.06	2702	100	0.02
LM1(3)	1384	100	0.04	1860	100	0.02
LM2(10)	9823	100	0.46	21258	100	0.31
MR(3)	2095	100	0.05	3857	100	0.02
MCP(4)	1555	100	0.05	2654	100	0.03
ML(5)	10466	80	0.30	24883	100	0.36
NF2(4)	72581	100	2.05	149692	96	1.17
NF3(10)	110318	100	8.24	157534	100	1.79
PP(10)	12292	100	0.53	25382	100	0.49
PQ(4)	6348	100	0.12	7430	100	0.07
RG(10)	42074	100	3.38	156654	100	3.15
SAL(5)	31214	3	0.24	0	0	0.00
SWF(10)	22800	100	1.12	50090	100	1.66
S5(4)	3775	69	0.08	7431	100	0.06
S7(4)	3726	86	0.08	6423	100	0.05
S10(4)	3625	90	0.08	6373	100	0.06
FX(5)	8115	12	0.18	17906	9	0.26
SIN(20)	40482	100	5.35	155916	100	5.88
WP(4)	21404	99	0.47	18464	100	0.13
Total	656370	2371	40.64	1862348	2501	42.32

results include the fe for SAL where DE was unsuccessful. In terms of cpu PHDE is also slightly superior to DE. On the other hand, DE is superior to PHDE by 130 successes. Overall comparisons show that PHDE is significantly superior to DE in terms of fe although it is slightly outperformed by DE in terms of sr.

5.3 Probabilistic Adaptation

We now show how the probabilistic adaptation takes place within PHDE. We present two figures to illustrate the adaptations. The figures have been plotted using the number of iterations k on the horizontal axis and values of α_k on the vertical axis. For a particular problem, we observed slight variations in plots from run to run. However, the general trend is the same for all successful runs. Therefore, we present each figure from a single run. Figures 1 and 2 respectively to the functions Hartman 6 (H6) and Shekel 5 (S5). The PHDE algorithm uses M_μ for $\alpha_k = 1$ and M_β for $\alpha_k = 0$ and a mixture of the two

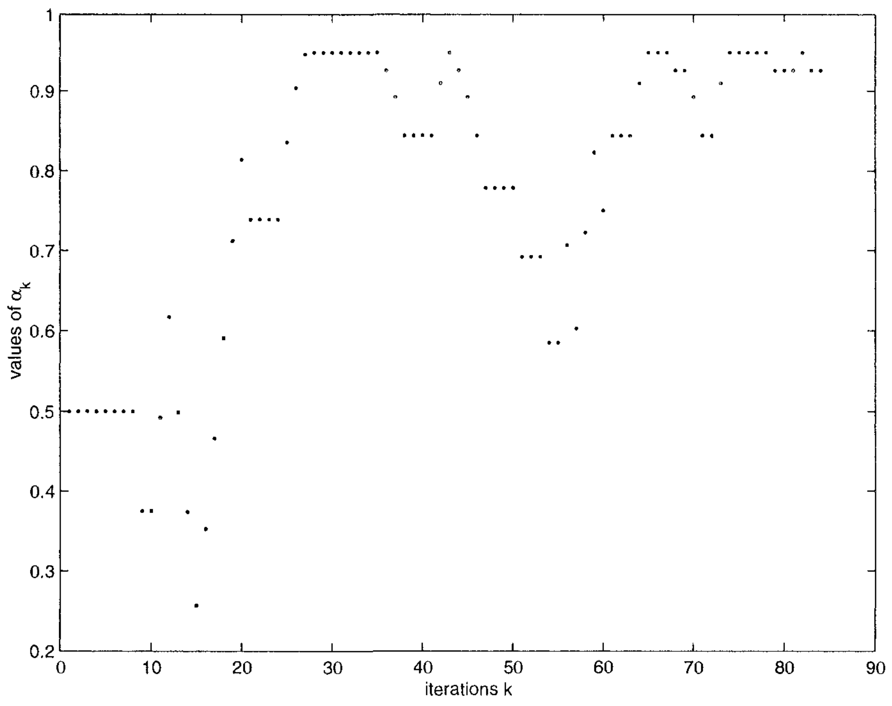


Fig. 2. Adaptation of PHDE on S5

for any $\alpha_k \in (0, 1)$. For the Hartman 6 problem (Fig. 1), PHDE tends to use more of M_β than M_μ for the first 30 iterations before almost switching completely to M_μ after about 35 iterations to solve the problem. For the Shekel 5 problem (Fig. 2), for the first 10 iterations PHDE uses both the schemes evenly. After about 15 iterations PHDE switches to M_β and after about another 10 iterations it uses about an even mixture of the two schemes before switching to a majority of M_μ. Figures 1 and 2 clearly demonstrate that at later stages the scheme M_μ was used, as intended.

6 Conclusion

We have developed and tested a version of the DE algorithm on a set of problems. Numerical results have shown that the new version is considerably better than their original counterpart in terms of the number of function evaluations and cpu times. The new version, PHDE, has more flexibility than the original DE. In effect, we have generalized the DE in that it is a special case of PHDE. We have also shown how the probabilistic adaptation in PHDE guides the algorithm. The new methods are slightly less efficient in terms sr.

However, this is compensated for by the large decrease in number of function evaluations.

References

[AF06] Ali, M.M., Fatti, L.P.: A differential free point generation scheme in the differential evolution algorithm. Journal of Global Optimization, in press

[AKZ05] Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization, **31**, 635–672 (2005)

[AT04] Ali, M.M., Törn, A.: Population set based global optimization algorithms: some modifications and numerical studies. Computers and Operations Research, **31**, 1703–1725 (2004)

[Che78] Cheng, R.C.H.: Generating beta variates with non-integral shape parameters. Communications of the ACM, **21**, 317–322 (1978)

[Kae05] Kaelo, P.: Some population set based methods for unconstrained and constrained global optimization. PhD thesis, Witwatersrand University, Johannesburg (2005)

[NPL90] Najim, K., Pibouleau, L., Le Lann, M.V.: Optimization technique based on learning automata. Journal of Optimization Theory and Applications, **64**, 331–347 (1990)

[SP97] Storn, R., Price, K.: Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, **11**, 341–359 (1997)

Nonadaptive Univariate Optimization for Observations with Noise

James M. Calvin

New Jersey Institute of Technology, Newark, NJ USA calvin@njit.edu

1 Introduction

It is much more difficult to approximate the minimum of a function using noise-corrupted function evaluations than when the function can be evaluated precisely. This chapter is concerned with the question of exactly how much harder it is in a particular setting; namely, on average when the objective function is a Wiener process, the noise is independent Gaussian, and nonadaptive algorithms are considered.

Statistical models of objective functions have long been used in the study of global optimization algorithms; see [TŽ89], [Žil85]. These models are well-suited for inclusion of noise in the function evaluations. Several papers have considered this problem, including [Žil80].

The most tractable model for a random objective function is the Wiener process; [Kus62], [Žil81]. For optimization without noise, K. Ritter [Rit90] showed that the best nonadaptive algorithms have error $\Omega(n^{-1/2})$ after n function evaluations. In the noisy case considered in this chapter, no method (adaptive or nonadaptive) can achieve a convergence rate better than $\Theta(n^{-1/2})$. Even if the location of the minimizer were known to the searcher beforehand, and all observations were made at that point, the error would be of order $\Theta(n^{-1/2})$.

In [CŽ05] it was shown that with equispaced observations the error is of order $n^{-1/4}$. This chapter is an extension of [CŽ05], considering another nonadaptive algorithm and giving tighter bounds.

In the next section we describe the problem. In Sect. 3 we summarize some background on the conditional distribution. In Sect. 4 we introduce the algorithms and give the main results.

2 Problem Description

We consider the problem of approximating the global minimum of a function $f : [0,1] \to \mathbf{R}$ using a fixed number of evaluations of the function corrupted by random error. We assume that f is an element of the class C_0 of continuous functions on $[0,1]$, with $f(0) = 0$. We will assume a Gaussian model both for the error and for the function f. Specifically, we suppose that if we choose to evaluate at $t \in [0,1]$, then we observe $f(t) + \xi(t)$, where $\{\xi(t) : t \in [0,1]\}$ are independent normal random variables with mean 0 and variance σ^2. We take f to be a sample path of a Wiener process. Let P denote the Wiener measure on $C([0,1])$. This is the Gaussian measure with mean 0 and covariance

$$\int_{C_0} f(s)f(t)\,dP(f) = \min\{s,t\}.$$

With probability one, the Wiener process paths are nowhere differentiable, and the set of local minimizers is dense in $[0,1]$.

A nonadaptive algorithm using $n \geq 1$ evaluations specifies a set of n points $\{t_i : 1 \leq i \leq n\}$ which we label in increasing order, so $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n \leq 1$. Let $y_i = f(t_i) + \xi(t_i)$, $1 \leq i \leq n$ be the observed values. Based on the n observations, the algorithm produces an estimate $t_n^*(y)$ of the global minimizer t^*. We consider the error

$$e_n = \left(\int_{C_0}\int_{\mathbf{R}^n} \left(f(t_n^*(y)) - f(t^*)\right)^2 \, d\mu(y|f)\,dP(f)\right)^{1/2}$$

where $\mu(\cdot|f)$ is the conditional distribution of the observations $\{y_1,\dots,y_n\}$ given f.

It is known that the optimal choice of t_n^* is the minimum of the conditional mean; see [Was92]. Since the conditional mean is given by linear interpolation of values at the knots, $t_n^* = t_i$ for some i.

In the case of no noise ($\sigma = 0$) it is known that the best nonadaptive algorithms have error $e_n = \Theta(n^{-1/2})$; see [Rit90]. With $\sigma > 0$, it is proved in [CŽ05] that for the algorithm that chooses equispaced points ($t_i = i/n$), $e_n = O(n^{-1/4})$.

3 The Conditional Distribution

Let us recall some facts about the conditional distribution; see [Pla96], [CŽ05]. Define sequences $\{a_i\}$, $\{c_i\}$, $\{d_i\}$, $0 \leq i \leq n$, and $\{b_i\}$, $1 \leq i \leq n$ as follows:

$$\begin{aligned} a_0 &= c_0 = 0 \\ c_i &= \frac{\sigma^2(t_i - a_{i-1})}{\sigma^2 + (t_i - a_{i-1})}, \\ a_i &= t_i - c_i \end{aligned}$$

for $1 \le i \le n$, and

$$\begin{aligned} d_n &= c_n \\ b_i &= a_{i-1} + \frac{(t_i - a_{i-1})^2}{(t_i - a_{i-1}) - d_i} \\ d_{i-1} &= \frac{(t_{i-1} - a_{i-1})(b_i - t_{i-1})}{b_i - a_{i-1}} \end{aligned}$$

for $1 \le i \le n$.

Since both function and noise are Gaussian, so is the conditional measure. Let R_n be the covariance function of the conditional distribution given $\{f(t_i) + \xi(t_i) : 1 \le i \le n\}$. Then for $t_{i-1} \le s \le t_i$,

$$R_n(s) = \frac{(s - a_{i-1})(b_i - s)}{b_i - a_{i-1}}.$$

The conditional mean m_n is constructed as follows.

Set $\Delta_i = t_i - t_{i-1}$ and define the sequences

$$Q_n = 1, Q_{n-1} = Q_n + r_{n-1}, \ldots, Q_i = Q_{i+1} + r_i, \ldots, Q_1 = Q_2 + r_1,$$

where

$$r_n = 1, r_{n-1} = r_n + \frac{\Delta_n Q_n}{\sigma^2}, \ldots, r_i = r_{i+1} + \frac{\Delta_{i+1} Q_{i+1}}{\sigma^2}, \ldots, r_1 = r_2 + \frac{\Delta_2 Q_2}{\sigma^2}.$$

Set

$$u_1 = \frac{t_1}{\sigma^2(\sigma^2 r_1 + t_1 Q_1)}, \; u_i = \frac{t_i - \sigma^2 r_i \sum_{j=1}^{i-1} t_j u_j}{\sigma^2(\sigma^2 r_i + t_i Q_i)}, \; i = 2, \ldots, n. \tag{1}$$

The conditional mean at the knots $\{t_i\}$ is given by

$$m_n(t_i|\mathcal{F}_n) = \sigma^2 \left(\sum_{j=1}^{i-1} y_j u_j \right) r_i + \sigma^2 \left(\sum_{j=i}^{n} y_j r_j \right) u_i = \sum_{j=1}^{n} w_{ij} y_j, \tag{2}$$

and elsewhere it is given by linear interpolation of the values at the knots.

If the knots are somewhat uniformly spread out (locally), then we have a more convenient approximation for the mean and covariance.

Lemma 1. *If $t_{i_n} \to t \in (0,1)$ and $n\Delta_{i_n} \to \alpha > 0$, then*

$$m_n(t_{i_n}) \sim \frac{1}{2\sqrt{n/\alpha}} \sum_{j=1}^{n} \exp\left(-\sqrt{n/\alpha}\,|t_j - t_{i_n}|\right) y_j$$

and

$$R_n(t_{i_n}, t_{i_n}) \sim \frac{\sigma^2}{2\sqrt{n/\alpha}}.$$

As shown in Fig. 1, the function observations are very scattered compared with the mean, which is much smoother than the actual sample function. It is clear from the figure that the noisy observations lead to much greater error in approximating the minimum compared to the case with no noise.

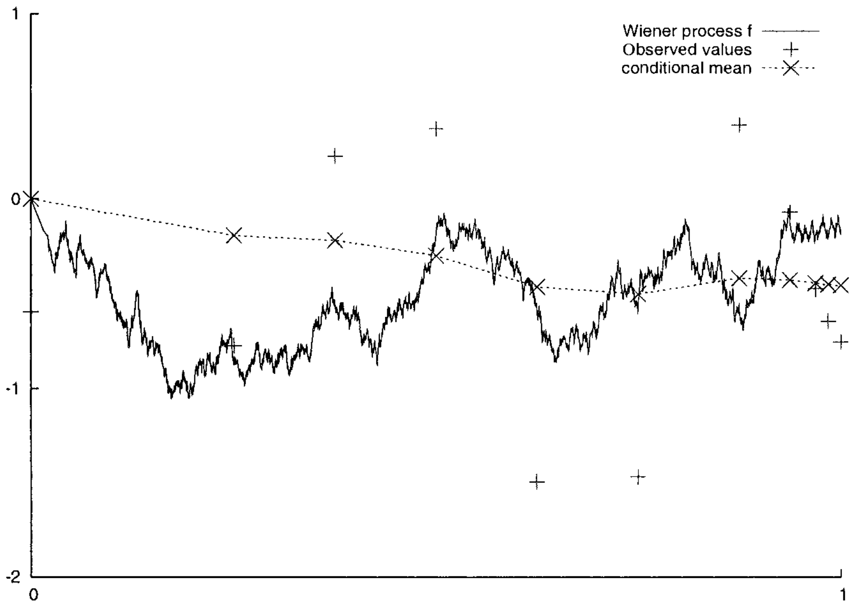


Fig. 1. Sample function and conditional mean

4 Regular Sequences

A common method to construct a sequence of nonadaptive algorithms is to base the algorithms on a regular sequence based on a probability distribution.

Let G be an increasing continuous function on $[0,1]$, with $G(0)=0$ and $G(1)=1$. For each n, we define a regular sequence $\{t_i : 1 \leq i \leq n\}$ by

$$\frac{i}{n} = G(t_i).$$

Let $H = G^{-1}$, so that $t_i = H(i/n)$. Then

$$n\left(t_i - t_{i-1}\right) = \frac{H\left(\frac{i}{n}\right) - H\left(\frac{i-1}{n}\right)}{\frac{1}{n}} \approx H'(i/n).$$

We define an algorithm associated with G as follows. For $t \in (0,1)$ define

$$m_n(t) = \frac{\sqrt{H'(t)}}{2\sqrt{n}} \sum_{j=1}^{n} \exp\left(-\sqrt{n/H'(t)}\,|t_j - t|\right) y_j \triangleq \sum_{j=1}^{n} w_j(t) y_j.$$

Let t_n^* be the minimizer of m_n.

We begin with the case of a uniform grid, $t_i = i/n$. This corresponds to $G_0(x) = x$, $0 \leq x \leq 1$.

Theorem 1. *For the uniform grid,*

$$\frac{n^{1/4}}{\sigma^{1/2}} e_n \to \gamma,$$

where

$$1/2 \leq \gamma \leq 3/2.$$

Numerical experiments suggest $\gamma \approx 1.09$.

We next consider the sequence derived from the density

$$g_1(x) = (x(1-x))^{-2/5} B(3/5, 3/5)^{-1},$$

where B denotes the beta function

$$B(x,y) = \int_{t=0}^{1} t^{x-1}(1-t)^{y-1}\, dt.$$

That is,

$$G_1(x) = \int_{y=0}^{x} g_1(y)\, dy.$$

Theorem 2. *For the regular sequence based on G_1,*

$$\frac{n^{1/4}}{\sigma^{1/2}} e_n \to \frac{B(3/5,3/5)^{5/4}}{\pi} \approx 0.958\gamma.$$

Thus G_1 gives a small improvement compared with the uniform grid based on G_0. The density g_1 is related to the density of the minimizer t^*, which has the arcsine density

$$h(t) = \frac{1}{\pi\sqrt{t(1-t)}};$$

see Fig. 2.

We now outline the proof of the theorems.

Let t^* denote the (first) point of global minimum. For $-t^*\sqrt{n}/\sigma \leq t \leq (1-t^*)\sqrt{n}/\sigma$, define the scaled processes

$$\Lambda_n(t) = \frac{n^{1/4}}{\sqrt{\sigma}} \sum_{j=1}^{n} w_j \left(t^* + t\sigma n^{-1/2}\right) (f(t_j) - f(t^*)) \tag{3}$$

and

$$\Gamma_n(t) = \frac{n^{1/4}}{\sqrt{\sigma}} \sum_{j=1}^{n} w_j \left(t^* + t\sigma n^{-1/2}\right) \xi(t_j). \tag{4}$$

Theorem 3. *As $n \to \infty$,*

$$\Lambda_n \xrightarrow{\mathcal{D}} \Lambda, \qquad \Gamma_n \xrightarrow{\mathcal{D}} \Gamma, \tag{5}$$

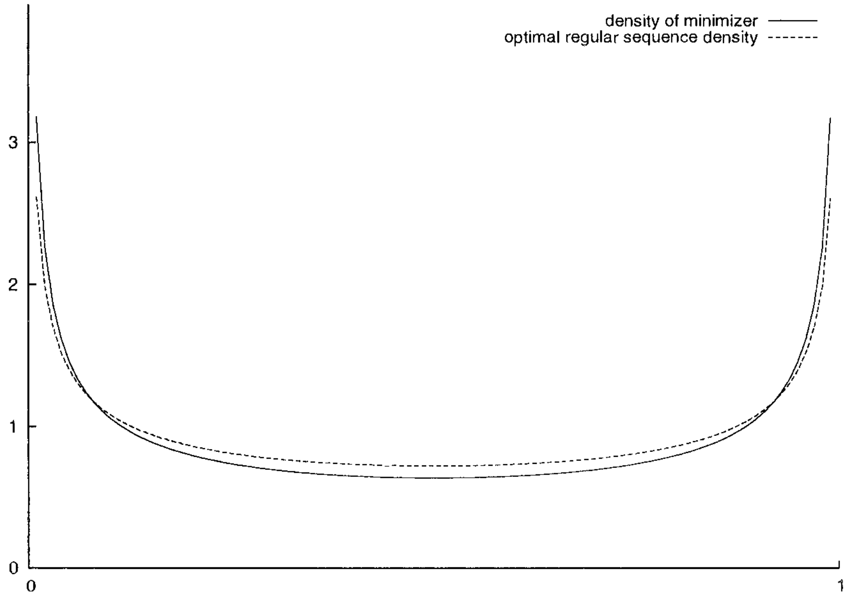


Fig. 2. Comparison of density of minimizer and of improved regular sequence

where the process Λ is realized as

$$\Lambda(t) = \frac{1}{2}\int_{y=-\infty}^{\infty} R(y)\exp(-|t-y|)\,dy, \tag{6}$$

where R is a two-sided 3-dimensional Bessel process. The limit process Γ is a zero-mean Gaussian process with covariance

$$K(t) = \frac{1}{4}\sigma^2(1+|t|)e^{-|t|}. \tag{7}$$

We have

$$E\Lambda(t) \geq E\Lambda(0) = \sqrt{2}.$$

A 3-dimensional Bessel process is, roughly speaking, a Wiener process starting at 0, conditioned to never return to 0. Conditional on $t^*, f(t^*), f(1)$, the post-minimum process $f(t^*+s)-f(t^*)$, $0 \leq s \leq t-t^*$ is a 3-dimensional Bessel bridge. Similarly, the process to the left of t^* is an independent Bessel bridge. The process Λ is a smoothed process derived from the two-sided Bessel process, as indicated in Fig. 3.

Note that Λ is positive and twice continuously differentiable, with

$$\Lambda''(0) = \Lambda(0),\ \Lambda'(0) = \frac{1}{2}\left(\int_{y=0}^{\infty} R(y)e^{-y}\,dy - \int_{y=-\infty}^{0} R(y)e^{y}\,dy\right).$$

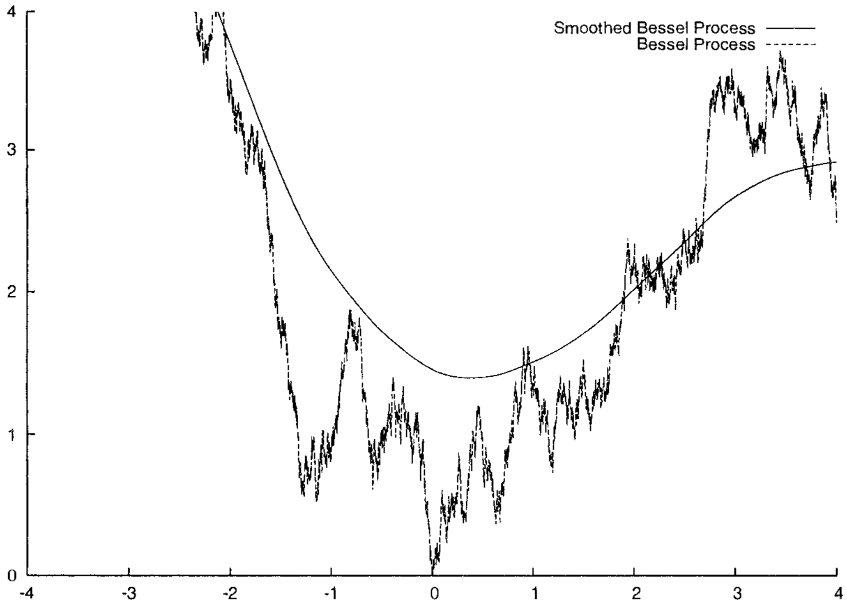


Fig. 3. Smoothed function near minimum

Let θ be the (first) minimizer of $\Lambda + \Gamma = F$. Then

$$\begin{aligned} E\left(F(\theta)\right)^2 &\le E\left(F(0)\right)^2 \\ &= E\left(\Lambda(0) + \Gamma(0)\right)^2 \\ &= E\left(\Lambda(0)\right)^2 + E\left(\Gamma(0)\right)^2 \\ &= E\left(\Lambda(0)\right)^2 + \frac{1}{4}, \end{aligned}$$

where we used the independence of Λ and Γ, and (7).

Now

$$0 \le E\left(\Lambda(0)\right)^2 \le 2.$$

Therefore, in the uniform grid case,

$$\frac{n^{1/4}}{\sigma^{1/2}} e_n \to \left(EF(\theta)^2\right)^{1/2} \in (1/2, 3/2).$$

References

[CŽ05] Calvin, J.M., Žilinskas, A.: One-dimensional global optimization for observations with noise. Computers and Mathematics with Applications, **50**, 157–169 (2005)

[Kus62] Kushner, H.: A versatile stochastic model of a function of unknown and time-varying form. Journal of Mathematical Analysis and Applications, **5**, 150–167 (1962)

[Pla96] Plaskota, L.: Noisy Information and Computational Complexity. Cambridge University Press (1996)

[Rit90] Ritter, K: Approximation and optimization on the Wiener space. Journal of Complexity, **6**, 337–364 (1990)

[TŽ89] Törn, A., Žilinskas, A.: Global Optimization. Springer (1989)

[Was92] Wasilkowski, G.: On average complexity of global optimization problems. Mathematical Programming, **57**, 313–324 (1992)

[Žil80] Žilinskas, A.: Mimun-optimization of one-dimensional multimodal functions in the presence of noise, algoritmus 44. Aplikace Matematiky, **25**, 392–402 (1980)

[Žil81] Žilinskas, A.: Two algorithms for one-dimensional multimodal minimization. Mathematische Operationsforschung und Statistik, ser. Optimization, **12**, 53–63 (1981)

[Žil85] Žilinskas, A.: Axiomatic characterization of a global optimization algorithm and investigation of its search strategies. Operations Research Letters, **4**, 35–39 (1985)

Estimating the Minimal Value of a Function in Global Random Search: Comparison of Estimation Procedures

Emily Hamilton, Vippal Savani, and Anatoly Zhigljavsky

School of Mathematics, Cardiff University, UK ZhigljavskyAA@cf.ac.uk

Summary. In a variety of global random search methods, the minimum of a function is estimated using either one of linear estimators or the the maximum likelihood estimator. The asymptotic mean square errors (MSE) of several linear estimators asymptotically coincide with the asymptotic MSE of the maximum likelihood estimator. In this chapter we consider the non-asymptotic behaviour of different estimators. In particular, we demonstrate that the MSE of the best linear estimator is superior to the MSE of the the maximum likelihood estimator.

1 Introduction

Let $f : A \to \mathbb{R}$ be an objective function defined in feasible region A and $m = \min_{x \in A} f(x)$ be its global minimum. We always assume that A is a compact subset of $\mathbb{R}^d$ for some $d \geq 1$, $\mathrm{vol}(X) > 0$ (where $\mathrm{vol}(\cdot)$ stands for 'volume'), $m > -\infty$ and there is at least one global minimiser; that is, the point $x^* \in A$ such that $f(x^*) = m$. We shall also assume that the objective function f is continuous in the neighbourhood of this minimiser x^*. For simplicity we also assume that the objective function f is bounded from above (this last condition is made purely for technical reasons and can be relaxed).

We consider several estimators of m that are based on taking random samples of n points from the feasible region A and computing the corresponding values of the objective function. The set of n points, $X_n = \{x_1, \ldots, x_n\}$, will be independent and identically distributed random variables/vectors (i.i.d.r.v.) with common distribution P, where P is a probability measure defined on A. We assume that there is a positive probability that a random point x_j will be in the vicinity of x^*.

Let $Y_n = \{f(x_1), \ldots, f(x_n)\} = \{y_1, \ldots, y_n\}$ be the i.i.d.r.v. obtained by computing $f(\cdot)$ at the elements of the sample X_n. The values y_j will have common cumulative distribution function (c.d.f.)

$$F(t) = \mathbb{P}\{x \in A : f(x) \leq t\} = \int_{f(x) \leq t} P(dx) . \qquad (1)$$

The minimum value of $f(\cdot)$ is the essential infimum of the r.v. η with the c.d.f. (1):

$$m = \min_{x \in A} f(x) = \operatorname{ess\,inf} \eta = \inf\{a : F(a) > 0\}\,.$$

2 Several Auxiliary Results

2.1 Asymptotic Distribution of the Minimum Order Statistic

Let the sample size n be fixed and $y_{1,n} \leq \ldots \leq y_{n,n}$ be the order statistics corresponding to the independent random sample Y_n. Here $y_{i,n}$ represents the i-th smallest member within the sample $Y_n = \{y_1, \ldots, y_n\}$.

Consider first the asymptotic distribution of the sequence of minimum order statistic $y_{1,n}$, as $n \to \infty$. Generally, in the case $m = \operatorname{ess\,inf} \eta > -\infty$ (where the random variable η has c.d.f. $F(t)$) there are two limiting distributions possible; however, in global random search applications, when $F(\cdot)$ has the form (1), only one asymptotic distribution arises; specifically, the Weibull distribution with the c.d.f.

$$\Psi_\alpha(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 - \exp\{-z^\alpha\} & \text{for } z \geq 0\,. \end{cases} \tag{2}$$

This c.d.f. has only one parameter, α, which is called 'tail index'. The mean of the Weibull distribution with tail index α is $\Gamma(1 + 1/\alpha)$; the density corresponding to the c.d.f. (2) is

$$\psi_\alpha(t) = (\Psi_\alpha(t))' = \alpha\, t^{\alpha-1} \exp\{-t^\alpha\}\,, \quad t > 0\,. \tag{3}$$

Let κ_n be the $(\frac{1}{n})$-quantile of a c.d.f. $F(\cdot)$; that is, $\kappa_n = \inf\{u | F(u) \geq 1/n\}$. Note that since the objective function $f(\cdot)$ is continuous in a neighbourhood of x^*, the c.d.f. $F(\cdot)$ is continuous in the vicinity of m and for n large enough we have $F(\kappa_n) = 1/n$.

The following classical result from the theory of extreme order statistics is of primary importance for us (see e.g. [DN03, EKM03, Gal87, KN00, Nev01] for proofs, discussions and generalisations).

Theorem 1. *Assume* $\operatorname{ess\,inf} \eta = m > -\infty$, *where* η *has c.d.f.* $F(t)$, *and the function*

$$V(v) = F\left(m + \frac{1}{v}\right), \qquad v > 0,$$

regularly varies at infinity with some exponent $(-\alpha)$, $0 < \alpha < \infty$; *that is,*

$$\lim_{v \to \infty} \frac{V(tv)}{V(v)} = t^{-\alpha}, \qquad \text{for each } t > 0\,. \tag{4}$$

Then

$$\lim_{n\to\infty} F_{1,n}(m + (\kappa_n - m)z) = \Psi_\alpha(z)\,, \tag{5}$$

where $F_{1,n}$ is the c.d.f. of the minimum order statistics $y_{1,n}$, the c.d.f. $\Psi_\alpha(z)$ is defined in (2) and κ_n is the $\left(\frac{1}{n}\right)$-quantile of $F(\cdot)$.

The asymptotic relation (5) means that the distribution of the sequence of random variables $(y_{1,n} - m)/(\kappa_n - m)$ converges (as $n \to \infty$) to the random variable with c.d.f. $\Psi_\alpha(z)$.

The c.d.f. $\Psi_\alpha(z)$, along with its limiting case $\Psi_\infty(z) = \lim_{\alpha\to\infty} \Psi_\alpha(1 + z/\alpha)$ $= 1 - \exp\{-\exp(z)\}$, $-\infty < z < \infty$, are the only nondegenerate limits of the c.d.f.'s of the sequences $(y_{1,n} - a_n)/b_n$, where $\{a_n\}$ and $\{b_n\}$ are arbitrary sequences of positive numbers.

If there exist numerical sequences $\{a_n\}$ and $\{b_n\}$ such that the c.d.f.'s of $(y_{1,n} - a_n)/b_n$ converge to Ψ_α, then we say that $F(\cdot)$ belongs to the domain of attraction of $\Psi_\alpha(\cdot)$ and express this as $F \in D(\Psi_\alpha)$. The conditions stated in Theorem 1 are necessary and sufficient for $F \in D(\Psi_\alpha)$. There are two conditions: $m = \operatorname{ess\,sup} \eta < \infty$ and the condition (4). The first one is always valid in global random search applications. The condition (4) demands more attention. For example, it is never valid in discrete optimization problems as in these problems the c.d.f. $F(\cdot)$ is not continuous (at the same time, (4) implies that $F(\cdot)$ has to be continuous in the vicinity of m). In fact, for a c.d.f. with a jump at its lower end-point no non-degenerate asymptotic distribution for $y_{1,n}$ exists, whatever the normalization (that is, sequences $\{a_n\}$ and $\{b_n\}$).

The condition (4) can be written as

$$F(t) = c_0(t - m)^\alpha + \mathrm{o}((t - m)^\alpha) \qquad \text{as } t \downarrow m\,, \tag{6}$$

where c_0 is a function of $v = 1/(t - m)$, slowly varying at infinity as $v \to \infty$. Of course, any positive constant is a slowly varying function, but the actual range of eligible functions c_0 is much wider.

The following condition is stronger that the condition (6) and often used for justifying properties of the maximum likelihood estimators (MLE):

$$F(t) = c_0(t - m)^\alpha \left(1 + \mathrm{O}((t - m)^\gamma)\right) \qquad \text{as } t \downarrow m \tag{7}$$

for some positive constants c_0, α and γ. This condition is sometimes called Hall's condition as it has appeared in P.Hall's paper [Hal82] which was devoted to the MLE estimators of m.

2.2 Determining the Exact Value of the Tail Index

There is extensive literature devoted to the problem of making statistical inference about the value of the tail index α, see [EKM03, BGTS04] for reviews, discussions and references.

In problems of global optimization we can establish a direct link between the form (1) of the c.d.f. $F(\cdot)$, the condition (6), and the value of the tail index α. The basic result is as follows.

Theorem 2. *Assume that the objective function f is such that its global minimiser x^* is unique and*

$$f(x)-m = w(||x-x^*||)H(x-x^*) + O(||x-x^*||^{\beta}), \quad ||x-x^*|| \to 0, \tag{8}$$

for some homogeneous function $H : \mathbb{R}^d \backslash \{0\} \to (0, \infty)$ of order $\beta > 0$ (for H the relation $H(\lambda z) = \lambda^{\beta} H(z)$ holds for all $\lambda > 0$ and $z \in \mathbb{R}^d$) and function $w : \mathbb{R} \to \mathbb{R}$ is positive and continuous. Then the condition (6) for the c.d.f. (1) holds and the value of the tail index α is equal to $\alpha = d/\beta$.

Proof of this result and many generalisations can be found in [dH81, Zhi79, Zhi91, ZZ06].

Two important particular cases of (8) are:

- let $f(\cdot)$ be twice continuously differentiable in the vicinity of x^*, $\nabla f(x^*) = 0$ (here $\nabla f(x^*)$ is the gradient of $f(\cdot)$ in x^*) and the Hessian $\nabla^2 f(x^*)$ of $f(\cdot)$ at x^* is nondegenerate; in this case, we can take
$$w(\cdot) = 1, \quad H(z) = -z'[\nabla^2 f(x^*)]z\,,$$
which implies $\beta = 2$ and $\alpha = d/2$;
- let all components of $\nabla f(x^*)$ be finite and non-zero which often happens if the global minimum of $f(\cdot)$ is achieved at the boundary of A. Then we may take $H(z) = z'\nabla f(x^*)$, $w(\cdot) = 1$; this gives $\beta = 1$ and $\alpha = d$.

2.3 Asymptotic Behaviour of $\kappa_n - m$

The quantity $\kappa_n - m$, where $m = \operatorname{ess\,inf} \eta$ and κ_n is the $(\frac{1}{n})$-quantile of $F(\cdot)$, enters many formulae below and therefore its asymptotic behaviour is very important. Fortunately, the asymptotic behaviour of $\kappa_n - m$ is clear. Indeed, as long as (6) holds with some c_0, we have

$$\frac{1}{n} = F(\kappa_n) \sim c_0\,(\kappa_n - m)^{\alpha} \qquad \text{as } n \to \infty$$

implying

$$(\kappa_n - m) \sim (c_0 n)^{-1/\alpha} \qquad \text{as } n \to \infty\,. \tag{9}$$

3 Defining the Estimators

Let $Y_n = \{y_1, \dots, y_n\}$ be an independent sample of values from the c.d.f. (1) and $y_{1,n} \leq \dots \leq y_{n,n}$ be the corresponding order statistics. In this section, following the exposition of [Zhi91], we define five estimators of m that will be numerically studied below. Note that there are some other estimators available; we only define the estimators which we believe are the most important ones.

For constructing the estimators of m, only the first k order statistics $\{y_{1,n}, \dots, y_{k,n}\}$ will be used (here k is much smaller than n). There are two major reasons for this: (a) the higher order statistics contain very little information about m; and (b) we can use the limit theorem for extreme order statistics (Theorem 1) only if k is such that $k/n \to 0$ as $n \to \infty$.

The estimators rely on the assumption that the c.d.f. $F(\cdot)$ satisfies the condition (6) with known value of the tail index α.

3.1 Maximum Likelihood Estimator

In defining the maximum likelihood estimator (MLE) we have to assume the condition (7) which is stronger than (6). Additionally, we have to assume $\alpha \geq 2$ (numerical study shows that the MLE as defined below coincides with the minimum order statistic $y_{1,n}$ for $\alpha < 2$).

Taking the asymptotic form of the likelihood function as exact (for details see [Hal82] and [Zhi91], Chapter 7), we obtain that the maximum likelihood estimator of the minimum, $\hat{m}^*$, is the solution in z to the following likelihood equation:

$$(\alpha - 1) \sum_{j=1}^{k-1} \frac{(y_{k,n} - y_{j,n})}{(y_{j,n} - z)} = k$$

conditionally $z < y_{1,n}$; if there are no solutions to this equation for $z < y_{1,n}$, then we set $\hat{m}^* = y_{1,n}$; if there are more than one solution in the region $z \in (-\infty, y_{1,n})$, then we take the smallest of these solutions.

If the conditions (7), $\alpha \geq 2$, $k \to \infty$, $k/n \to 0$ (as $n \to \infty$) are satisfied then the maximum likelihood estimators of m are asymptotically normal and asymptotically efficient in the class of asymptotically normal estimators and their mean square error $E(\hat{m} - m)^2$ is asymptotically (as $n \to \infty$)

$$E(\hat{m} - m)^2 \sim \begin{cases} (1 - \frac{2}{\alpha})(\kappa_n - m)^2\, k^{-1+2/\alpha} & \text{for } \alpha > 2, \\ (\kappa_n - m)^2 \log k & \text{for } \alpha = 2\,. \end{cases} \tag{10}$$

3.2 Linear Estimators

We will now define four different linear estimators.

A general linear estimator of m can be written as

$$\hat{m}_{n,k}(a) = \sum_{i=1}^{k} a_i y_{i,n}\,,$$

where $a = (a_1, \ldots, a_k)' \in \mathbb{R}^k$ is a vector of coefficients. It can be shown (see [Zhi91], Sect. 7.3) that as $n \to \infty$ we have

$$E\hat{m}_{n,k}(a) = m \sum_{i=1}^{k} a_i - (\kappa_n - m)a'b + o(\kappa_n - m) = m \sum_{i=1}^{k} a_i + o(1) . \quad (11)$$

Here $b = (b_1, \ldots b_k)' \in \mathbb{R}^k$, where $b_i = \Gamma(i + 1/\alpha) \,/\, \Gamma(i)$.

From (11) it is clear that (as the objective function f is bounded and therefore the variances of all $y_{i,n}$ are finite), a necessary and sufficient condition for an estimator with vector of coefficients a to be consistent is:

$$\sum_{i=1}^{k} a_i = 1 . \quad (12)$$

Additionally, to ensure a small bias we may require

$$\sum_{i=1}^{k} a_i b_i = 0 . \quad (13)$$

The main criteria is the mean square error (MSE) given by

$$E(\hat{m}_{n,k}(a) - m)^2 \sim (\kappa_n - m)^2 \, a'\Lambda a, \quad n \to \infty , \quad (14)$$

where $\Lambda = \|\lambda_{ij}\|_{i,j=1}^{k}$ is a symmetric $k \times k$-matrix with elements λ_{ij} defined for $i \geq j$ by the formula

$$\lambda_{ij} = \frac{\Gamma(i+2/\alpha)\,\Gamma(j+1/\alpha)}{\Gamma(i+1/\alpha)\,\Gamma(j)} .$$

Optimal Linear Estimator

The r.h.s. of (14) is a natural optimality criterion for selecting the vector a. The optimal consistent estimator $m^\circ = \hat{m}_{n,k}(a^\circ)$, we shall call it *the optimal linear estimator*, is determined by the vector of coefficients

$$a^\circ = \arg \min_{a: a'\mathbf{1}=1} a'\Lambda a = \frac{\Lambda^{-1}\mathbf{1}}{\mathbf{1}'\Lambda^{-1}\mathbf{1}} . \quad (15)$$

The estimator m° has been suggested in [Coo79], where the form (15) for the vector of coefficients was obtained. Solving the quadratic programming problem in (15) is straightforward. In the process of doing that, we obtain

$$\min_{a: a'\mathbf{1}=1} a'\Lambda a = (a^*)'\Lambda a^* = 1/\mathbf{1}'\Lambda^{-1}\mathbf{1} . \quad (16)$$

Lemma 7.3.4 in [Zhi91] gives the following expression for the r.h.s. of (16):

$$\mathbf{1}'\Lambda^{-1}\mathbf{1} = \begin{cases} \frac{1}{\alpha-2}\left(\frac{\alpha\Gamma(k+1)}{\Gamma(k+2/\alpha)} - \frac{2}{\Gamma(1+2/\alpha)}\right) & \text{for } \alpha \neq 2, \\ \sum_{i=1}^{k} 1/i & \text{for } \alpha = 2\,; \end{cases}$$

this expression is valid for all $\alpha > 0$ and $k = 1, 2, \ldots$

The components a_i° $(i = 1, \ldots, k)$ of the vector a° can be evaluated explicitely:

$$a_i^\circ = u_i/\mathbf{1}'\Lambda^{-1}\mathbf{1} \quad \text{for} \quad i = 1, \ldots, k \tag{17}$$

with

$$\begin{aligned} u_1 &= (\alpha+1)\,/\,\Gamma(1+2/\alpha), \\ u_i &= (\alpha-1)\,\Gamma(i)/\Gamma(i+2/\alpha) \qquad \text{for } i = 2, \ldots, k-1, \\ u_k &= -(\alpha k - \alpha + 1)\Gamma(k)\,/\,\Gamma(k+2/\alpha). \end{aligned}$$

Deriving this expression for the coefficients of the vector a° is far from trivial, see [Zhi91], Sect. 7.3.3.

The asymptotic properties of the optimal linear estimators coincide with the properties of the maximum likelihood estimators and hold under the same regularity conditions (we again refer to [Zhi91], Sect. 7.3.3). In particular, the optimal linear estimators $m^\circ = \hat{m}_{n,k}(a^\circ)$ of m are asymptotically normal and their MSE $E(m^\circ - m)^2$ asymptotically behaves like the r.h.s. of (10). Unlike the MLE, this estimator is defined for all $\alpha > 0$ and indeed behaves well for small α, see below.

The Second Optimal Linear Estimator

The coefficients of this estimator (which was suggested in [Coo80]) are chosen to minimise the MSE in the class of linear estimators satisfying the conditions (12) and (13):

$$a^\triangle = \arg \min_{\substack{a:\, a'\mathbf{1}=1, \\ a'b=0}} a'\Lambda a = \frac{\Lambda^{-1}\mathbf{1} - (b'\Lambda^{-1}\mathbf{1})\Lambda^{-1}b/(b'\Lambda^{-1}b)}{\mathbf{1}'\Lambda^{-1}\mathbf{1} - (b'\Lambda^{-1}\mathbf{1})^2/(b'\Lambda^{-1}b)}\,.$$

Asymptotic properties of the estimator $m^\triangle = \hat{m}(a^\triangle)$ (for $\alpha \geq 2$, $n \to \infty$, $k \to \infty$ and $k/n \to 0$) are the same as the asymptotic properties of the MLE and the estimator m°. For fixed k, the behaviour of the MSE is different, see below.

Csörgö-Mason Estimator

The Csörgö-Mason estimator suggested and studied in [CM89] is linear consistent and has similar asymptotic properties to the maximum likelihood estimator and the first two linear estimators for $\alpha \geq 2$, $n \to \infty$, $k \to \infty$ and $k/n \to 0$; for fixed k, the behaviour of the MSE of the Csörgö-Mason estimator is different.

The estimator is defined by the vector of coefficients $a^{\square} = \left(a_1^{\square}, \dots, a_n^{\square}\right)'$, where

$$a_i^{\square} = \begin{cases} v_i & \text{for } \alpha > 2, \quad i = 1, \dots, k-1 \\ v_k + 2 - \alpha & \text{for } \alpha > 2, \quad i = k \\ 2/\log(k) & \text{for } \alpha = 2, \quad i = 1 \\ \log(1+1/i)/\log(k) & \text{for } \alpha = 2, \quad i = 2, \dots, k-1 \\ \left(\log(1+1/k) - 2\right)/\log(k) & \text{for } \alpha = 2, \quad i = k \end{cases}$$

with

$$v_j = (\alpha - 1)k^{2/\alpha - 1}\left(j^{1-2/\alpha} - (j-1)^{1-2/\alpha}\right).$$

Minimum Order Statistic

The simplest estimator that can be used is the minimum order statistic, $m^{\bullet} = \hat{m}(a^{\bullet}) = y_{1,n}$. The vector of coefficients for this estimator is

$$a^{\bullet} = (1, 0, \dots, 0)'.$$

4 Simulation Study for Comparing the Estimators

In this section we make a comparison of the efficiency and bias for the maximum likelihood and linear estimators of m given finite samples of size n drawn from the Weibull distribution with density function (3).

The values of α considered when sampling from the Weibull distribution will be $\alpha \in \{1, 2, 5, 10\}$ and the parameter k for the number of order statistics used in the computation of estimators will be $k \le 20$. We must assume that the original sample size n is large enough so that the asymptotic distribution for the minimal statistic has been reached, this asymptotic distribution will also be Weibull.

Note that the c.d.f. of the Weibull distribution can be represented in the form of (7) with $m = 0$ since $1 - \exp\{-t^{\alpha}\} = t^{\alpha}(1 + O(t^{\alpha}))$ as $t \to 0$.

By definition, the optimal linear estimator $m^{\circ} = \hat{m}_{n,k}(a^{\circ})$, with a° given in (17), provides the lowest MSE in the class of all linear consistent estimators as $n \to \infty$. In view of (14) and (16), we have for the asymptotic MSE of m°:

$$\lim_{n \to \infty} \frac{\mathbf{1}'\Lambda^{-1}\mathbf{1}}{(\kappa_n - m)^2} \mathrm{MSE}(m^{\circ}) = 1\,, \tag{18}$$

for any k. Therefore, for fixed n and k it is natural to define the efficiency of an estimator $\hat{m}$ as

$$\frac{(\kappa_n - m)^2}{\mathbf{1}'\Lambda^{-1}\mathbf{1}} / \mathrm{MSE}(\hat{m})\,. \tag{19}$$

Since we consider finite samples, it is possible for the efficiency to be slightly greater than 1.

The efficiency of an estimator $\hat{m}$ is estimated based on taking $R = 10\,000$ estimators of $\hat{m}_j$, where each $\hat{m}_j$ is estimated from a sample of size n:

$$\mathrm{MSE}(\hat{m}) \approx \frac{1}{R}\sum_{j=1}^{R}(\hat{m}_j - m)^2 .$$

Thus, for fixed k, n and R, we use the following definition of efficiency of an estimator $\hat{m}$:

$$\mathrm{eff}(\hat{m}) = \left[\frac{(\kappa_n - m)^2}{\mathbf{1}'\Lambda^{-1}\mathbf{1}}\right] \Big/ \left[\frac{1}{R}\sum_{j=1}^{R}(\hat{m}_j - m)^2\right] , \qquad (20)$$

where in our case $m = 0$, $R = 10\,000$ and k varies. As $R \to \infty$, the finite-sample efficiency (20) tends to (19).

Figure 1 shows the efficiency of the maximum likelihood estimator (MLE), the consistent linear estimators and the minimum order statistic for α=1, 2, 5 and 10 against different values of k. The efficiency is based on taking $R = 10\,000$ estimators of $\hat{m}_i$, where each $\hat{m}_i$ is estimated from a sample of size $n = 100$. The efficiency curves of the MLE are rather uneven.

Figure 1 demonstrates that the MSE of the optimal linear estimator m° is very close to the asymptotically optimal value of the MSE given by (18) for all $\alpha \geq 1$ (sometimes this MSE is even smaller than the best possible asymptotic value). The estimator m° clearly provides the lowest MSE in the class of estimators considered. The efficiency of the MLE is typically lower than the efficiency of m°, especially when α is small; note that MLE can only be used for $\alpha \geq 2$.

The efficiency of the minimum order statistic decreases monotonically as $k \to \infty$, this is because the estimator is not using $k-1$ out of k order statistics. The efficiency of the linear estimator $m^{\triangle}$ is poor for small k (as the unbiasedness condition (13) takes away one degree of freedom for the coefficients) but increases monotonically as k increases. The efficiencies of the minimum order statistic and the $m^{\triangle}$ estimators are equal for $k = 2$. This can be verified by considering the asymptotic MSE (as $n \to \infty$) of these two estimators at this point. The efficiency of the Csörgö-Mason estimators $\overset{\square}{m}$ is poor for small α (note this estimator is only defined for $\alpha \geq 2$) but gets better when α increases; thus, for $\alpha = 10$ the efficiency of $\overset{\square}{m}$ is basically 1.

Figure 2 shows the normalized bias, $n^{1/\alpha}\frac{1}{R}\sum_{i=1}^{R}|\hat{m}_i - m|$, of different estimators for $n = 100$, $R = 10\,000$, $\alpha = 1, 2, 5, 10$ and variable k. The short summary is: the normalized bias of all four estimators (namely, MLE, m°, $m^{\triangle}$ and $\overset{\square}{m}$) improves as both k and α increase; for large k and α this bias is approximately the same; for small α the bias of the Csörgö-Mason estimator is large but the bias of the other three estimators is comparable for all $\alpha \geq 2$ (note again that MLE is properly defined only for $\alpha \geq 2$).

The case of small values of α has a particular interest. Unlike the MLE and the Csörgö-Mason estimator, the linear estimators m° and $m^\triangle$ are defined in the region $0 < \alpha < 2$ and, as Figs. 3 and 4 demonstrate, behave rather well.

The smallest value of α when all four estimators under consideration are defined is $\alpha = 2$. Figures 5 and 6 display the efficiency and bias of the four estimators in this case. The optimal linear estimator m° is clearly the best. The estimator $m^\triangle$ is also rather efficient, except when k is very small. The efficiency (and bias) of the MLE is worse than the efficiency of the optimal linear estimator m° and for $k \geq 4$ it is also worse than the efficiency (and bias) of $m^\triangle$. The behaviour of the Csörgö-Mason estimator is reasonably poor for $\alpha = 2$; it is especially poor when the ratio k/n is not small enough.

Figures 7, 8 and 9 provide a number of histograms of the normalized estimators $n^{(1/\alpha)}(\hat{m}_j - m)$, $j = 1, \ldots, 10\,000$, plotted with a normal probability density function whose parameters are taken from the sample. One of the conclusions that can be derived from comparing these histograms with the behaviour of the MSE and bias of the corresponding estimators is that a relatively poor behaviour of some of these estimators for small α and/or small k is related more to an increase in bias than to an increase in variance of the estimators.

Figures 10, 11 and 12 display the scatter plots of all five estimators showing points of $n^{(1/\alpha)}(\hat{m}_j - m)$ for each estimator calculated for the same sample. The samples are ordered according to the values of the MLE and only every 10th point of the original sample of $R = 10\,000$ is plotted in order to make the graphs clearer. The minimal order statistics are included to show the upper bounds for the estimators. Graphs on the right-hand side show the same normalized estimators, each averaged over non-intersecting groups of 200 i.e. $j \in \{(1 \ldots 200), \ldots, (9\,801 \ldots 10\,000)\}$ and plotted at the midpoints of each group. These figures show that when $\alpha = 2$ the four main estimators are rather different (except the linear estimators m° and $m^\triangle$ which are becoming more and more similar as k increases). However, as α and j increase the three estimators, MLE, m° and $m^{\square}$, become almost indistinguishable.

Figure 13 shows the scatter plots of $R = 10\,000$ points of $n^{(1/\alpha)}(\hat{m}_i - m)$ for MLE plotted against $n^{(1/\alpha)}(m_i^\circ - m)$ (the normalised optimal linear estimator). This figure suggests that one of the reasons why the MLE looses in efficiency to the optimal linear estimator is the fact that for certain samples the likelihood function does not have maximum in the region $(-\infty, y_{1,n})$ implying that MLE coincides with $y_{1,n}$ which is an inefficient estimator. The same reason explains the uneven behaviour of the MLE efficiency graphs (see Figs. 1, 3 and 4).

5 Simulation Results

Efficiency

MLE ($\cdots$) m° ($\circ$) $m^{\triangle}$ ($\triangle$) $m^{\square}$ ($\square$) $m^{\bullet}$ ($\bullet$)

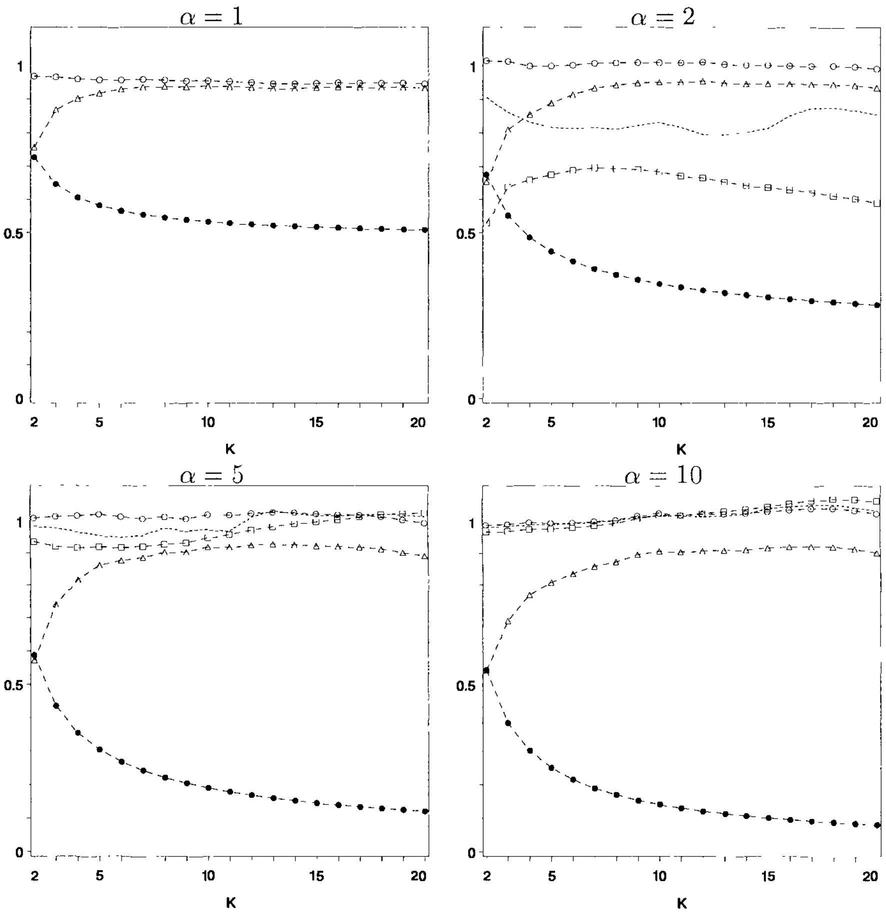


Fig. 1. Efficiency as defined in (20) of different estimators; $n = 100$, $R = 10\,000$, $\alpha = 1, 2, 5, 10$, against k

Bias

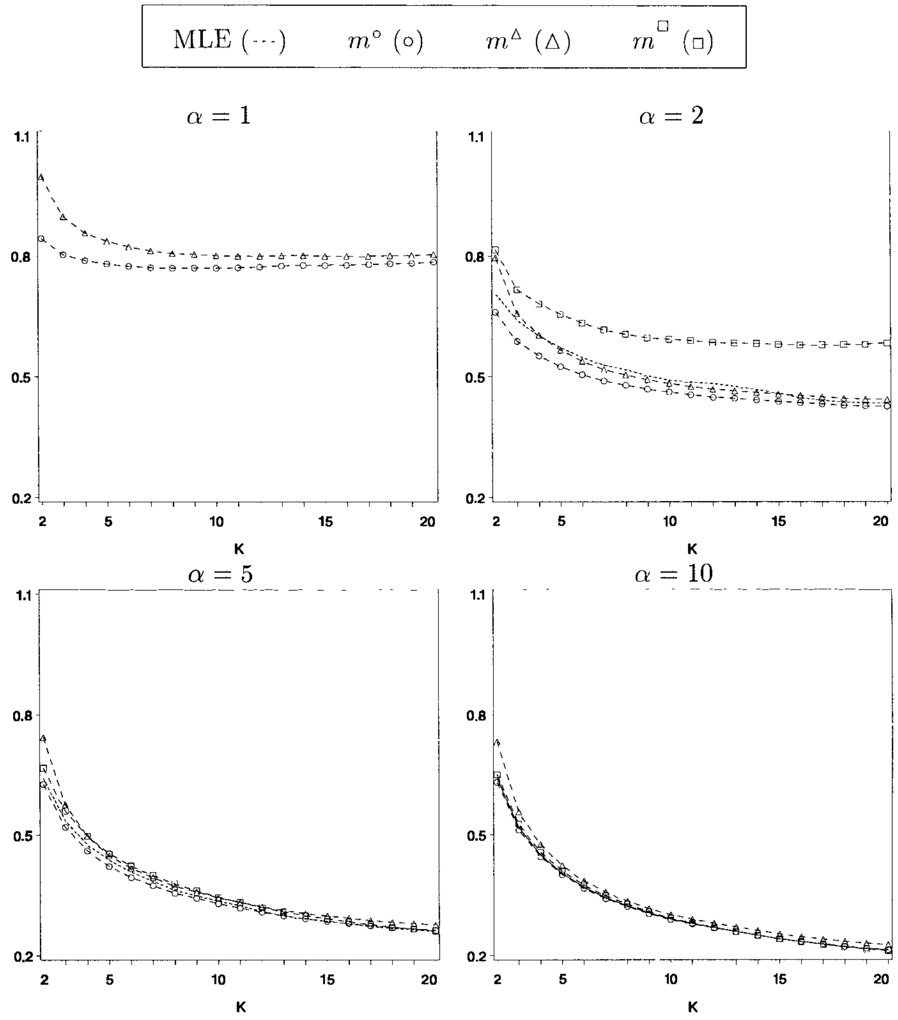


Fig. 2. Normalized bias, $n^{1/\alpha}\frac{1}{R}\sum_{i=1}^{R}|\hat{m}_i - m|$, of different estimators for $n = 100$, $R = 10\,000$, $\alpha = 1, 2, 5$ and 10, against k

Behaviour of Two Linear Estimators for $\alpha = 1$

··· $n = 100$ – – $n = 500$ — $n = 1000$

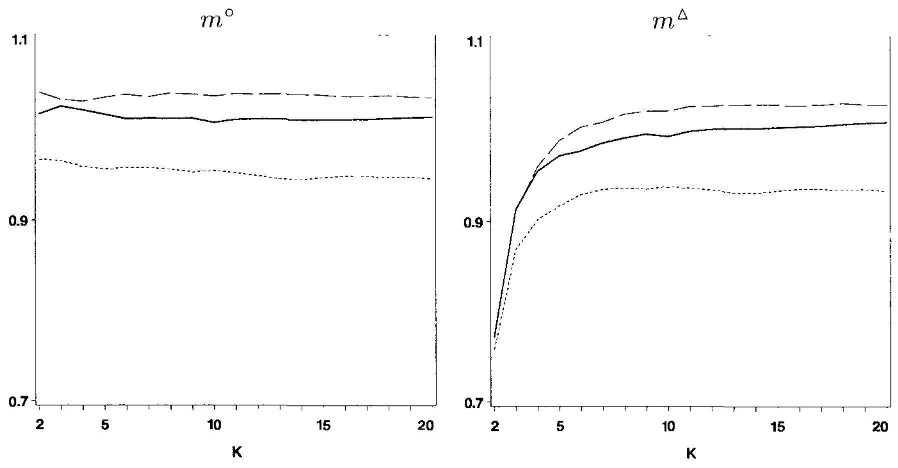


Fig. 3. Efficiency as defined in (20) of two linear estimators m° and m^Δ for $\alpha = 1$, sample sizes $n = 100, 500, 1000$ against k ($R = 10\,000$)

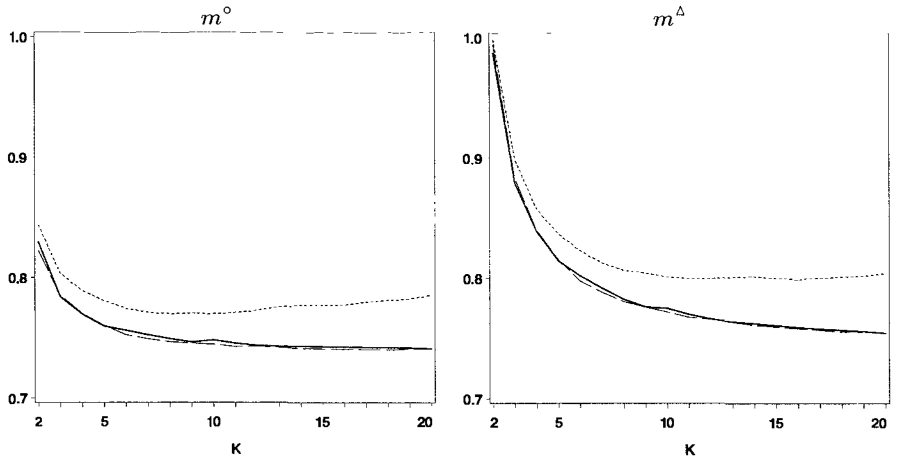


Fig. 4. Normalized bias, $n^{1/\alpha} \frac{1}{R} \sum_{i=1}^{R} |\hat{m}_i - m|$, of two linear estimators m° and m^Δ for $\alpha = 1$ and sample sizes $n = 100, 500, 1000$ against k ($R = 10\,000$)

Efficiency of the Four Estimators for $\alpha = 2$

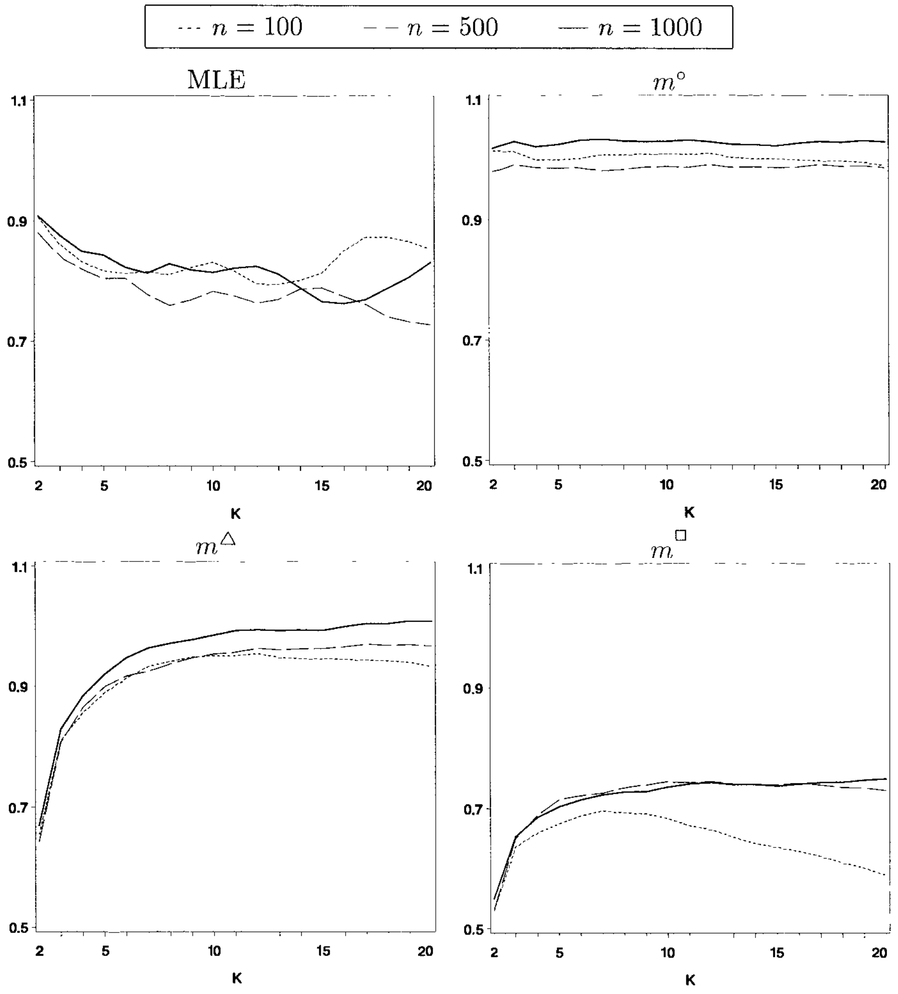


Fig. 5. Efficiency as defined in (20) of different estimators for $\alpha = 2$ and sample sizes $n = 100$, 500, 1000 against k ($R = 10\,000$)

Bias of the Four Estimators for $\alpha = 2$

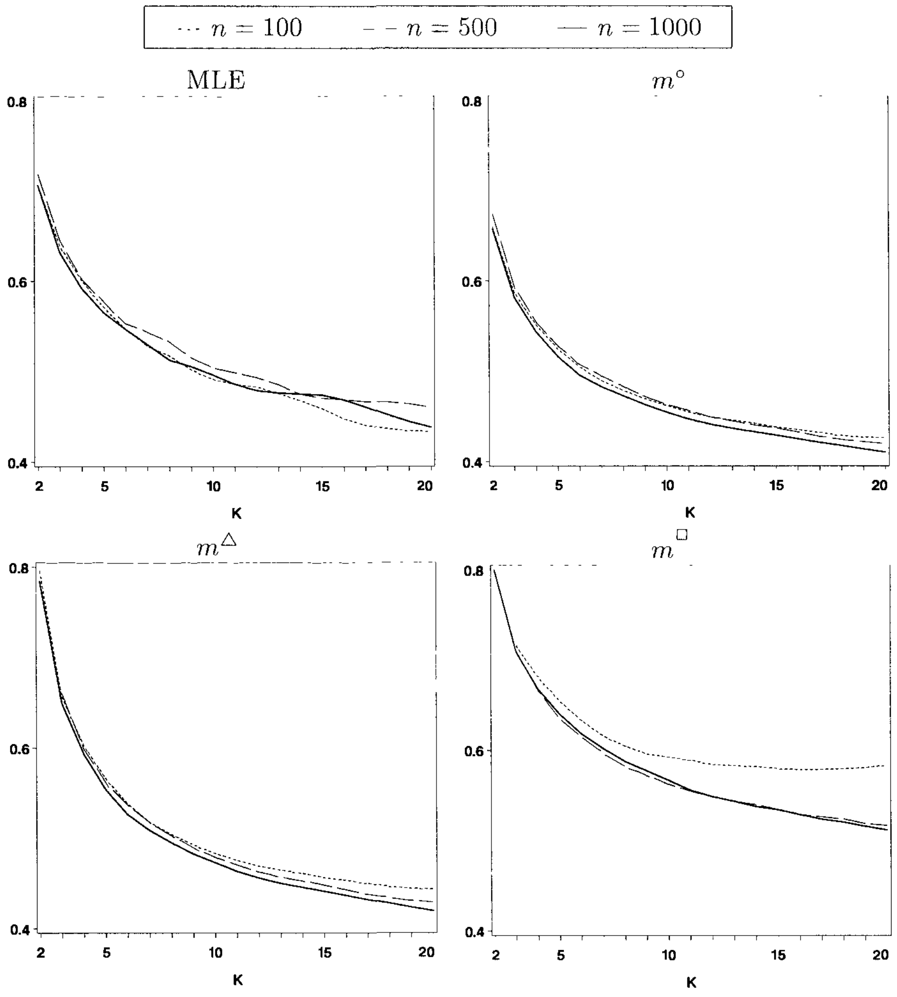


Fig. 6. Normalised bias, $n^{1/\alpha} \frac{1}{R} \sum_{i=1}^{R} |\hat{m}_i - m|$, of different estimators for $\alpha = 2$ and sample sizes $n = 100, 500, 1000$ against k ($R = 10\,000$)

References

[BGTS04] Beirlant, J., Goegebeur, Y., Teugels, J., Segers, J.: Statistics of Extremes. John Wiley & Sons Ltd., Chichester (2004)

[CM89] Csörgő, S., Mason, D.M.: Simple estimators of the endpoint of a distribution. In: Extreme Value Theory (Oberwolfach, 1987). Vol. 51 of Lecture Notes in Statist., pp. 132–147. Springer, New York (1989)

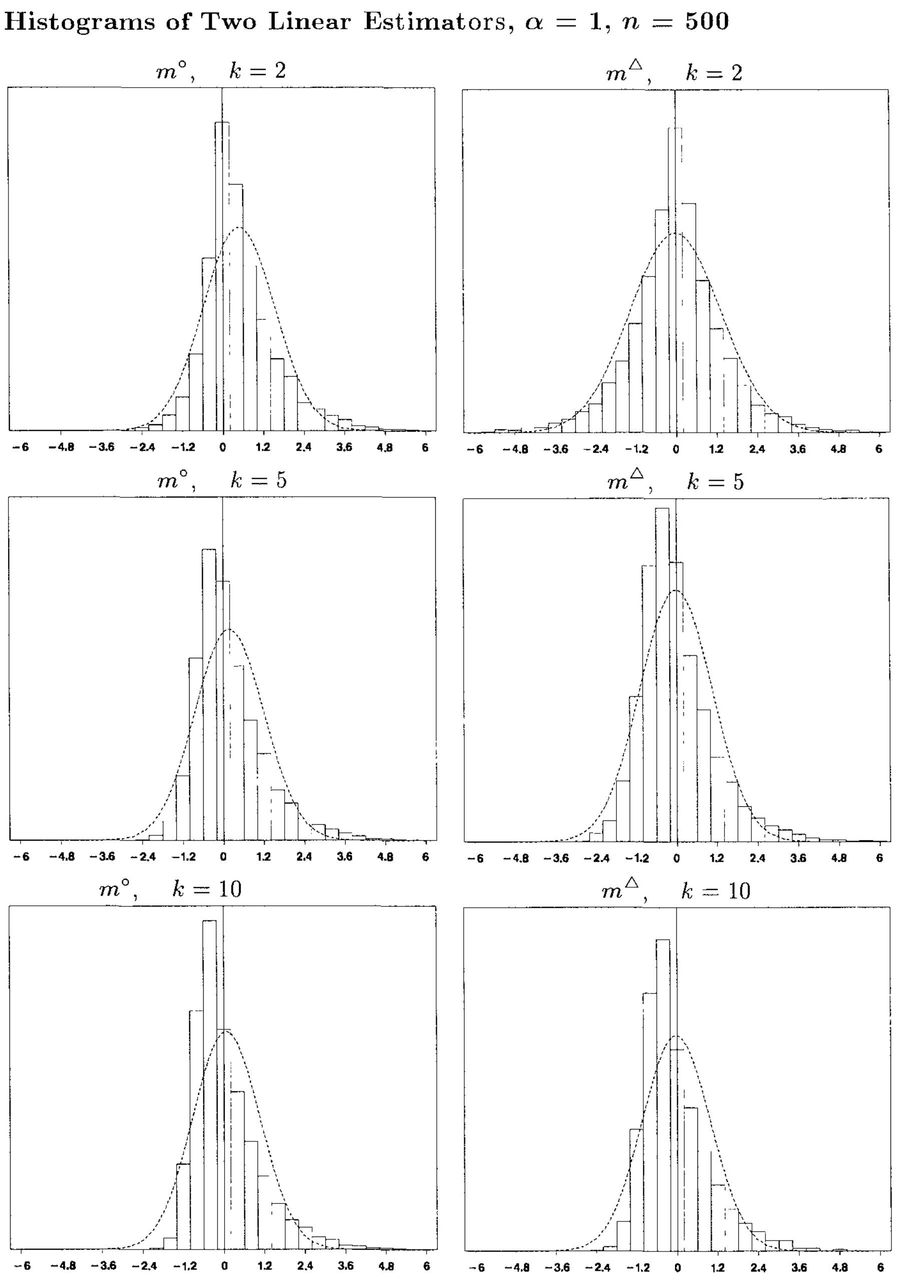


Fig. 7. Histograms of the normalized estimators $n^{(1/\alpha)}(\hat{m}_j - m)$, $j = 1, \dots, 10\,000$, plotted with a normal probability density function whose parameters are taken from the sample

Histograms of the Four Estimators, $\alpha = 2$, $k = 2$, $n = 500$

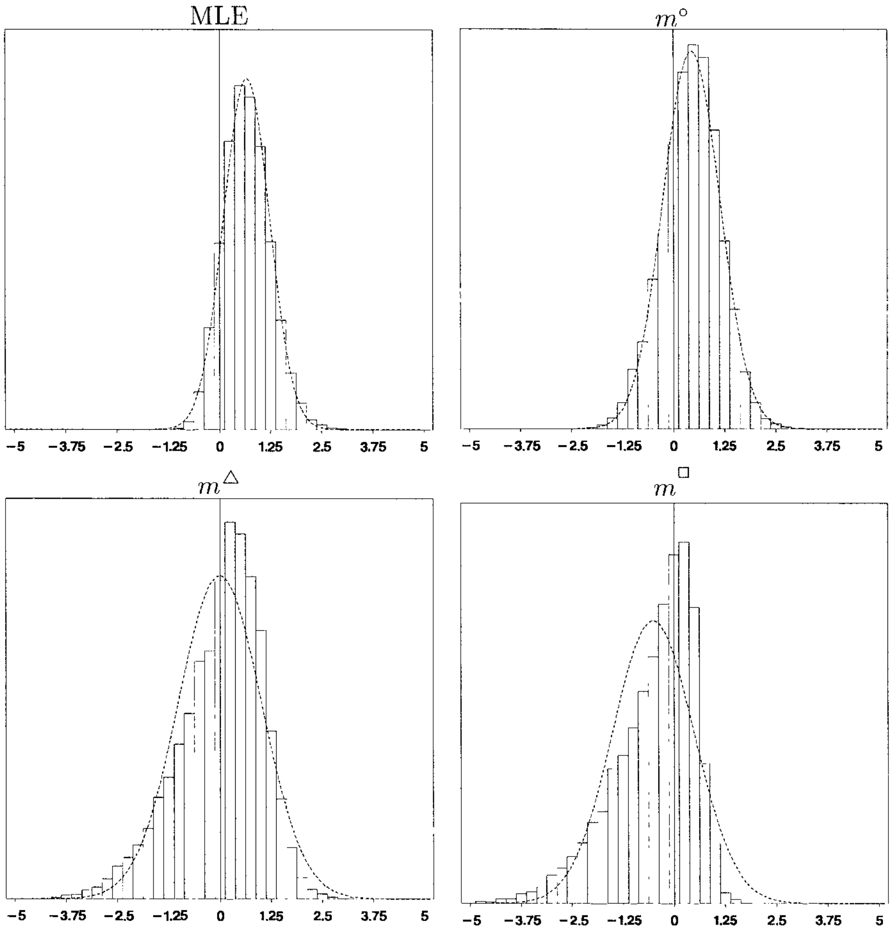


Fig. 8. Histograms of the normalized estimators $n^{(1/\alpha)}(\hat{m}_j - m)$, $j = 1, \ldots, 10\,000$, plotted with a normal probability density function whose parameters are taken from the sample

[Coo79] Cooke, P.: Statistical inference for bounds of random variables. Biometrika, **66**(2), 367–374 (1979)

[Coo80] Cooke, P.: Optimal linear estimation of bounds of random variables. Biometrika, **67**(1), 257–258 (1980)

[dH81] de Haan, L.: Estimation of the minimum of a function using order statistics. J. Amer. Statist. Assoc., **76**(374), 467–469 (1981)

[DN03] David, H.A., Nagaraja, H.N.: Order Statistics. Wiley Series in Probability and Statistics. 3rd ed. John Wiley & Sons, N.Y (2003)

[EKM03] Embrechts, P., Klüppelberg, C., Mikosch, T.: Modelling extremal events for insurance and finance. Springer-Verlag, Berlin (2003)

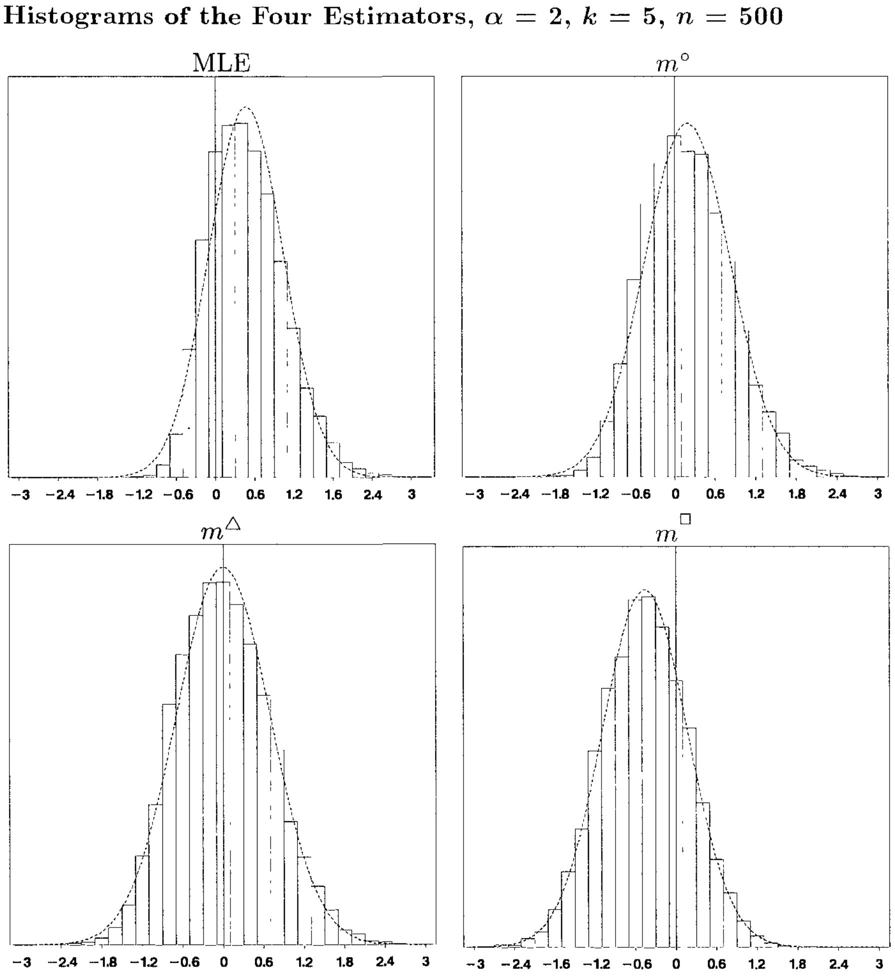


Fig. 9. Histograms of the normalized estimators $n^{(1/\alpha)}(\hat{m}_j - m)$, $j = 1, \ldots, 10\,000$, plotted with a normal probability density function whose parameters are taken from the sample

[Gal87] Galambos, J.: The asymptotic theory of extreme order statistics. 2nd ed. Robert E. Krieger Publishing Co. Inc., Melbourne, FL (1987)

[Hal82] Hall, P.: On estimating the endpoint of a distribution. Ann. Statist., **10**(2), 556–568 (1982)

[KN00] Kotz, S., Nadarajah, S.: Extreme Value Distributions. Imperial College Press, London (2000)

[Nev01] Nevzorov, V.B.: Records: Mathematical Theory. Vol. 194 of Translations of Mathematical Monographs. American Mathematical Society, Providence, RI (2001) Translated from the Russian manuscript by D.M. Chibisov

Scatter Plots of the Estimators, $\alpha = 2$, $n = 500$

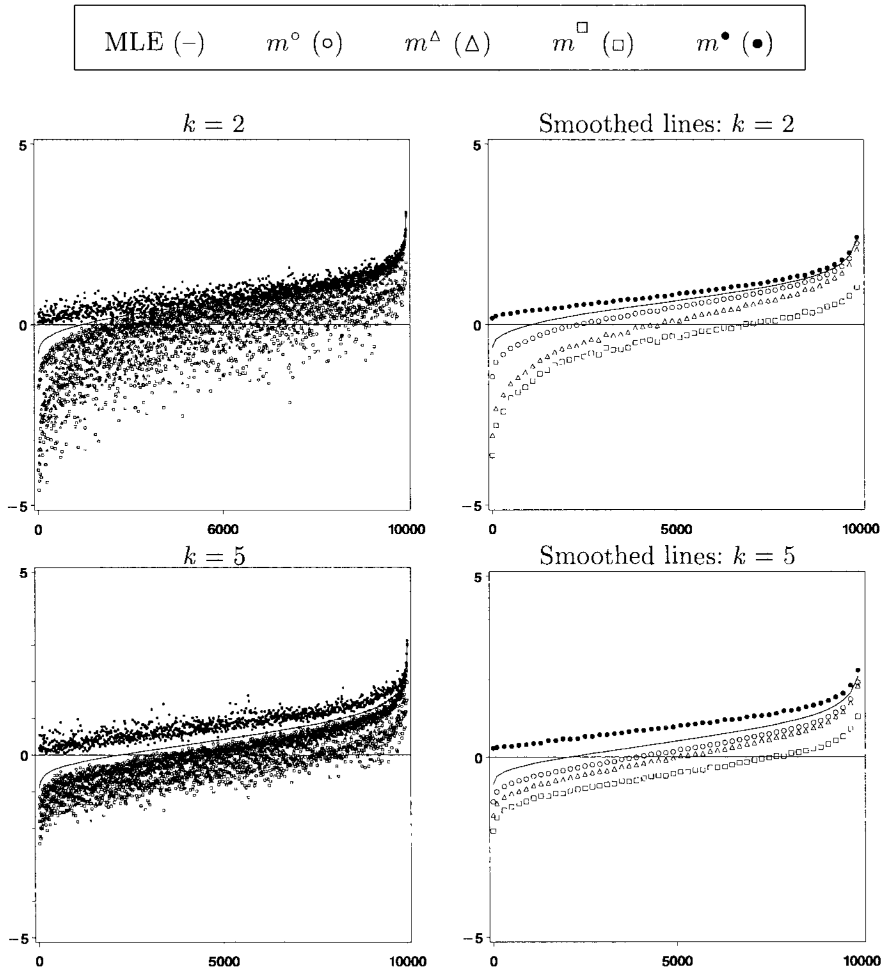


Fig. 10. Scatter plots showing points of $n^{(1/\alpha)}(\hat{m}_j^* - m)$ with the corresponding normalized linear estimators $n^{(1/\alpha)}(\hat{m}_j - m)$ for $j \in \{1, 10, 20 \ldots 10\,000\}$, where $\hat{m}_j^*$ is the j^{th} smallest MLE from a sample of $R = 10\,000$. Graphs on the right-hand side show the same normalized estimators, each averaged over non-intersecting groups of 200 i.e. $j \in \{(1 \ldots 200), \ldots, (9\,801 \ldots 10\,000)\}$ and plotted at the midpoints of each group

Scatter Plots of the Estimators, $\alpha = 5$, $n = 500$

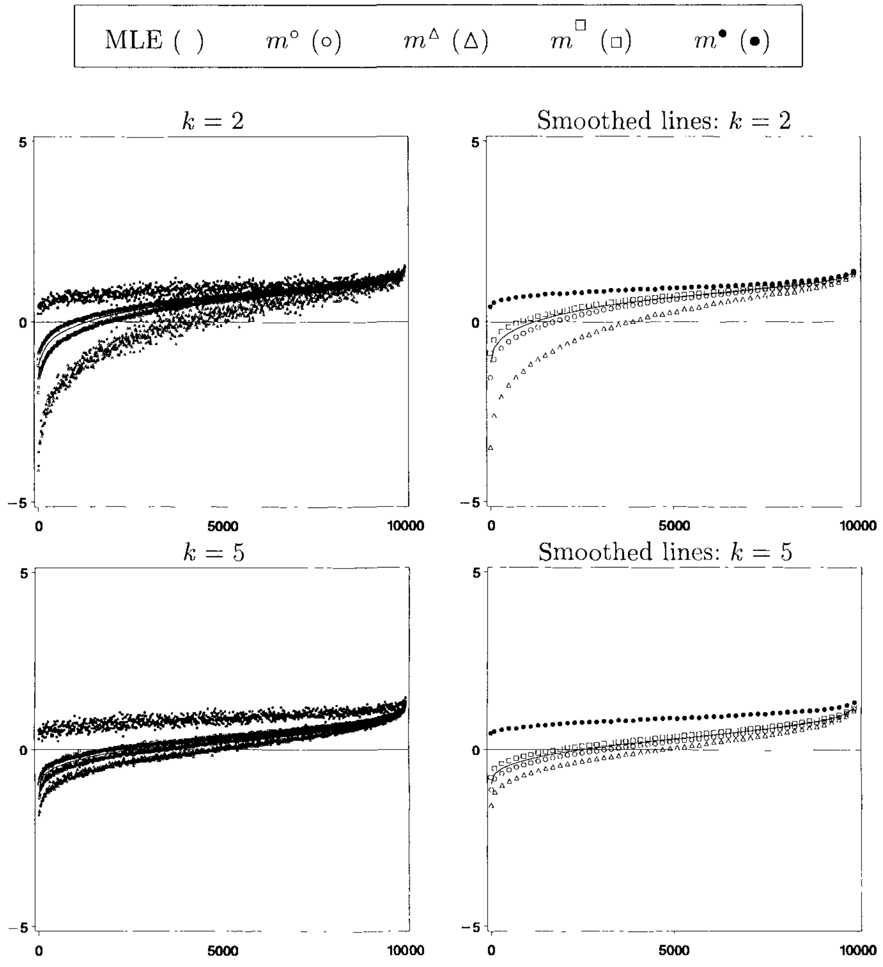


Fig. 11. Scatter plots showing points of $n^{(1/\alpha)}(\hat{m}_j^* - m)$ with the corresponding normalized linear estimators $n^{(1/\alpha)}(\hat{m}_j - m)$ for $j \in \{1,\ 10,\ 20 \ldots 10\,000\}$, where $\hat{m}_j^*$ is the j^{th} smallest MLE from a sample of $R = 10\,000$. Graphs on the right-hand side show the same normalized estimators, each averaged over non-intersecting groups of 200 i.e. $j \in \{(1 \ldots 200), \ldots, (9\,801 \ldots 10\,000)\}$ and plotted at the midpoints of each group

Scatter Plots of the Estimators, $\alpha = 10$, $n = 500$

MLE (-)	m° (○)	$m^{\triangle}$ (△)	$m^{\square}$ (□)	$m^{\bullet}$ (●)

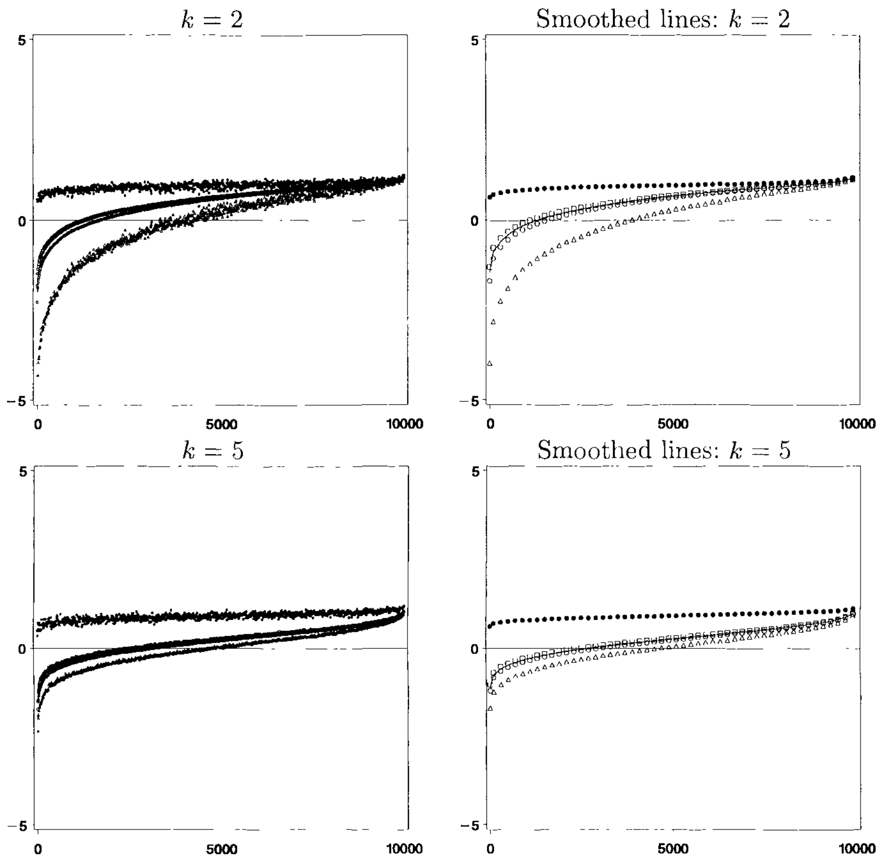


Fig. 12. Scatter plots showing points of $n^{(1/\alpha)}(\hat{m}_j^* - m)$ with the corresponding normalized linear estimators $n^{(1/\alpha)}(\hat{m}_j - m)$ for $j \in \{1,\ 10,\ 20 \ldots 10\,000\}$, where $\hat{m}_j^*$ is the j^{th} smallest MLE from a sample of $R = 10\,000$. Graphs on the right-hand side show the same normalized estimators, each averaged over non-intersecting groups of 200 i.e. $j \in \{(1 \ldots 200), \ldots, (9\,801 \ldots 10\,000)\}$ and plotted at the midpoints of each group

Scatter Plots of MLE Against the Optimal Linear Estimator

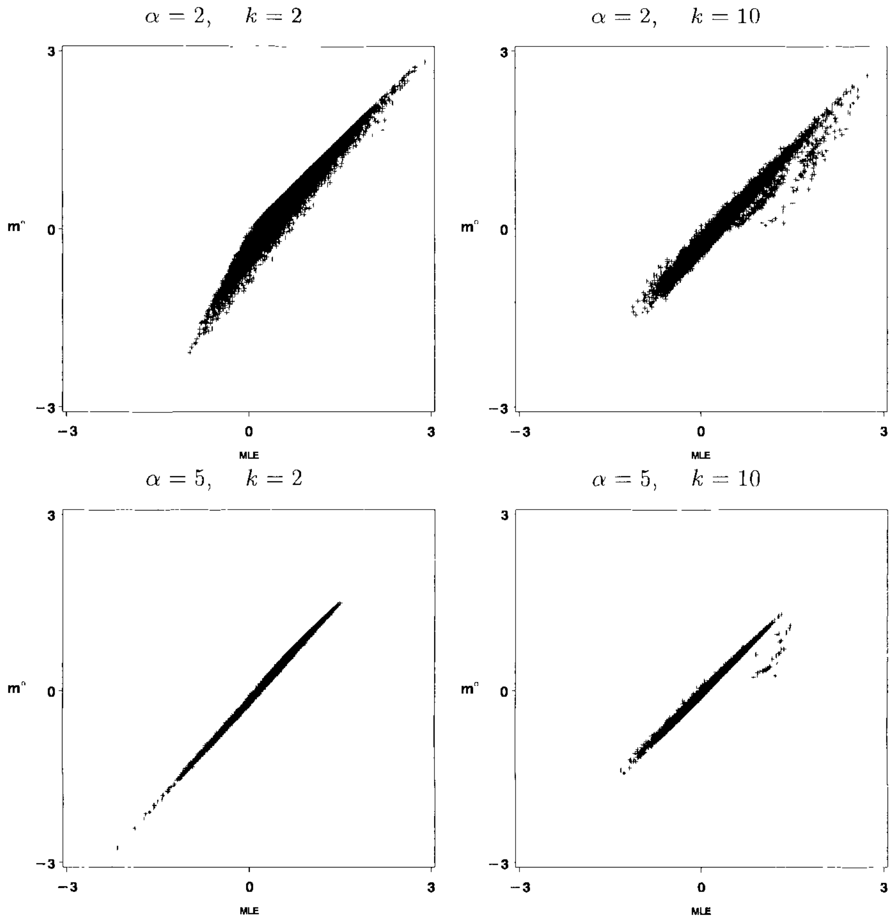


Fig. 13. Scatter plots of $R = 10\,000$ points of $n^{(1/\alpha)}(\hat{m}_i - m)$ for MLE plotted against $n^{(1/\alpha)}(m_i^\circ - m)$ (the optimal linear estimator). Here $n = 500$

[Zhi79] Zhigljavsky, A.: PhD: Monte-Carlo methods in global optimisation. University of St.Petersburg, St.Petersburg (1979)

[Zhi91] Zhigljavsky, A.: Theory of global random search. Kluwer Academic Publishers, Dordrecht (1991)

[ZZ06] Zhigljavsky, A., Zilinskas, A.: Stochastic Global Optimization. Springer, N.Y. et al (2006) to appear

Multi-particle Simulated Annealing*

Orcun Molvalioglu[1], Zelda B. Zabinsky[2], and Wolf Kohn[3]

[1] Industrial Engineering Program, University of Washington, Seattle, WA 98195-2650, USA orcun@u.washington.edu
[2] Industrial Engineering Program, University of Washington, Seattle, WA 98195-2650, USA zelda@u.washington.edu
[3] Clearsight Systems Inc., 3655 131st Avenue SE Suite 602, Bellevue, WA 98006-1390, USA wolf.kohn@clearsightsystems.com

Summary. Whereas genetic algorithms and evolutionary methods involve a population of points, simulated annealing (SA) can be interpreted as a random walk of a single point inside a feasible set. The sequence of locations visited by SA is a consequence of the Markov Chain Monte Carlo sampler. Instead of running SA with multiple independent runs, in this chapter we study a multi-particle version of simulated annealing in which the population of points interact with each other. We present numerical results that demonstrate the benefits of these interactions on algorithm performance.

1 Introduction

Simulated annealing (SA) is a Monte Carlo method introduced by Kirkpatrick et al. [KGV83] for solving global optimization problems. It can be considered as the random walk of a single particle inside the feasible set. This random walk is generated by a nonhomogeneous Markov chain which is controlled by a temperature parameter. As the temperature goes to zero, under various conditions and assumptions, the random walk of the particle asymptotically converges to the global optima; for example see [HS88, Loc00, MM99, NP95, RS94].

Essential to the SA methodology is the tradeoff between the success probability of the algorithm and convergence. Most of the convergence results require the assumption of sufficiently slow cooling of the temperature so that the transition distribution of the random walk can reach its equilibrium distribution which is the Gibbs measure at some temperature. On the other hand fast cooling schedules reduce the run time but also fail to guarantee asymptotic convergence to the optima, i.e. the algorithm can get stuck in a local

* This research was supported in part by National Science Foundation under the grant DMI-0244286. Authors would also like to thank Clearsight Systems for supporting this research.

optima and the probability that it can escape when it is in such a situation can be sufficiently small. In this case one way to speed up the process is to terminate the run and make new restarts. When these restarts are independent the success rate only increases linearly with the number of runs. A common methodology to increase the success rate even further is to impose interactions between the restarts instead of making them independent and as studied by Sadeh et al. [SNT97] this makes the overall process more effective and efficient.

Making multi-restarts can also be viewed as having multi-particles instead of a single particle. When the restarts are independent this is equivalent to having non interacting particles and when there is dependence between the restarts we can say particles are interacting. Aldous and Vazirani [AV94] study an algorithm with multiple particles which are searching for the deepest vertex in a rooted tree by random walks. The particles interact with a simple scheme called "go with the winners" which classifies the particles as losers and winners. The interaction involves reassigning the loser particles to the position of the winners and the search continues on from the new locations. Their mathematical results indicate that this method can boost the success probability by more than a linear multiple of the number of particles. A similar idea appears in [Mor04] as an interacting Metropolis algorithm. The particles interact through a selection process and some particles are randomly assigned to the location of the other particles just like in the "go with the winners" scheme. Both of these interacting particle algorithms are similar to genetic algorithms or evolutionary methods. They have a mutation and selection mechanism.

In this chapter we numerically study an annealing algorithm with multiple particles. The particles interact through a selection mechanism. This interaction can be viewed as bridging the gap between simulated annealing and genetic algorithms. The algorithm is still an annealing type algorithm as the underlying concept is still to converge to the Gibbs measure at some temperature. In the next section we introduce the algorithm and then present numerical results illustrating the performance of the algorithm. The final section concludes the chapter and discusses further research steps.

2 The Multi-particle Simulated Annealing Algorithm

Throughout the rest of the chapter we consider an optimization program in the following general form,

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in S \end{aligned}$$

where x is an n-dimensional vector, S is a subset of $\Re^n$, and $f : S \to \Re$ is a measurable real-valued objective function defined over S. We denote the set of global optima by S^* and let x^* denote one of the global optima. We use y_* to denote the global minimum value.

As stated in the introduction the underlying idea of the simulated annealing algorithm is to converge to the Gibbs measure μ_T, at some temperature T, where

$$\mu_T := \frac{\exp(-f(x)/T)}{\int_S \exp(-f(y)/T)\mu_0(dy)}\mu_0$$

and μ_0 is a probability measure on S that charges every point. The traditional way to approximate this measure is to use the Markov chain Monte Carlo (MCMC) method. It samples candidate points using a Markov kernel $K(x,d\xi)$ and accepts/rejects them according to the Metropolis ratio, $G(x,z,T) := \exp(-(f(x)-f(z))/T)$. As in [Mor04], the transition probabilities of the MCMC methodology can be expressed in the following compact format,

$$M(x_i,d\xi_i,T) := (1 \wedge G(x,\xi,T))K(x,d\xi) + \left(1 - \int_S 1 \wedge G(x,z,T))K(x,dz)\right)\delta_x(d\xi).$$

This method can be interpreted as a single particle moving in S according to the Markov transitions $M(x,.,T)$. One crucial issue is the possibility of getting stuck at a point $x \in S \setminus S^*$ for a long time. This may happen when there is a low Metropolis ratio between x and the points that are easily accessible from x by the Markov kernel $K(x,.)$. The likelihood of getting stuck depends directly on the cooling schedule, the Markov kernel, and the structure of the objective function.

One way to reduce the possibility of getting stuck is to use multiple particles instead of a single particle. This increases the probability of finding points with high Metropolis ratios. Below we introduce the multi-particle simulated annealing algorithm which is motivated by the interacting Metropolis algorithm in [Mor04]. The algorithm initially samples N independent points in the feasible region. The set X denotes the current set of points and the set Y contains the corresponding function values. In every iteration we use the Markov kernel to sample candidate points Ξ from the points in X, and create a new set X by randomly selecting candidate points according to their Metropolis ratio. We interpret this as moving the particles randomly based on the Metropolis ratio.

Multi-Particle Simulated Annealing (MPSA):

Step 0. Set $k = 0$. Generate N uniformly distributed points x_i on S, $i = 1,\dots,N$. Let $X_0=\{x_1,x_2,\dots,x_N\}$, $Y_0 = \{f(x_1),f(x_2),\dots,f(x_N)\}$. Set $T_0 = \tau(X_0,f(.),0)$.

Step 1. For $i = 1,\dots,N$ sample a candidate point $\xi_i \in S$ with probability distribution $K_i(x_i,d\xi_i)$. Set $\Xi_k = \bigcup_i \xi_i$.

Step 2. Check the stopping criteria $S(X_k,Y_k,k)$. If the criteria is not met, set $X_{k+1} = Q(X_k,\Xi_k,T_k)$, update Y_{k+1}, $T_{k+1} = \tau(X_{k+1},f(.),k+1)$ and the incumbent solution. Increment k and return to Step 1. If the criteria is met, stop.

The annealing (cooling) schedule is the $\tau(X_k, f(.), k)$ function. We allow it to be a function of several variables but in our numerical experiments our choice of annealing schedule does not depend on all of them.

The overall process of moving points in the feasible set has two main stages. We first sample candidate points in *Step 1* using the probability distribution $K_i(x_i, d\xi_i)$. This can be interpreted as the mutation stage. It is possible to allow the Markov kernel to interact during mutation, but in our numerical results we use a Markov kernel that is identical for all particles and independent of each other.

Next in *Step 2* we select our new points using a function $Q(X_k, \Xi_k, T_k)$. We interpret this as moving each point in X to a new location with a probability that depends on the current points, candidate points and the current temperature. With probability $\varepsilon_k G(x_i, \xi_i, T_k)$ we move the particle to its candidate point ξ_i and with probability $1 - \varepsilon_k G(x_i, \xi_i, T_k)$ we reject it and move the particle to a randomly chosen candidate point from the set Ξ_k, i.e.

$$x_i := \begin{cases} \xi_i \text{ with probability } \varepsilon_k G(x_i, \xi_i, T_k) \\ \xi_j \text{ with probability } (1 - \varepsilon_k G(x_i, \xi_i, T_k)) \left(\frac{G(x_j, \xi_j, T_k)}{\sum_{l=1}^{N} G(x_l, \xi_l, T_k)} \right), \ j \in \overline{(1, N)} \end{cases}$$

where the parameter ε_k is chosen so that the probabilities stated above are normalized. One way to choose it is to set $\varepsilon_k = \max_j G(x_j, \xi_j, T_k)$.

This selection mechanism favors candidate points with high Metropolis ratios. When a particle i has the maximum Metropolis ratio in the population we definitely move that particle to its candidate location ξ_i, using our choice of ε_k. Otherwise with some probability we may reject ξ_i and randomly choose a new point from the set Ξ_k of all candidate points with probability proportional to the candidate point's Metropolis ratio.

Finally the stopping criteria of the algorithm is imposed by the function $S(.)$. The stopping criteria can be triggered by reaching a small temperature or reaching a desired function value. We can also stop if we do not see an improvement for a maximum number of attempts.

3 Numerical Results

We have implemented the MPSA algorithm to minimize the sinusoidal function [Zab03] over box constraints, i.e.

$$\begin{array}{ll} \text{minimize} & -2.5 \prod_{i=1}^{n} \sin(x_i - z) - \prod_{i=1}^{n} \sin(5(x_i - z)) \\ \text{subject to} & x \in S := [0, 180]^n \end{array}$$

where x is in degrees. The parameter z controls the location of the global optima. We set $z = 0$ and $z = 60$. We call the first choice, the *centered* sinusoidal problem as the global optimum is located at the center of S, i.e. $x^* = [90, 90, ..., 90]$ with $y_* = -3.5$. The second choice is referred as the *shifted* sinusoidal problem as the location of the global optimum is shifted to $x^* = [150, 150, ..., 150]$ with $y_* = -3.5$. For both types of problems we set the problems' dimension to $n = 5$, 10 and 20. Thus overall we have a set of six problems with different structure and dimensions.

In MPSA we make the particles interact only through the function $Q(.)$. We set the population sizes as $N = 50$, $N = 100$ and $N = 250$ for dimensions $n = 5$, 10 and 20 respectively. The Markov kernel $K(x, d\xi)$ that we used for the sampling distribution of each particle is the Hit-and-Run generator [Smi84]. Hit-and-Run has been used in a simulated annealing context with positive results (see [Zab03]).

The cooling schedule is set to $(1 + k)^{-0.99}$ where k is the iteration count. With this cooling schedule when the number of particles N is sufficiently large, the Markov process generated by MPSA asymptotically converges to the Gibbs measure at some temperature (see [Mor04]).

The stopping criteria, $S(.)$, checks the maximum number of function evaluations before an improvement in the incumbent solution. We terminate a run if we do not see an improvement in 5,000 function evaluations.

We also implemented the classical SA which is a MCMC method on our problem set with the same cooling schedule, Hit-and-Run sampling distribution and stopping criteria as in MPSA. For both algorithms we made 40 independent runs on each problem. In Table 1 we report the average success rate over all these runs. We consider a run to be successful if it reaches the ϵ-optimum where we set ϵ to be 0.01. The value of ϵ was chosen so that a successful point is within approximately 1% of the range of function values over the set S. Thus if during the run the algorithm visits an $\hat{x} \in S$ such that $f(\hat{x}) \in [y_*, y_* + \epsilon]$ with $\epsilon := 0.01$, we consider that run a success.

Table 1. Success Rate over 40 Runs

	Centered sinusoidal problem			Shifted sinusoidal problem		
	n=5	n=10	n=20	n=5	n=10	n=20
MPSA	100%	100%	100%	100%	100%	100%
SA	100%	95%	83%	90%	60%	25%

As in Table 1, the MPSA algorithm was successful in every run. SA on the other hand had a lower success rate on higher dimensional problems. For the 20 dimensional shifted problem, which is our hardest test problem, SA could only find the global minimum 25% of the time.

One way to increase the effectiveness of SA, as mentioned in the introduction, is to make restarts. In Fig. 1 we plot the incumbent solution versus the number of function evaluations for SA with independent restarts applied to solve the 20 dimensional centered sinusoidal problem. One run of SA with restarts consists of executing the classical SA with several restarts, where a restart is initiated when there is no improvement in the incumbent solution after 5,000 function evaluations. The number of restarts per run varied, as we kept on restarting until we had a successful run, i.e. hit the ϵ-optimum.

For comparison purposes, we made 40 runs of MPSA applied to the same centered sinusoidal problem in 20 dimensions. Figure 2 illustrates the progress of the 40 MPSA runs. Comparing Figs. 1 and 2 shows that SA with restarts has a larger variation in number of function evaluations than MPSA. While SA with independent restarts could sometimes achieve less function evaluations than MPSA, sometimes it required more.

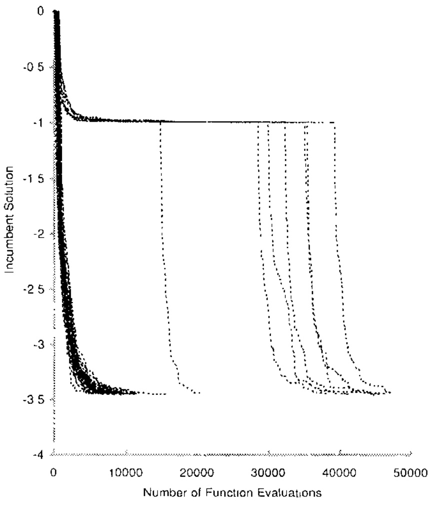


Fig. 1. Incumbent solution vs. the number of function evaluations of SA with restarts on the centered sinusoidal problem with $n = 20$

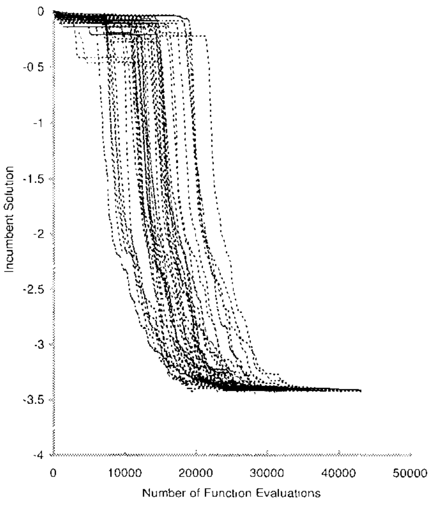


Fig. 2. Incumbent solution vs. the number of function evaluations of MPSA on the centered sinusoidal problem with $n = 20$

We performed a similar comparison between SA with restarts and MPSA on the shifted sinusoidal problem in 20 dimensions. SA with restarts showed even more variation in its performance as we can observe from Fig. 3. However, MPSA was more robust, and its results represented in Fig. 4 on the shifted problem were similar to its results on the centered problem. Notice that the scale on function evaluations in Fig. 3 goes to 170,000, whereas the number of function evaluations in the other three figures stops at 50,000. The vertical line in Fig. 3 at 50,000 indicates the point where the other algorithms had already achieved success. It appears that the interaction between points in MPSA makes the algorithm less sensitive to problem structure than SA with independent restarts.

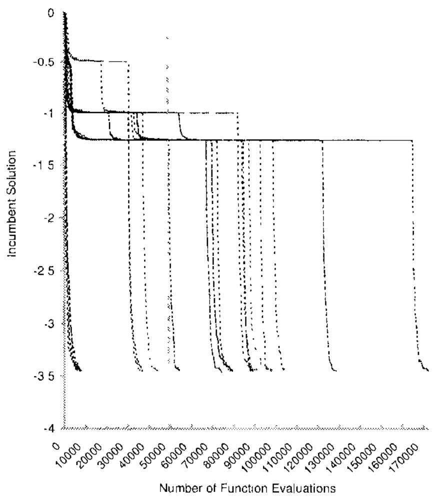


Fig. 3. Incumbent solution vs. the number of function evaluations of SA with restarts on the shifted sinusoidal problem with $n = 20$

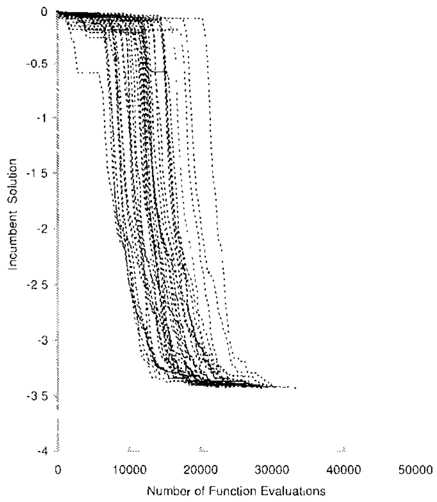


Fig. 4. Incumbent solution vs. the number of function evaluations of MPSA on the shifted sinusoidal problem with $n = 20$

4 Conclusions

In this chapter we numerically study a multi-particle version of simulated annealing algorithm which is an interacting Metropolis algorithm as it appears in [Mor04]. The MPSA algorithm achieved successful results on all test problems while SA had poor success rate on difficult cases. In order to increase the success of SA we included independent restarts of the algorithm. While this increased the effectiveness of the algorithm, the efficiency demonstrated large variation. One way to deal with the latter issue is to introduce dependencies on the restarts as in [SNT97], and the selection mechanism of MPSA is very similar to the interaction mechanism of restarts appearing in that study. Making dependent restarts and having interacting particles are analogous to each other. The benefit of a multi-particle point of view is that the progress of the algorithm can easily be expressed as a stochastic process which brings benefits in theoretical analysis. In fact, when the population size increases, the path measures of the particles in MPSA asymptotically converge to Feynmann-Kac path measures associated with the potential/kernel pair given by $G(x,.,T)/K(x,.)$, c.f. [Mor04]. With some error one may directly use these path measures to claim something about the behavior of the algorithm and the number of particles can be determined based on how much we want to approximate these measures. Our future research involves studying these ideas and exploring ways to use the information generated by the particles to approximate the particles' possible paths.

References

[AV94] Aldous, D., Vazirani, U.: Go with the winners algorithms. Proc. 35th Symp. Foundations of Computer Sci., pp. 492 501 (1994)

[HS88] Holley, R., Stroock, D.: Simulated annealing via Sobolev inequalities. Comm. Math Phys., **115**, 553–569 (1988)

[KGV83] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science, **220**, 671–680 (1983)

[Loc00] Locatelli, M.: Simulated annealing algorithms for continuous global optimization:convergence conditions. Journal of Optimization Theory and Applications, **104**(1), 121–133 (2000)

[MM99] Moral, P.D., Miclo, L.: On the convergence and applications of generalized simulated annealing. Comm. Math Phys., **37**(4), 1222-1250 (1999)

[Mor04] Moral, P.D.: Feynman-Kac Formulae: Genological and Interacting Particle Systems with Applications. Springer-Verlag, New York (2004)

[NP95] Niemiro, W., Pokarowki, P.: Tail events of some nonhomogeneous Markov chains. The Annals of Applied Probability, **5**(1), 261–293 (1995)

[RS94] Romeijn, H.E., Smith, R.L.: Simulated annealing and adaptive search in global optimization. Probability in the Engineering and Informational Science, **8**, 571–590 (1994)

[Smi84] Smith, R.L.: Efficient Monte Carlo procedures for generating points uniformly distributed over bounded region. Operations Research, **32**, 1296–1308 (1984)

[SNT97] Sadeh, N.M., Nakakuki, Y., Thangiah, S.R.: Learning to recognize (un)promising simulated annealing runs:efficient search procedures for job shop scheduling and vehicle routing. Ann. Oper. Res., **75**, 189–208 (1997)

[Zab03] Zabinsky, Z.B.: Stochastic Adaptive Search for Global Optimization. Kluwer, Boston (2003)

On the Goodness of Global Optimisation Algorithms, an Introduction into Investigating Algorithms

Eligius M.T. Hendrix

Operationele Research en Logistiek Groep, Wageningen Universiteit, The Netherlands eligius.hendrix@wur.nl

Summary. An early introductory text on Global Optimisation (GO), [TZ89], goes further than mathematical correctness in giving the reader an intuitive idea about concepts in GO. This chapter extends this spirit by introducing students and researchers to the concepts of Global Optimisation (GO) algorithms. The goal is to learn to read and interpret optimisation algorithms and to analyse their goodness. Before going deeper into mathematical analysis, it is good for students to get a flavour of the difficulty by letting them experiment with simple algorithms that can be followed by hand or spreadsheet calculations. Two simple one-dimensional examples are introduced and several simple NLP and GO algorithms are elaborated. This is followed by some lessons that can be learned from investigating the algorithms systematically.

1 Effectiveness and Efficiency of Algorithms

In this chapter, several criteria are discussed to measure effectiveness and efficiency of algorithms. Moreover, examples are given of basic algorithms that are analysed. This gives an opportunity to introduce in an illustrative way GO concepts such as *region of attraction*, *level set*, *probability of success* and *Performance Graph*. To investigate optimisation algorithms, we start with a definition. An algorithm is a description of steps, preferably implemented into a computer program, which finds an approximation of an optimum point. The aims can be several: reach a local optimum point, reach a global optimum point, find all global optimum points, reach all global and local optimum points. In general, an algorithm generates a series of points x_k that approximate an (or the or all) optimum point. According to the generic description of [TZ89]:

$$x_{k+1} = Alg(x_k, x_{k-1}, ..., x_0, \xi) \tag{1}$$

where ξ is a random variable and index k is the iteration counter. This represents the idea that a next point x_{k+1} is generated based on the information in all former points $x_k, x_{k-1}, ..., x_0$ (x_0 usually being the starting point) and possibly some random effect. This leads to three classes of algorithms discussed here:

- Nonlinear (local) optimisation algorithms, that from a starting point try to capture the "nearest" local minimum point.
- Deterministic GO methods which guarantee to approach the global optimum and require a certain mathematical structure.
- Stochastic GO methods based on the random generation of feasible trial points and nonlinear local optimization procedures.

We will consider several examples illustrating two questions to address to investigate the quality of algorithms (see [BH05]).

- Effectiveness: does the algorithm find what we want?
- Efficiency: what are the computational costs?

Several measurable performance indicators can be defined for these criteria.

1.1 Effectiveness

Consider minimisation algorithms. Focusing on effectiveness, there are several targets a user may have:

1. To discover all global minimum points. This of course can only be realised when the number of global minimum points is finite.
2. To detect at least one global optimal point.
3. To find a solution with a function value as low as possible. relevant for population based algorithms.

The first and second targets are typical satisfaction targets; was the search successful or not? What are good **measures of success**? In the literature, convergence is often used i.e. $x_k \rightarrow x^*$, where x^* is one of the minimum points. Alternatively one observes $f(x_k) \rightarrow f(x^*)$. In tests and analyses, to make results comparable, one should be explicit in the definitions of success. We need not only specify ϵ and/or δ such that

$$||x_k - x^*|| < \epsilon \text{ and/or } \; f(x_k) < f(x^*) + \delta \tag{2}$$

but also specify whether success means that there is an index K such that (2) is true for all $k > K$. Alternatively, success may mean that a record $min_k f(x_k)$ has reached level $f(x^*) + \delta$. Whether the algorithm is effective also depends on the its stochastic nature. When we are dealing with stochastic algorithms, effectiveness can be expressed as the probability that a success has been reached. In analysis, this probability can be derived from sufficient assumptions on the behaviour of the algorithm. In numerical experiments, it can be estimated by counting repeated runs how many times the algorithm converges. We will give some examples of such analysis. In Sect. 2.4 we return to the topic of efficiency and effectiveness considered simultaneously.

1.2 Efficiency

Globally efficiency is defined as the effort the algorithm needs to be successful. A usual indicator for algorithms is the (expected) number of **function evaluations** necessary to reach the optimum. This indicator depends on many factors such as the shape of the test function and the termination criteria used. The indicator more or less suggests that the calculation of function evaluations dominates the other computations of the algorithm. Several other indicators appear in literature.

In nonlinear programming (e.g. [Sca85] and [GMW81]) the concept of **convergence speed** is common. It deals with the convergence limit of the series x_k. Let $x_0, x_1, \ldots, x_k, \ldots$ converge to point x^*. The largest number α for which

$$\lim_{k \to \infty} \frac{||x_{k+1} - x^*||}{||x_k - x^*||^\alpha} = \beta < \infty \tag{3}$$

gives the order of convergence, whereas β is called the convergence factor. In this terminology, the special instances are

- linear convergence with $\alpha = 1$ and $\beta < 1$
- quadratic convergence with $\alpha = 2$ and $0 < \beta < 1$
- superlinear convergence: $1 < \alpha < 2$ and $\beta < 1$, i.e. $\beta = 0$ when using $\alpha = 1$ in (3).

Mainly in deterministic GO algorithms, information on the past evaluations is stored in the computer memory. This requires efficient data handling for looking up necessary information during the iterations. Furthermore, **memory requirements** become a part of the computational burden as retrieving actions cannot be neglected compared to the computational effort due to function evaluations.

In stochastic GO algorithms, an efficiency indicator is the **success rate** defined as the probability that the next iterate is an improvement on the record value found thus far, i.e. $P(f(x_k) < min_{l=1,\ldots,k-1} f(x_l))$. Its theoretical relevance to convergence speed was analysed by [ZS92] and [BMW$^+$95], who showed that a fixed success rate of an effective algorithm (in the sense of so-called uniform covering, see e.g. [HK00]) gives an algorithm with the expected number of function evaluations growing polynomially with the dimension of the problem. However, empirical measurements can only be established in the limit when such an algorithm stabilises, and only for specifically designed test cases [HOG01].

We do not go deeper into theoretical aspects here of performance indicators. Instead some basic algorithms are introduced and analysed. In Sect. 3, systematic investigation of algorithms is expanded upon.

2 Some Basic Algorithms and Their Goodness

2.1 Introduction

In this section, several classes of algorithms are analysed for effectiveness and efficiency. Two test cases are introduced first for which the performance of

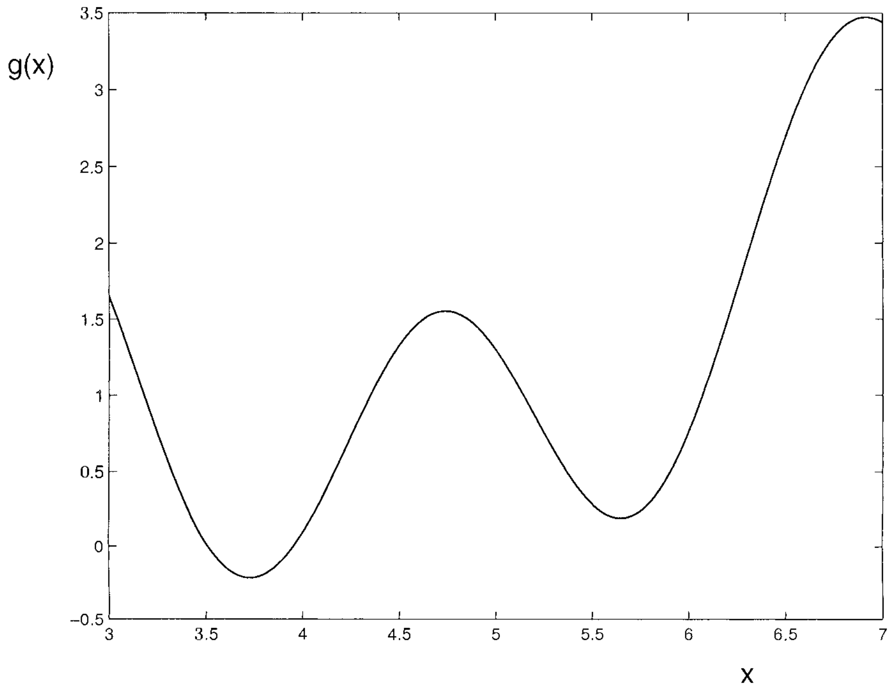


Fig. 1. Test case $g(x)$ with two optima

the algorithms is investigated. We consider the minimisation of the following functions.

$$g(x) = \sin(x) + \sin(3x) + \ln(x), x \in [3, 7] \tag{4}$$

Function g is depicted in Fig. 1 and has three minimum points on the interval. The global minimum is attained at about $x^* = 3.73$, where $f(x^*) = -0.220$. The derivative function is

$$g'(x) = \cos(x) + 3\cos(3x) + \frac{1}{x} \tag{5}$$

on the interval $[3, 7]$. Alternatively to function g, we introduce a function h with more local minimum points by adding to function g a bubble function based on $\mathrm{frac}(x) = x - \mathrm{round}(x)$ where $\mathrm{round}(x)$ rounds x to the nearest integer. Now the second case is defined as

$$h(x) = g(x) + 1.5\mathrm{frac}^2(4x) \tag{6}$$

In Fig. 2, the graph of function h is shown. It has 17 local minimum points

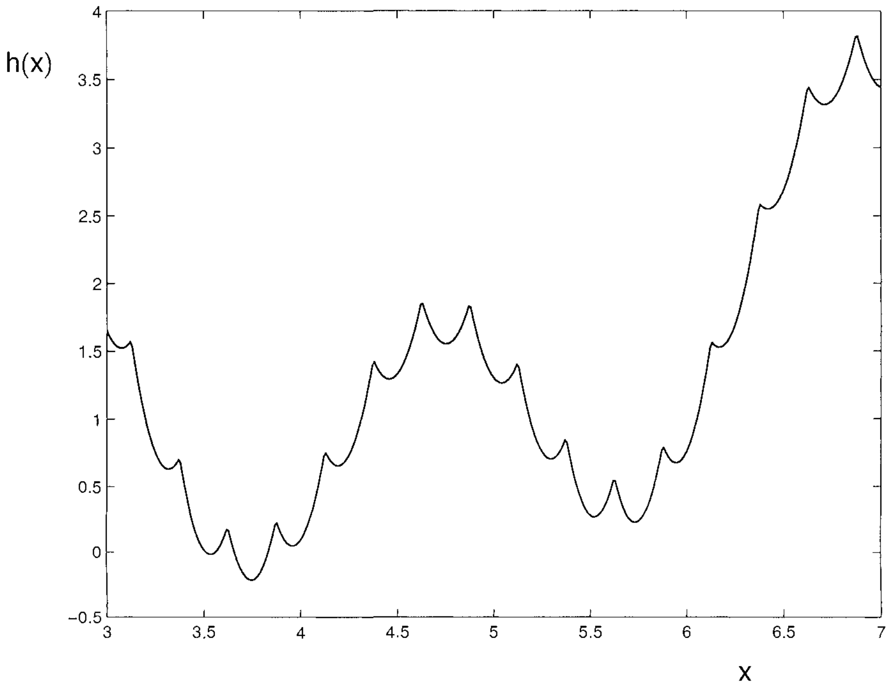


Fig. 2. Test case $h(x)$ with 17 optima

on the interval $[3, 7]$. Although g nor h are convex on the interval, at least function h is piecewise convex on the intervals in between the points of $\mathbb{S} = \{x = \frac{1}{4}k + \frac{1}{8}, k \in \mathbb{Z}\}$. At these points, h is not differentiable. For the rest of the interval one can define the derivative

$$h'(x) = g'(x) + 12 \times \mathrm{frac}(4x) \quad \text{for} \quad x \notin \mathbb{S} \tag{7}$$

The global minimum point of h on $[3, 7]$ is shifted slightly compared to g towards $x^* = 3.75$, where $f(x^*) = -0.217$.

In the following Sections, we will test algorithms on their ability to find minima of these two functions. One should set a target on what is considered an acceptable or successful result. For instance one can aim at detecting a local minimum or detecting the global minimum. For the neighbourhood we

```
Set $k = 0$, $l_0 = l$ and $r_0 = r$
while $(r_k - l_k > \epsilon)$
    $x_k = \frac{l_k + r_k}{2}$
    if $f'(x_k) < 0$
        $l_{k+1} = x_k$ and $r_{k+1} = r_k$
    else
        $l_{k+1} = l_k$ and $r_{k+1} = x_k$
    $k = k + 1$
End while
```

Algorithm 1: Bisect($[l, r], f, \epsilon$)

will take an acceptance of $\epsilon = 0.01$. For determining an acceptable low value of the objective function we take $\delta = 0.01$. Notice that ϵ represents 0.25% of the argument range and δ is about 0.25% of the function values range.

2.2 NLP Local Optimisation: Bisection and Newton

Two nonlinear programming algorithms are sketched and their performance measured for the two test cases. First the bisection algorithm is considered.

The algorithm departs from a starting interval $[l, r]$ that is halved iteratively based on the sign of the derivative in the midpoint. This means that the method is only applicable when the derivative is available at the generated midpoints. The point x_k converges to a minimum point within the interval $[l, r]$. If the interval contains only one minimum point, it converges to that. In our test cases, several minima exist and one can observe the convergence to one of them. The algorithm is effective in the sense of converging to a local

Table 1. Bisection for functions g and h, first 6 iterations

	function g					function h				
k	l_k	r_k	x_k	$g'(x_k)$	$g(x_k)$	l_k	r_k	x_k	$h'(x_k)$	$h(x_k)$
0	3.00	7.00	5.00	-1.80	1.30	3.00	7.00	5.00	-1.80	1.30
1	5.00	7.00	6.00	3.11	0.76	5.00	7.00	6.00	3.11	0.76
2	5.00	6.00	5.50	-1.22	0.29	5.00	6.00	5.50	-1.22	0.29
3	5.50	6.00	5.75	0.95	0.24	5.50	6.00	5.75	0.95	0.24
4	5.50	5.75	5.63	-0.21	0.20	5.50	5.75	5.63	-6.21	0.57
5	5.63	5.75	5.69	0.36	0.20	5.63	5.75	5.69	-2.64	0.29
6	5.63	5.69	5.66	0.07	0.19	5.69	5.75	5.72	-0.85	0.24

(nonglobal) minimum point for both cases. Another starting interval could have lead to another minimum point. In the end, we are certain that the current iterate x_k is not further away than ϵ from a minimum point. Many

```
Set $k = 0$,
while ($| f'(x_k) | > \alpha$)
    $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
            ! safeguard for staying in interval
        if $x_{k+1} < l$, $x_{k+1} = l$
        if $x_{k+1} > r$, $x_{k+1} = r$
        if $x_{k+1} = x_k$, STOP
    $k = k + 1$
End while
```

Algorithm 2: Newt$([l, r], x_0, f, \alpha)$

other stopping criteria like convergence of function values or derivatives going to zero could be used. The current stopping criterion is easy for analysis of efficiency. One question could be: How many iterations (i.e. corresponding (derivative) function evaluations) are necessary to come closer than ϵ to a minimum point. The bisection algorithm is a typical case of linear convergence with a convergence factor of $\frac{1}{2}$, $\frac{|r_{k+1}-l_{k+1}|}{|r_k-l_k|} = \frac{1}{2}$. This means one can determine the number of iterations necessary for reaching ϵ-convergence:

$$| r_k - l_k | = (\tfrac{1}{2})^k \times | r_0 - l_0 | < \epsilon \Rightarrow$$
$$(\tfrac{1}{2})^k < \frac{\epsilon}{|r_0 - l_0|} \Rightarrow k > \frac{\ln \epsilon - \ln|r_0 - l_0|}{ln \frac{1}{2}}$$

The example case requires at least 9 iterations to reach an accuracy of $\epsilon = 0.01$.

An alternative for finding the zero point of an equation, in our case the derivative, is the so-called method of **Newton**. The idea is that its efficiency is known to be superlinear (e.g. [Sca85]), so it should be faster than bisection. We analyse its efficiency and effectiveness for the two test cases. In general, the aim of the Newton algorithm is to converge to a point where the derivative is zero. Depending on the starting point x_0, the method may converge to a minimum or maximum. Also it may also not converge at all, for instance when a minimum point does not exist. Specifically in the version of Algorithm 2, a safeguard is built in to ensure the iterates remain in the interval; it can converge to a boundary point. If x_0 is in the neighbourhood of a minimum point where f is convex, then convergence is guaranteed and the algorithm is effective in the sense of reaching a minimum point. Let us consider what happens for the two test cases. When choosing the starting point x_0 in the middle of the interval $[3, 7]$, the algorithm converges to the closest minimum point for function h and to a maximum point for the function g, i.e. it fails for this starting point. This gives rise to introducing the concept of a **region of attraction** of a minimum point x^*. A region of attraction of point x^*, is the region of starting points x_0 where the local search procedure converges to point x^*. We elaborate this concept further in Sect. 2.4.

Table 2. Newton for functions g and h, $\alpha = 0.001$

	function g				function h			
k	x_k	$g'(x_k)$	$g''(x_k)$	$g(x_k)$	x_k	$h'(x_k)$	$h''(x_k)$	$h(x_k)$
0	5.000	-1.795	-4.934	1.301	5.000	-1.795	43.066	1.301
1	4.636	0.820	-7.815	1.511	5.042	0.018	43.953	1.264
2	4.741	-0.018	-8.012	1.553	5.041	0.000	43.944	1.264
3	4.739	0.000	-8.017	1.553	5.041	0.000	43.944	1.264

One can observe here when experimenting further, that when x_0 is close to a minimum point of g, the algorithm converges to one of the minimum points. Morever, notice now the effect of the safeguard to keep the iterates in the interval $[3, 7]$. If for instance $x_{k+1} < 3$, it is forced to a value of 3. In this way, also the left point $l = 3$ is an attraction point of the algorithm. Function h is piecewise convex, such that the algorithm always converges to the closest minimum point.

2.3 Deterministic GO: Grid Search, Piyavskii-Shubert

The aim of deterministic GO algorithms is to approach the optimum with a given certainty. We sketch two algorithms for the analysis of effectiveness and efficiency. The idea of reaching the optimum with an accuracy of ϵ can be done by so-called "everywhere dense sampling", as introduced in literature on Global Optimisation (see e.g.[TZ89]). In a rectangular domain this can be done by constructing a grid with a mesh of ϵ. By evaluating all points on the grid, the best point found is a nice approximation of the global minimum point. The difficulty of GO is, that even this best point found may be far away from the global minimum point, as the function may have a needle shape in another region in between the grid points. As shown in literature, one can always construct a polynomial of sufficiently high degree, that fits all the evaluated points and has a minimum point more than ϵ away from the best point found. Actually, grid search is theoretically not effective if no further assumptions are posed on the optimisation problem to be solved.

Let us have a look at the behaviour of the algorithm for our two cases. For the ease of the formulation we write down the grid algorithm for one dimensional functions. The algorithm starts with the domain $[l, r]$ written as an interval and generates $M = \lceil (r - l)/\epsilon \rceil + 1$ grid points, where $\lceil x \rceil$ is the lowest integer greater than or equal to x. Experimenting with test functions g and h gives reasonable results for $\epsilon = 0.01$, ($M = 401$) and $\epsilon = 0.1$, ($M = 41$). In both cases one finds an approximation x^U less than ϵ from the global minimum point. One knows exactly how many function evaluations are required to reach this result in advance.

The efficiency of the algorithm in higher dimensions is also easy to establish. Given the lower left vector l and upper right vector r of a rectangular domain, one can determine easily how many grid co-ordinates $M_j, j = 1, \ldots, n$

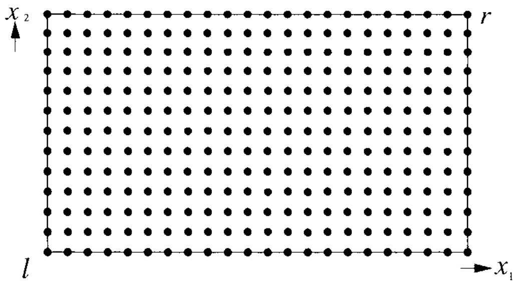


Fig. 3. Equidistant grid over rectangular feasible set

```
$M = \lceil (r-l)/\epsilon \rceil + 1$, $f^U = \infty$
for ($k = 1$ to $M$) do
    $x_k = l + \frac{(k-1)\times(r-l)}{M-1}$
    if $f(x_k) < f^U$
        $f^U = f(x_k)$ and $x^U = x_k$
End for
```

Algorithm 3: Grid($[l, r], f, \epsilon$)

in each direction should be taken and the total number of grid points is $\prod_j M_j$. This number is growing exponentially in the dimension n. As mentioned before, the effectiveness is not guaranteed in the sense of being closer than ϵ from a global minimum point, unless we make an assumption on the behaviour of the function. A usual assumption in the literature is Lipschitz continuity.

Definition 1. *L is called a Lipschitz constant of f on X if:*

$$| f(x) - f(y) | \leq L\|x - y\|, \quad \forall x, y \in X$$

In a practical sense it means that big jumps do not appear in the function value; slopes are bounded. With such an assumption, the δ-accuracy in the function space translates into an ϵ-accuracy in the x-space. Choosing $\epsilon = \delta/L$ gives that the best point x^U is in function value finally close to minimum point x^*:

$$| f^U - f^* | \leq L\|x^U - x^*\| \leq L\epsilon = \delta \tag{8}$$

In higher dimension, one should be more exact in the choice of the distance norm $\|\cdot\|$. Here, for the one-dimensional examples we can focus on deriving the accuracy for our cases in a simple way. For a one-dimensional differentiable function f, L can be taken as

$$L = \max_{x \in X} | f'(x) | \tag{9}$$

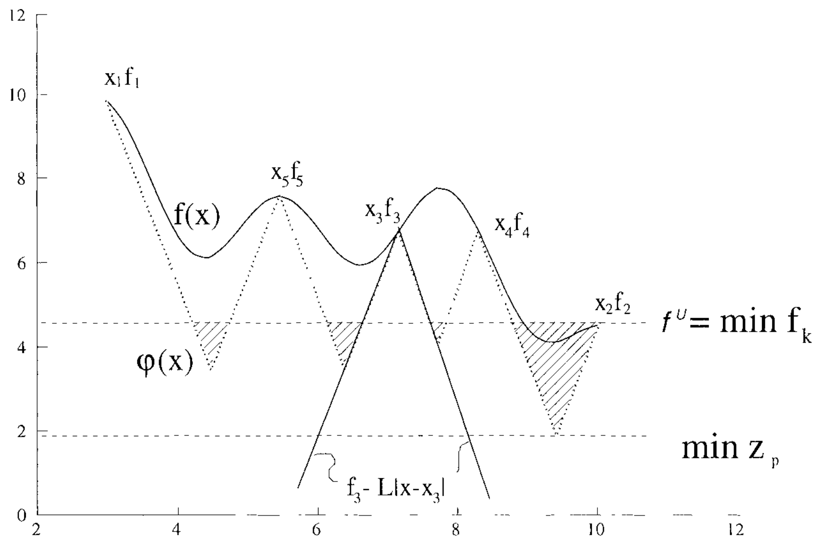


Fig. 4. Piyavskii-Shubert algorithm

Using (9), one can now derive valid estimates for the example functions h and g. One can derive an over estimate L_g for the Lipschitz constant of g on $[3, 7]$ as

$$\begin{aligned}&\max_{x\in[3,7]} \mid g'(x) \mid = \max_{x\in[3,7]} \mid \cos(x) + 3\cos(3x) + \tfrac{1}{x} \mid \leq \\ &\max_{x\in[3,7]}\{\mid \cos(x) \mid + \mid 3\cos(3x) \mid + \mid \tfrac{1}{x} \mid\} \leq \\ &\max_{x\in[3,7]} \mid \cos(x) \mid + \max_{x\in[3,7]} \mid 3\cos(3x) \mid + \max_{x\in[3,7]} \mid \tfrac{1}{x} \mid = \\ &1 + 3 + \tfrac{1}{3} = L_g\end{aligned} \tag{10}$$

The estimate of L_h based on (7) is done by adding the maximum derivative of the bubble function $12 \times \frac{1}{2}$ to L_g for illustrative purposes rounded down to $L_h = 10$. We can now use (8) to derive a guarantee for the accuracy. One arrives certainly closer than $\delta = 0.01$ to the minimum in function value by taking for function g a mesh size of $\epsilon = \frac{0.01}{4.33} = 0.0023$ and for function h taking $\epsilon = 0.001$. For the efficiency of grid search this means that reaching the δ-guarantee requires the evaluation of $M = 1733$ points for function g and $M = 4001$ points for function h. Note, that due to the one dimensional nature of the cases, ϵ can be taken twice as big, as the optimum point x^* is not further than half the mesh size from an evaluated point.

The main idea of most deterministic algorithms is not to generate and evaluate points everywhere dense, but to throw out those regions, where the optimum cannot be situated. Giving a Lipschitz constant, independently **Piyavskii and Shubert** constructed similar algorithms, see [Shu72] and [DP67]. From the point of view of the graph of the function f to be minimised and an evaluated point (x_k, f_k), one can say that the region described

```
Set $p = 1$, $l_1 = l$ and $r_1 = r$, $\Lambda = \{[l_1, r_1]\}$
$z_1 = \frac{f(l_1)+f(r_1)}{2} - \frac{L(r_1-l_1)}{2}$, $f^U = \min\{f(l), f(r)\}$, $x^U = argmin\{f(l), f(r)\}$
while ($\Lambda \neq \emptyset$)
    remove an interval $[l_k, r_k]$ from $\Lambda$ with $z_k = \min_p z_p$
    evaluate $f(m_k) = f(\frac{f(l_k)-f(r_k)}{2L} + \frac{r_k+l_k}{2})$
    if $f(m_k) < f^U$
        $f^U = f(m_k)$, $x^U = m_k$ and remove all $C_p$ from $\Lambda$ with $z_p > f^U - \delta$
    split $[l_k, r_k]$ into 2 new intervals $C_{p+1} = [l_k, m_k]$ and$C_{p+2} = [m_k, r_k]$
        with corresponding lower bounds $z_{p+1}$ and $z_{p+2}$
    if $z_{p+1} < f^U - \delta$ store $C_{p+1}$ in $\Lambda$
    if $z_{p+2} < f^U - \delta$ store $C_{p+2}$ in $\Lambda$
    $p = p + 2$
End while
```

Algorithm 4: PiyavShub($[l, r], f, L, \delta$)

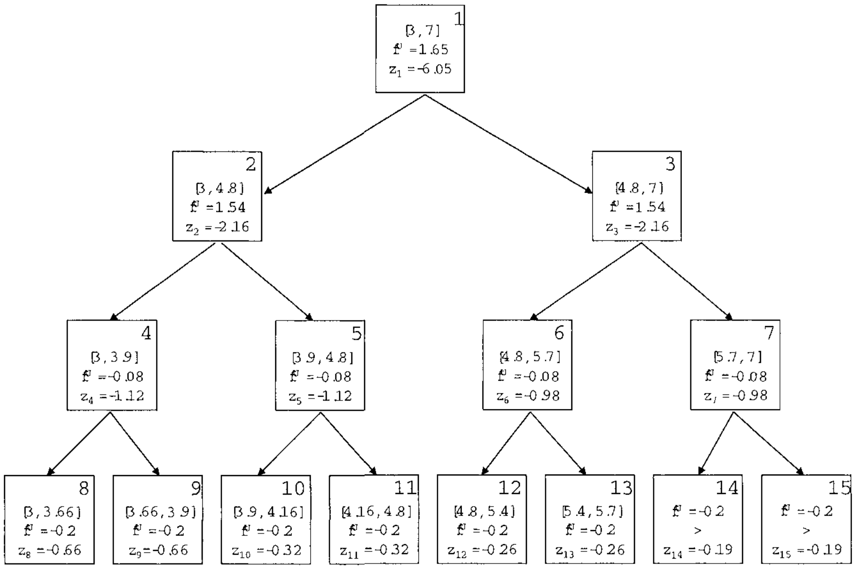
by $x, f < f_k - L \mid x - x_k \mid$ cannot contain the optimum; the graph is above the function $f_k - L \mid x - x_k \mid$. Given a set of evaluated points $\{x_k\}$, one can construct a lower bounding function, a so-called saw-tooth underestimator that is given by $\varphi(x) = \max_k(f_k - L \mid x - x_k \mid)$ as illustrated by Fig. 4. Given that we have also an upper bound f^U on the minimum of f being the best function value found thus far, one can say that the minimum point has to be in one of the shaded areas. We will describe here the algorithm from a Branch-and-Bound point of view, where the subsets are defined by intervals $[l_p, r_p]$ and the end points are given by evaluated points. The index p is used to represent the intervals in Λ. For each interval, a lower bound is given by

$$z_p = \frac{f(l_p) + f(r_p)}{2} - \frac{L(r_p - l_p)}{2} \tag{11}$$

The gain with respect to grid search is that an interval can be thrown out as soon as $z_p > f^U$. Moreover, δ works as a stopping criterion as the algorithm implicitly (by not storing) compares the gap between f^U and $\min_p z_p$; stop if $(f^U - \min_p z_p) < \delta$. The algorithm proceeds by selecting the interval corresponding to $\min_p z_p$ (most promising) and splitting it over the minimum point of the saw tooth cover $\varphi(x)$ defined by:

$$m_p = \frac{f(l_p) - f(r_p)}{2L} + \frac{r_p + l_p}{2} \tag{12}$$

being the next point to be evaluated. By continuing evaluating, splitting and throwing out intervals where the optimum cannot be, finally the stopping criterion is reached and we are certain to be δ from f^* and therefore $\epsilon = \delta/L$ from one of the global minimum points. The consequence of using such an algorithm, in contrast to the other algorithms, is that we now have to store information in a computer consisting of a list Λ of intervals. This compu-



1 [3,7] f^U =1.65 z1 =-6.05
2 [3,4.8] f^U =1.54 z2 =-2.16
3 [4.8,7] f^U =1.54 z3 =-2.16
4 [3,3.9] f^U =-0.08 z4 =-1.12
5 [3.9,4.8] f^U =-0.08 z5 =-1.12
6 [4.8,5.7] f^U =-0.08 z6 =-0.98
7 [5.7,7] f^U =-0.08 z7 =-0.98
8 [3,3.66] f^U =-0.2 z8 =-0.66
9 [3.66,3.9] f^U =-0.2 z9=-0.66
10 [3.9,4.16] f^U =-0.2 z10 =-0.32
11 [4.16,4.8] f^U =-0.2 z11 =-0.32
12 [4.8,5.4] f^U =-0.2 z12 =-0.26
13 [5.4,5.7] f^U =-0.2 z13 =-0.26
14 f^U =-0.2 > z14 =-0.19
15 f^U =-0.2 > z15 =-0.19

Fig. 5. Branch-and-Bound tree of Piyavskii-Shubert for function g

tational effort is now added to that of evaluating sample points and doing intermediate calculations. This concept becomes more clear when running the algorithm on the test function g using an accuracy of $\delta = 0.01$. The Lipschitz constant $L_g = 4.2$ is used for illustrative purposes. As can be read from Table 3, the algorithm is slowly converging. After some iterations, 15 intervals have been generated of which 6 are stored and 2 can be discarded due to the bounding; it has been proven that the minimum cannot be in the interval $[5.67, 7]$. The current estimate of the optimum is $x^U = 3.66, f^U = -0.19$ and the current lower bound is given by $\min_p z_p = -0.68$. Figure 5 illustrates the appearing binary structure of the search tree.

The maximum computational effort with respect to storing intervals is reached when the branching proceeds and no parts can be thrown out; 2^K intervals appear at the bottom of the tree, where K is the depth of the tree. This mainly happens when the used Lipschitz parameter L is overestimating the maximum slope drastically, or seen in the other way, the function is very flat compared to the used constant L. In that case, the function is evaluated in more than the M points of the regular grid. With a correct constant L, the number of evaluated points is less, as part of the domain can be discarded as illustrated here.

Table 3. Piyavskii-Shubert for function g, $\delta = 0.01$

p	l_p	r_p	$f(l_p)$	$f(r_p)$	m_p	z_p	f^U	x^U	
1	3.00	7.00	1.65	3.44	4.79	-5.85	1.65	3.00	split
2	3.00	4.79	1.65	1.54	3.91	-2.16	1.54	4.79	split
3	4.79	7.00	1.54	3.44	5.67	-2.16	1.54	4.79	split
4	3.00	3.91	1.65	-0.08	3.66	-1.12	-0.08	3.91	split
5	3.91	4.79	-0.08	1.54	4.15	-1.12	-0.08	3.91	split
6	4.79	5.67	1.54	0.20	5.39	-0.98	-0.08	3.91	split
7	5.67	7.00	0.20	3.44	5.95	-0.98	-0.08	3.91	split
8	3.00	3.66	1.65	-0.20	3.55	-0.66	-0.20	3.66	
9	3.66	3.91	-0.20	-0.08	3.77	-0.66	-0.20	3.66	
10	3.91	4.15	-0.08	0.47	3.96	-0.32	-0.20	3.66	
11	4.15	4.79	0.47	1.54	4.34	-0.32	-0.20	3.66	
12	4.79	5.39	1.54	0.46	5.22	-0.26	-0.20	3.66	
13	5.39	5.67	0.46	0.20	5.56	-0.26	-0.20	3.66	
14	5.67	5.95	0.20	0.61	5.76	-0.19	-0.20	3.66	discarded
15	5.95	7.00	0.61	3.44	6.14	-0.19	-0.20	3.66	discarded

2.4 Stochastic GO: PRS, Multistart, Simulated Annealing

Stochastic methods are understood to contain some stochastic elements. Either the outcome of the method is a random variable or the objective function itself is considered a realisation of a stochastic process. For an overview on stochastic methods we refer to [BR95, TZ89, TAV99]. Two classical approaches from Global Optimization, Pure Random Search (PRS) and Multistart are analysed for the test cases. This is followed by a classical variant of Simulated Annealing, a so-called heuristic.

Pure Random Search (PRS) generates points uniformly over the domain and stores the point corresponding to the best value as the approximation of the global minimum point. The algorithm is popular as a reference algorithm as it can easily be analysed. The question is how it behaves for our test cases g and h. The domain is clearly the interval $[3, 7]$, but what can be defined as the success region now? Let the success be defined as the case that one of the generated points is closer than $\epsilon = 0.01$ to the global minimum point. The probability we do NOT hit this region after $N = 50$ trials is $(3.98/4)^{50} \approx 0.78$. In the specific case, the size of the success region is namely $2 \times \epsilon$ and the size of the feasible area is 4. The probability of NOT hitting is $(1 - \frac{0.02}{4})$ and of NOT hitting 50 times is $(1 - \frac{0.02}{4})^{50}$. This means that the probability of success as efficiency indicator has a value of about 0.22 for both cases h and g. A similar analysis can be done for determining the probability that the function value of PRS after $N = 50$ iterations is less than $f^* + \delta$ for $\delta = 0.01$. The usual tool in the analysis on the function space is to introduce $y{=}f(x)$ as a random variate representing the function value, where x is uniformly distributed over X. Value y has distribution function $F(y) = P(f(x) \leq y)$. Keeping this in mind, analysis with so-called extreme

```
$f^U = \infty$
for ($k = 1$ to $N$) do
    Generate $x_k$ uniformly over $X$
    if $f(x_k) < f^U$
        $f^U = f(x_k)$ and $x^U = x_k$
End for
```

Algorithm 5: PRS(X, f, N)

order statistics has shown that the outcome of PRS as record value of N points can be easily derived from $F(y)$. For a complete introduction into extreme order statistics in optimisation, we refer to [Zhi91]. Under mild assumptions it can be shown that $\boldsymbol{y}_{(1)} = \min\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)\}$ has the distribution function $F_{(1)}(y) = 1 - (1 - F(y))^N$. This means that for the question about the probability that $\boldsymbol{y}_{(1)} \leq f^* + \delta$, we dont have to know the complete distribution function F, but only the probability mass $F(f^* + \delta)$ of the success level set where $f(x) \leq f^* + \delta$, i.e. the probability that one sample point hits this low level set. Here the 2 test cases differ considerably. One can verify that the level set of the more smooth test function g is about 0.09 wide, whereas that of function h is only 0.04 wide for a δ of 0.01. This means that the probability of PRS to reach a level below $f^* + \delta$ after 50 evaluations for function g is $1 - (1 - \frac{0.09}{4})^{50} = 0.68$, whereas the same probability for function h is $1 - (1 - \frac{0.04}{4})^{50} = 0.40$.

An early obvservation based on the extreme order statistic analysis is due to [Kar63]. Surprisingly enough, Karnopp showed that the probability of finding a better function value with one draw more after N points have been generated, is $\frac{1}{N+1}$, independent of the problem to be solved. Generating K more points increases the probability to $\frac{K}{N+K}$. The derivation which also can be found in [TZ89] is based on extreme order statistics and only requires F not to behave too strange, e.g. F is continuous such that f should not have plateaus on the domain X.

In conclusion, stochastic algorithms show something which in the literature is called the **infinite effort property**. This means that if one proceeds long enough (read $N \rightarrow \infty$) in the end the global optimum is found. The problem with such a concept is that infinity can be pretty far away. Moreover, we have seen in the earlier analyses that the probability of reaching what one wants, depends considerably on the size of the success region. One classical way of increasing the probability of reaching an optimum is to use (nonlinear optimisation) local searches. This method is called **multistart**.

Define a local optimisation routine $LS(x) : X \rightarrow X$ as a procedure which given a starting point returns a point in the domain that approximates a local minimum point. As an example, one can consider the Newton method

```
$f^U = \infty$
for ($k = 1$ to $N$) do
    Generate $x$ uniformly over $X$
    $x_k = LS(x)$
    if $f(x_k) < f^U$
        $f^U = f(x_k)$ and $x^U = x_k$
End for
```

Algorithm 6: Multistart(X, f, LS, N)

of Sect. 2.2. Multistart generates convergence points of a local optimisation routine from randomly generated starting points.

Notice that the number of iterations N is not comparable with that in PRS, as every local search requires several function evaluations. Let us for the example cases assume that the Newton algorithm requires 5 function evaluations to detect an attraction point, as is also implied by Table 2. As we were using $N = 50$ function evaluations to assess the success of PRS on the test cases, we will use $N = 10$ iterations for Multistart. In order to determine a similar probability of success, one should find the relative size of the region of attraction of the global minimum point. Notice again, that the Newton algorithm does not always converge to the nearest optimum; it only converges to a minimum point in a convex region around it.

For function g, the region of attraction of the global minimum is not easy to determine. It consists of a range of about 0.8 on the feasible area of size 4, such that the probability one random starting point leads to success is 0.8/4 = 0.2. For function h, the good region of attraction is simply the bubble of size 0.25 around the global minimum point, such that the probability of finding the global minimum in one iteration is about 0.06. Reaching the optimum after $N = 10$ restarts is $1 - 0.8^{10} \approx 0.89$ for g and $1 - 0.94^{10} \approx 0.48$ for h. In both examples, the probability of success is larger than that of PRS.

As sketched sofar, the algorithms of Pure Random Search and Multistart have been analysed widely in the literature of GO. Algorithms that are far less easy to analyse, but very popular in applications are the collection of so-called metaheuristics. This term was introduced by Fred Glover in [Glo86] and includes Simulated Annealing, Evolutionary algorithms, Genetic algorithms, tabu search and all fantasy names derived from crossovers of the other names. Originally these algorithms were not only aimed at continuous optimisation problems, see [AL97]. An interesting research question is whether they are really better than combining classical ideas of random search and nonlinear optimisation local searches. We discuss here a variant of **Simulated annealing**, a concept that also got attention in the GO literature, see [Rom92]. Simulated annealing describes a sampling process in the decision space where new sample points are generated from a so-called neighbourhood of the current

```
$f^U = \infty$, $T_1 = 1000$
Generate $x_1$ uniformly over $X$
**for** ($k = 1$ to $N$) do
    Generate $x$ from a neighbourhood of $x_k$
    **if** $f(x) < f(x_k)$
        $x_{k+1} = x$
        **if** $f(x) < f^U$
            $f^U = f(x)$ and $x^U = x$
    **else** with probability $e^{\frac{f(x_k)-f(x)}{T_k}}$ let $x_{k+1} = x$
    $T_{k+1} = CR \times T_k$
End **for**
```

Algorithm 7: SA(X, f, CR, N)

iterate. The new sample point is always accepted when it is better and with a certain probability when it is worse. The probability depends on the so-called temperature that is decreasing (cooling) during the iterations. The algorithm contains the parameter CR representing the *Cooling rate* with which the temperature variable decreases. A fixed value of 1000 was taken for the initial temperature to avoid creating another algorithm parameter. The algorithm accepts a worse point depending on how much it is worse and the development of the algorithm. This is a generic concept in Simulated Annealing. There are several ways to implement the concept of "sample from neighbourhood". In one dimension one would perceive intuitively a neighbourhood of x_k in real space $[x_k - \epsilon, x_k + \epsilon]$, which can be found in many algorithms e.g. see [BDH+05]. As originally such heuristics were not aiming at continuous optimisation problems, but at integer problems, one of the first approaches was the coding of continuous variables in bit strings. For the illustrations, we elaborate this idea for the test case. Each point $x \in [3, 7]$ is represented by a bitstring $(B_1, \ldots, B_9) \in \{0, 1\}^9$ where,

$$x = 3 + 4\frac{\sum_{i=1}^{9} B_i 2^{i-1}}{511} \tag{13}$$

Formula (13) describes a regular grid over the interval, where each of the M=512 bitstrings is one of the grid points, such that the mesh size is $\frac{4}{511}$. The sampling from a neighbourhood of a point x is done by flipping at random one of its bit variables B_i from a value of 0 to 1, or the other way around. Notice that by doing so, the generated point is not necessarily in what one would perceive as a neighbourhood in continuous space. The question is therefore, whether the described SA variant will perform better than an algorithm where the new sample point does not depend on the current iterate, PRS. To test this, a figure is introduced that is quite common in experimenting with metaheuristics. It is a graph with on the x-axis the effort and on the y-axis the reached success. The GO literature often looks at the two criteria effectiveness and efficiency separately. Figure 6 being a special case of what in [BH05] was called the **performance graph**, gives a trade-off between the two main criteria. One can also consider the x-axis to give a budget with which one has to reach a level as low as possible, see [HR96]. In this way one can change the search strategy depending on the amount of available running time. The figure suggests for instance that a high cooling rate (the process looks like PRS) is doing better for a lower number of function values and doing worse for a higher number of function values. Figure 6 gives an estimation of the expected

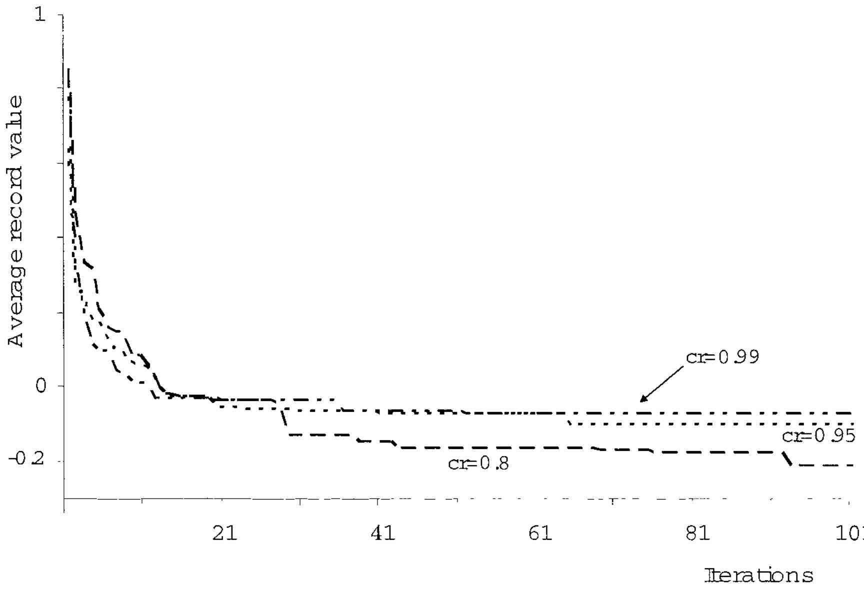


Fig. 6. Average record value over 10 runs of SA, 3 different values of the Cooling Rate cr for a given amount of function evaluations, test case g

level one can reach by running SA on function g. Implicitly it says the user wants to reach a low function value; not necessarily a global minimum point. Theoretically, one can derive the expected level analytically by considering the process from a Markov chain perspective, see e.g. [BW98]. However, usually the estimation is done empirically and the figure is therefore very common in metaheuristic approaches. The reader will not be surprised that the figure looks similar for function h, as the number of local optima is not relevant for the bitstring perspective and the function value distribution is similar. Theoretically, one can also derive the expected value of the minimum function value reached by PRS. It is easier to consider the theoretical behaviour from the perspective where success is defined boolean, as has been done so far.

Let us consider again the situation that the algorithm reaches the global optimum as success. For stochastic algorithms we are interested in the probability of success. Define reaching the optimum again as finding a point with function value closer than $\delta = 0.01$ to the minimum. For function g this is about 2.2% (11 out of the 512 bitstrings) of the domain. For PRS, one can determine the probability of success as $P_{PRS}(N) = 1 - 0.978^N$. For SA this is much harder to determine, but one can estimate the probability of success empirically. The result is the performance graph in Fig. 7. Let us have a look

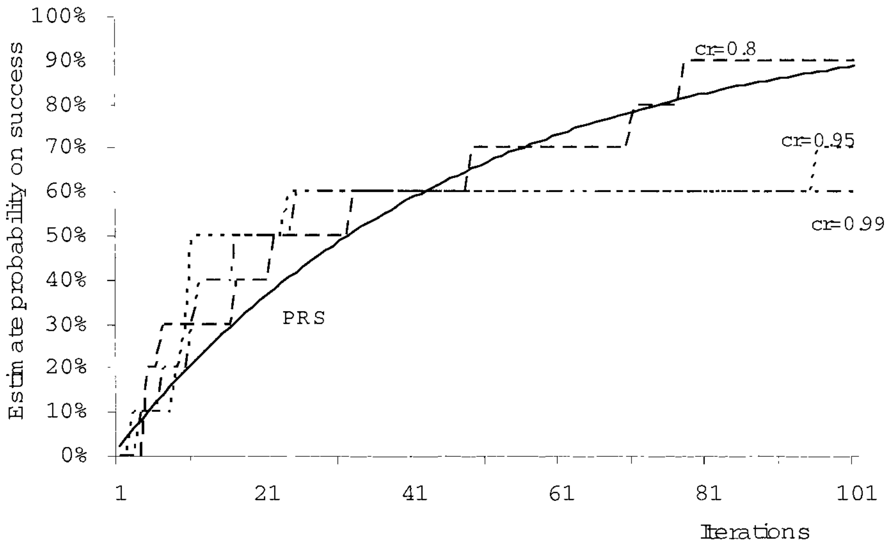


Fig. 7. Estimate of probability of reaching the minimum for PRS and SA (average over 10 runs) on function g given a number of function evaluations

at the figure critically. In fact it suggests that PRS is doing as good as the SA algorithms. As this is verifying a hypothesis (not falsifying), this is a reason to be suspicious. The following critical remarks can be made.

- the 10 runs are enough to illustrate how the performance can be estimated, but is too low to discriminate between methods. Perhaps the author has even selected a set of runs which fits the hypothesis nicely.
- one can choose the scale of the axes to focus on an effect. In this case, one can observe that up to 40 iterations, PRS does not look better than the

SA variants. By choosing the x-axis to run to 100 iterations, it looks much better.
- the graph has been depicted for function g, but not for function h, where the size of the success region is twice as small. One can verify, that in the given range, the SA variants are nearly always doing better.

This brings us to the general scientific remark, that all results should be described in such a way that they can be **reproduced**, i.e. one should be able to repeat the experiment. For the exercises reported in this section, this is relatively simple. Spreadsheet calculations and for instance matlab implementations can easily be made.

3 Investigating Algorithms

In Sect. 2, we have seen how several algorithms behave on two test cases. What did we learn from that? How to do the investigation systematically? Figure 8 depicts some relevant aspects. All aspects should be considered together. The

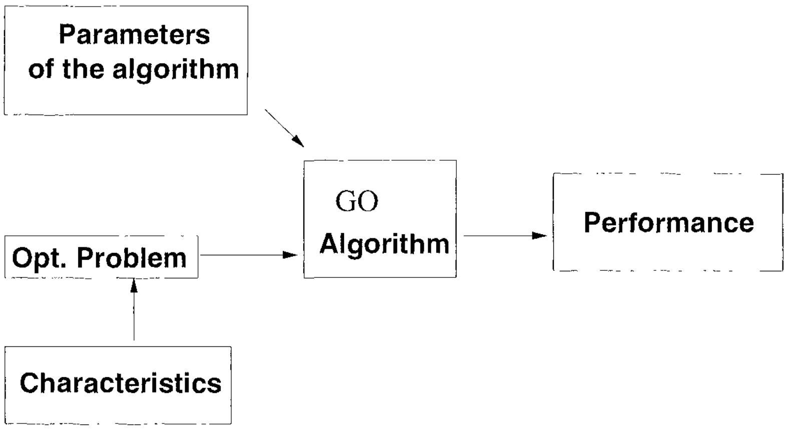


Fig. 8. Aspects of investigating Global Optimization Algorithms

following steps are described in [BH05].

1. Formulation of performance criteria.
2. Description of the algorithm(s) under investigation.
3. Selection of appropriate algorithm parameters.
4. Production of test functions (instances, special cases) corresponding to certain landscape **structures or characteristics**.

5. Analysis of its theoretical performance, or empirical testing.

Several criteria and performance indicators have been sketched: to get a low value, to reach a local minimum point, a high probability to hit an ϵ neighborhood of a global minimum point, obtain a guarantee to be less than δ from the minimum function value, etc. Several classes of algorithms have been outlined. The number of parameters has been kept low, a Lipschitz constant L, the number of iterations N, cooling rate CR, stopping accuracy α. Many modern heuristics contain so many tuning parameters, that it is hard to determine the effect of their value on the performance of the algorithm.

Only two test functions were introduced to experiment with. The main difference between them is the number of optima, which in literature is seen as an important characteristic. However, in the illustrations, the number of optima was only important for the performance of multistart. The piecewise convexity appeared important for the local search Newton algorithm and further mainly the difference in size of the δ-level set was of importance for the probability of success of stochastic algorithms. It teaches us, that in a research setting, one should think carefully, make hypotheses and design corresponding experiments, to determine which characteristics of test functions are relevant for the algorithm under investigation.

3.1 Characteristics

In [BDH$^+$05], an attempt is made to analyse the interrelation between the characteristics, called "landscape" in their view, of the test cases and the behaviour of the algorithms. What we see experimentally, is that often an algorithm is run over several test functions and its performance compared to other algorithms and/or other parameter settings. To understand behaviour, we need to study the relationships to characteristics (landscapes) of test functions. The main question is how to define appropriate characteristics. We will discuss some ideas which appear in literature. The main idea, as illustrated before, is that relevant characteristics depend on the type of algorithm as well as on the performance measure. Extending previous stochastic examples to more dimensions, it is only the relative size of the sought for success region that matters, characteristics such as the number of optima, the shape of regions of attraction, the form of the level sets, barriers in the landscape do not matter.

It is also important to vary the test cases systematically between the extreme cases, in order to understand how algorithms behave. In an experimental setting, depending on what one measures, one tries to design experiments which yield as much information as possible. To derive analytical results, it is not uncommon to make highly specific assumptions which make the analysis tractable. In GO literature, the following classes of problems with respect to available information are distinguished.

- Black box (or oracle) case: in this case it is assumed that nothing is known about the function to be optimized. Often the feasible set is defined as a box, but information about the objective function can only be obtained by evaluating the function at feasible points.
- Grey box case: something is known about the function, but the explicit form is not necessarily given. We may have a lower bound on the function value or on the number of global and/or local optima. As has proven useful for deterministic methods, we may have structural information such as: a concave function, a known Lipschitz constant, a known so-called d.c-decomposition. Stochastic methods often don't require this type of information, but this information may be used to derive analytical or experimental results.
- White box case: explicit analytical expressions of the problem to be solved are assumed to be available. Specifically so-called interval arithmetic algorithms require this point of view on the problem to be solved.

When looking at the structure of the instances for which one studies the behaviour of the algorithm, we should keep two things in mind.

- In experiments, the researcher can try to influence the characteristics of the test cases such that the effect on what is measured is as big as possible. Note that the experimentalist knows the structure in advance, but the algorithm doesn't.
- The algorithm can try to generate information which tells it about the structure of the problems. We will enumerate some information which can be measured in the black box case.

When we have a look at the lists of test functions in literature (a.o. [TZ89]), we observe as characteristics the number of Global minimum points, the number of local minimum points and the dimension of the problem.

A difficulty in the analysis of a GO algorithm in the multiextremal case is, that everything seems to influence behaviour: The orientation of components of lower level sets with respect to each other determine how iterates can jump from one place to the other. The number of local optima up in the "hills" determine how algorithms may get stuck in local optima. The difference between the global minimum and the next lowest minimum affects the ability of detecting the global minimum point. The steepness around minimum points, valleys, creeks etc. which determine the landscape influences the success. However, we stress, as shown by the examples, the problem **characteristics** which are important for the behaviour depend on the **type of algorithm** and the **performance criteria** that measure the success.

In general, stochastic algorithms require no structural information about the problem. However, one can adapt algorithms to make use of structure information. Moreover, one should notice, that even if structural information is not available, other so-called **value information**, becomes available when running algorithms: the number of local optima found thus far, the average number of function evaluations necessary for one local search, best function value found, and the behavior of the local search etc. Such indicators can be measured empirically and can be used to get insight into what factors determine the behaviour of a particular algorithm and perhaps can be used to improve the performance of an algorithm. From the perspective of designing algorithms, running them empirically may **generate information** about the landscape of the problem to be solved. The following list of information can be collected during running a stochastic GO algorithm on a black box case, see [Hen98]:

- Graphical information on the decision space.
- Current function value.
- Best function value found so far (record).
- Number of evaluations in the current local phase.
- Number of optima found.
- Number of times each detected minimum point is found.
- Estimates of the time of one function evaluation.
- Estimates of the number of function evaluations for one local search.
- Indicators on the likelihood to have found an optimal solution

For the likelihood indicator, a probability model is needed. Measuring and using the information in the algorithm, usually leads to more extended algorithms, called "adaptive". Often, these have additional parameters complicating the analysis of what are good parameter settings.

3.2 Comparison of Algorithms

When comparing algorithms, a specific algorithm is **dominated**, if there is another algorithm which performs better (e.g. has a higher probability performance graph) in all possible cases under consideration. Usually however, one algorithm runs better on some cases and another on other cases.

So basically, the performance of algorithms can be compared on the same test function, or preferably for many test functions with the same characteristic, where that characteristic is the only parameter that matters for the performance of the compared algorithms. As we have seen, it may be very hard to find out such characteristics. The following principles can be useful:

- Comparability: When comparing several algorithms, they should all make use of the same type of (structural) information (same stopping criteria, accuracies etc).
- Simple references: It is wise to include in the comparison simple benchmark algorithms such as Pure Random Search, Multistart and Grid Search in order not to let analysis of the outcomes get lost in parameter tuning and complicated schemes.
- Reproducibility: In principle, the description of the method that is used to generate the results, has to be so complete, that someone else can repeat the exercise obtaining similar results (not necessarily the same).

Often in applied literature, we see algorithms used for solving "practical" problems, see e.g. [AST97, Hen98, Pin96] for extensive studies. As such this illustrates the relevance of the study on Global Optimization algorithms. In papers where one practical problem is approached with one (type of) algorithm, the reference is lacking. First of all we should know the performance of simple benchmark algorithms on that problem. Secondly, if structure information is lacking, we do not learn a lot of the performance of the algorithm under study on a class of optimization problems. We should keep in mind this is only one problem and up to now, it does not represent all possible practical problems.

4 Summary and Discussion Points

- Results of investigation of GO algorithms consist of a description of the performance of algorithms (parameter settings) depending on characteristics of test functions or function classes.
- To obtain good performance criteria, one needs to identify the target of an assumed user and to define what is considered as success.
- The performance graph is a useful instrument to compare performance.
- The relevant characteristics of the test cases depend on the type of algorithm and performance criterion under consideration.
- Algorithms to be comparable must make use of same information, accuracies, principles etc.
- It is wise to include the performance of simple benchmark algorithms like Grid Search, PRS and Multistart as references in a study on a GO algorithm.
- Description of the research path followed should allow reproduction of experiments.

Acknowledgement. Special thanks are due to Bill Baritompa, Joke van Lemmen and two anonymous referees for reading and commenting on earlier versions of the manuscript.

References

[AL97] Aarts, E.H.L., Lenstra, J.K.: Local Search Algorithms. Wiley, New York (1997)

[AST97] Ali, M.M., Story, C., Törn, A.: Application of stochastic global optimization algorithms to practical problems. Journal of Optimization Theory and Applications, **95**, 545–563 (1997)

[BDH$^+$05] Baritompa, W.P., Dür, M., Hendrix, E.M.T., Noakes, L., Pullan, W.J., Wood, G. R.: Matching stochastic algorithms to objective function landscapes. Journal of Global Optimization, **31**, 579–598 (2005)

[BH05] Baritompa, W.P., Hendrix, E.M.T.: On the investigation of stochastic global optimization algorithms. Journal of Global Optimization, **31**, 567–578 (2005)

[BMW+95] Baritompa, W.P., Mladineo, R.H., Wood, G.R., Zabinsky, Z.B., Baoping, Z.: Towards pure adaptive search. Journal of Global Optimization, **7**, 73–110 (1995)

[BR95] Boender, C.G.E., Romeijn H.E.: Stochastic methods. In: Horst, R., Pardalos, P.M. (eds) Handbook of Global Optimization, pp. 829–871. Kluwer, Dordrecht (1995)

[BW98] Bulger, D.W., Wood, G.R.: Hesitant adaptive search for global optimisation. Mathematical Programming, **81**, 89–102 (1998)

[DP67] Danilin, Y., Piyavskii, S.A.: An algorithm for finding the absolute minimum. Theory of Optimal Decisions, **2**, 25–37 (1967) (in Russian)

[Glo86] Glover, F.W.: Future paths for integer programming and link to artificial intelligence. Computers and Operations Research, **13**, 533–554 (1986)

[GMW81] Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, New York (1981)

[Hen98] Hendrix, E.M.T.: Global Optimization at Work. PhD thesis, Wageningen University, Wageningen, June (1998)

[HK00] Hendrix, E.M.T., Klepper, O.: On uniform covering, adaptive random search and raspberries. Journal of Global Optimization, **18**, 143–163 (2000)

[HOG01] Hendrix, E.M.T., Ortigosa, P.M., García, I.: On success rates for controlled random search. Journal of Global Optimization, **21**, 239–263 (2001)

[HR96] Hendrix, E.M.T., Roosma, J.: Global optimization with a limited solution time. Journal of Global Optimization, **8**, 413–427 (1996)

[Kar63] Karnopp, D.C.: Random search techniques for optimization problems. Automatica, **1**, 111–121 (1963)

[Pin96] Pintér, J.D.: Global Optimization in Action; Continuous and Lipschitz Optimization: Algorithms, Implementations and Application. Kluwer, Dordrecht (1996)

[Rom92] Romeijn, H.E.: Global Optimization by Random Walk Sampling Methods. PhD thesis, Erasmus University Rotterdam, Rotterdam, May (1992)

[Sca85] Scales, L.E.: Introduction to Non-Linear Opimization. Macmillan, London (1985)

[Shu72] Shubert, B.O.: A sequential method seeking the global maximum of a function. SIAM Journal of Numerical Analysis, **9**, 379–388 (1972)

[TAV99] Törn, A., Ali, M.M., Viitanen, S.: Stochastic global optimization: Problem classes and solution techniques. Journal of Global Optimization, **14**, 437–447 (1999)

[TZ89] Törn, A., Zilinskas, A.: Global Optimization. Vol. 350 of Lecture Notes in Computer Science. Springer, Berlin (1989)

[Zhi91] Zhigljavsky, A.A.: Theory of Global Random Search. Kluwer, Dordrecht (1991)

[ZS92] Zabinsky, Z.B., Smith, R.L.: Pure adaptive search in global optimization. Mathematical Programming, **53**, 323–338 (1992)

Experimental Investigation of Distance Graduate Studies of the Open Source Environment by Models of Optimal Sequential Decisions and the Bayesian Approach

Jonas Mockus

[1] Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania
[2] Kaunas Technological University, K.Donelaičio g. 73, LT-44029 Kaunas, Lithuania jmockus@gmail.com

Summary. Development and applications of the open source software is a new and dynamic field. Important changes often happen in days and weeks. Thus some new non-traditional approaches of education should be investigated to meet the needs of open software adequately.

The general consideration of this problem is difficult. Therefore we start by relevant case studies. In this chapter we consider models of optimal sequential decisions with multiple objective functions as an example. The aim is to show that models can be implemented and updated by graduate students themselves. That reflects the usual procedures of the open source development. This way students not only learn the underlaying model but obtain the experience in the development of open source software.

In this case the step-by-step improvement of the model and software is at least as important as the final result that is never achieved in open source environment as usual. A short presentation of the basic ideas is in [Moc00]. Note that doing this we accumulate some experience in the completely new field of education when all the information can be easily obtained by Internet. The users are doing just the creative part by filtering and transforming the information to meet their own objectives, to build their own models. The natural way is computer experimentation.

To make the task as easy as possible all the algorithms considered in this chapter are implemented as platform independent Java applets or servlets therefore readers can easily verify and apply the results for studies and for real life optimization models.

To address this idea the chapter is arranged in a way convenient for the direct reader participation. Therefore a part of the chapter is written as some 'user guide'. The rest is a short description of optimization algorithms and models. All the remaining information is on web-sites, for example *http : //pilis.if.ktu.lt/mockus.*

Key words: sequential decisions, recurrent equations, Bayesian approach, distance studies.

1 Introduction

In this chapter traditional tables and graphs are replaced by short description of algorithms and models with references to web-sites where on-line models are presented. The web-sites contain a set of Java applets and servlets thus all the comparisons can be made by the readers. To start Java applets the web browser must have some Java plug-in. For Java servlets any browser works.

The main scientific tool is the theory of optimal sequential statistical decisions using the Bayesian approach [Wal50, DeG70]. The examples are simplified economic and social models transformed into the optimization problems. The models are not difficult for understanding. The computing time does not exceed reasonable limits as usual.

The full set of the on-line examples and the complete theoretical explanation is on the six mirror web-sites. This increases the reliability of Internet communications.

There are no "perfect" examples in these web-sites. All examples has some advantages and some disadvantages. Improvement of "non-perfect" models is interesting both for students and for colleagues. The main objective of this chapter is to establish scientific collaboration in the Internet environment with distant colleagues and students.

The chapter can be regarded as an extended introduction. The complete set of on-line models is available on the web-sites. Important part of this introduction is the presentation of the main ideas and goals for developing those web-sites.

In this chapter two well-known optimization topics - Sequential Decisions [Wal50] and Bayesian Approach [DeG70] are regarded using different optimization models as examples. We regard that as a reasonable first step because both these topics are related to most of real life decisions. We often have to make decisions with incomplete information. The future is uncertain, too, as usual.

2 Sequential Statistical Decisions

Most of the decisions in a personal life and in organizations are made sequentially. That means that, at any given moment, one either makes the final decision, or postpones it hoping for better times. Statistical part of sequential decisions represents uncertainties of future events and a limited reliability of our observations.

The Bride problem is a good academic example of sequential statistical decisions [Wal47, Wal50]. The dynamic programming is a conventional technique to optimize sequential decisions [Bel57]. Applying the dynamic programmingto specific problems, one develops specific algorithms, as usual. The algorithms, similar to these of bride's decision making, can be applied choosing the optimal time to buy durable goods, such as cars, houses, computers, to start new business, etc.

Typical industrial applications of sequential decisions are the optimal stopping rules. For example, one can consider stopping of some nuclear plant as a marriage. One can evaluate stopping cost directly. Risks involved in running the plant, when one observes some deviations from the normal routine, can be considered as risks of waiting for the next proposal. That means that one hopes for something better while risking a bad outcome.

The important personal example is buying a PC. The feature of this example is rapid growth of average goodness of the product. We can buy much better machine after a year or two for the same money. However we may lost a lot of time working with an obsolete equipment.

We start by considering the simplest "single bride" example (no divorce). The underlying reason is the simplicity of calculations. We may solve this problem by textbook recurrent equations [Bel57]. That is a good introduction. The "no-change" condition is not the realistic one therefore all the remaining examples regard the more realistic "multiple-bride" problems. Here changes are permitted for some price.

We often need many parameters to evaluate the real objective. That means a vector optimization problem. The important task is correct separation of the objective and the subjective factors. Both are important for decision making but in the different ways. In good decision making systems users can include all the available objective information and may easily represent the subjective opinion. A simple way is by fixing some "weights" defining the subjective importance of various objective factors. The objective is a multi-modal, in general. However we will consider mainly uni-modal problems for simplicity.

First some examples of The "Demo" systems are regarded. Here the goodness of future grooms are defined by random numbers generator. The demo examples are extended by adding the "Expert" versions where the actual parameters of the object can be entered by users. The systems response is not "mandatory". A user may accept or reject the systems "advice" depending on his/her subjective judgement.

We finish the study by some generalization of the brides problem. The example is the RAM problem. Here the zero-one case (yes or no) is generalized to multi-valued problem; how many RAM units to order. We start by simple case when both the retail and the wholesale prices are known in advance. Then we consider more realistic case when only statistical distribution of future wholesale prices is known. The optimal retail prices are obtained by market elasticity. The market elasticity shows how the demand depends on rte retail price.

In the last example we consider the Bayesian optimization of parameters of some Logistic Model representing the real case in the static mode since the dynamic solution is difficult and not easily understandable.

Note that each semester most of the examples are tested and updated. Observed bugs are eliminated, accuracy and user interface are improved. The illustration is the interval of numerical integration. When expected values change we need to change this interval, too. Now that is implemented just in "the well tested example of optimal sequential statistical decisions using dynamic programming" (see example "Optimal Marriage Time Problem" , part 'Discrete Optimization', web-site). In other examples the interval is fixed. This limits the 'trend' – the rate of change of expected values. In the 'Buy-a-PC' example the trend is the most important factor. Scalarization of the vector objective function is not always correct and convenient. Some important parameters are 'hidden' so recompilation is needed to change them. That shows how different students understand the vector optimization problem. There are comments about the advantages and disadvantages of examples. Testing and updating the existing examples is important part of graduate studies. The result is general improvement of the web-sites. That reflects the ideas of the open source development.

3 Bayesian Approach (BA)

The Bayesian Approach (BA) is defined by fixing a prior distribution P on a set of functions $f(x)$ and by minimizing the Bayesian risk function $R(x)$ [DeG70, Moc89a]. The risk function describes the average deviation from the global minimum. The distribution P is regarded as a stochastic model of $f(x)$, $x \in R^m$ where $f(x)$ can be a deterministic or a stochastic function. This is possible because using BA uncertain deterministic functions can be regarded as some stochastic functions [Lin71, DeG70, Sav54, Fin73, Zil86]. That is important feature of the Bayesian approach in this setup. For example, if several values of some deterministic function $z_i = f(x_i)$, $i = 1, ..., n$ are known then the level of uncertainty can be represented as the conditional standard deviation $s_n(x)$ of the corresponding stochastic function $f(x) = f(x, \omega)$ where ω is a stochastic variable. The aim of BA is to provide as small average error as possible.

4 Average Utility

The Bride problem is to maximize the average utility of marriage by the optimal choice of a groom. Denote the actual goodness of a groom i by ω_i. Denote by s_i the bride's impression about the groom i. $p(\omega_i)$ is a prior probability density of goodness ω_i. $p_s(s_i|\omega_i)$ is a probability density of impression s_i. Assume that goodness of different grooms are independent and identically distributed. This means that a prior probability density of goodness is

$$\begin{aligned} p(\omega_i, \omega_j) &= p(\omega_i)p(\omega_j), \\ p(\omega_i) &= p(\omega_j) = p(\omega). \end{aligned} \tag{1}$$

Suppose the probability density of an impression s_i, given the goodness $omega_i$, is

$$p_s(s_i, s_j|\omega) = p_s(s_i|\omega)p_s(s_j|\omega), \qquad (2)$$
$$p_s(s_i|\omega) = p_s(s_j|\omega) = p(s|\omega).$$

Assume the Gaussian prior probability density of goodness

$$p(\omega) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-1/2(\frac{\omega-\alpha_0}{\sigma_0})^2}. \qquad (3)$$

Here σ_0 is a prior standard deviation and α_0 is a prior mean of goodness. For example, $\alpha_0 > 0$ shows an optimistic bride. $\alpha_0 < 0$ shows that a bride is pessimistic. Suppose, that a prior probability density of bride impressions is

$$p(s|\omega) = \frac{1}{\sqrt{2\pi}\sigma} e^{-1/2(\frac{s-\omega}{\sigma})^2}. \qquad (4)$$

Here σ^2 is a variance of impressions around the true goodness ω.

We tacitly assumed that both the groom goodness and the bride impression are random variables depending on many independent factors. This explains the Gaussian distributions (3) and (4). One defines a posterior probability density of goodness ω, given the impression s, by the Bayesian formula [Bay83]

$$p(\omega|s) = \frac{p(s|\omega)p(\omega)}{p_s(s)}. \qquad (5)$$

Here

$$p_s(s) = \int_{-\infty}^{\infty} p(s|\omega)p(\omega)d\omega. \qquad (6)$$

5 Single-Marriage Case

Denote by d_i the bride's decision about the groom i

$$d_i = \begin{cases} 1, & \text{if bride marry the groom } i, \\ 0, & \text{otherwise.} \end{cases} \qquad (7)$$

Suppose that

$$\sum_{i=1}^{m} d_i = 1. \qquad (8)$$

The last condition means that brides marry, and marry only once. Formally condition (8) defines the set of feasible decisions when the n-th groom proposes

$$D_{N-n} = \begin{cases} 0 \text{ and } 1, & \text{if } g_{N-n} = 0, \\ 0, & \text{if } g_{N-n} = 1. \end{cases} \qquad (9)$$

Here g_{N-n} is the marriage index and N is a number of the last proposal.

$$g_{N-n} = 1 - \sum_{i=1}^{N-n-1} d_i. \tag{10}$$

The marriage index is zero, if the bride is marred. This prevents repeated marriages.

5.1 Bellman's Equations

The expected utility function is $u(s)$. Here s is the impression made by the successful groom[3].

$$u(s) = \int_{-\infty}^{\infty} \omega p(\omega|s) d\omega. \tag{11}$$

Denote by $u_N(s)$ the expected utility function, if the impression of the last groom is s

$$u_N(s) = \int_{-\infty}^{\infty} \omega p(\omega|s) d\omega. \tag{12}$$

Comparing (11) and (12) one observes that

$$u(s) = u_N(s). \tag{13}$$

This follows from independence assumptions (1) and (2). Denote by u_{N-1} the expected utility,if the impression of the $(N-1)$ th groom is s and a bride is making the optimal decision $d = d_{N-1}(s) \in D_{N-1}$

$$u_{N-1}(s) = \max_d (du(s) + (1-d)u_N), \tag{14}$$

$$d_{N-1}(s) = arg \max_d (du(s) + (1-d)u_N). \tag{15}$$

Here

$$u_N = \int_{-\infty}^{\infty} u_N(s) p_s(s) ds. \tag{16}$$

Following the same pattern, we define the expected utility, if the impression of the $(N-n)$-th groom is s and the bride is making the optimal decision $d = d_{N-n}(s) \in D_{N-n}$

$$u_{N-n}(s) = \max_d (du(s) + (1-d)u_{N-n+1}), \tag{17}$$

$$d_{N-n}(s) = arg \max_d (du(s) + (1-d)u_{N-n+1}). \tag{18}$$

[3] By the "successful groom" we mean the groom that a bride marries.

Here

$$u_{N-n+1} = \int_{-\infty}^{\infty} u_{N-n+1}(s)p_s(s)ds. \tag{19}$$

Note, that the utility $u(s)$ of accepting a proposal in expressions (17) and (18) is a function only of impression s. It does not depend on the proposal number $N - n$. That follows from independence assumptions (1) and (2). Solving these recurrent equationswe define the sequence of optimal decision functions $d_{N-n}(s) \in D_{N-n}$ and the expected utilities $u_{N-n}(s)$. This should be done for all possible impressions $s \in (-\infty, \infty)$ and for all numbers $n = 1, ..., N-1$. We cannot do that in continuous case. Therefore, we use a discrete approximation.

5.2 Discrete Approximation

From expressions (11) and (12), replacing the integrals by sums we obtain that

$$u_N(s) = u(s) = 2M/K \sum_{k=1}^{K} \omega_k p(\omega_k|s). \tag{20}$$

From expression (16)

$$u_N = 2M/K \sum_{k=1}^{K} u_N(s_k)p_s(s_k). \tag{21}$$

From expression (19)

$$u_{N-n+1} = 2M/K \sum_{k=1}^{K} u_{N-n+1}(s_k)p_s(s_k). \tag{22}$$

Here $\omega_k \in [-M + \alpha_0, M + \alpha_0]$, $\omega_1 = -M + \alpha_0$, $\omega_K = M + \alpha_0$ and $s_k \in [-M + \alpha_0, M]$, $s_1 = -M + \alpha_0$, $s_K = M + \alpha_0$ where α_0 is a prior mean of goodness. Note that α_0 often is a function of time $\alpha_0 = \alpha_0(t)$ that is a discrete approximation of the recurrent equations. All possible impressions $s_k \in [-M + \alpha_0, M + \alpha_0]$ and all numbers $n = 1, ..., N-1$ are considered. We set the number of iterations K by the accuracy needed.

The results are sequences of optimal decision functions $d_{N-n}(s_k)$ and the expected utilities $u_{N-n}(s_k)$. These sequences are stored in a set of arrays which define how the optimal decisions d and the expected utilities u depend on the possible impressions s_k, $k = 1, ..., K$. Doing this, one avoids the repeated calculations. Large arrays is a disadvantage of this procedure.

5.3 Including the Waiting Cost

The waiting lossesare important in many real-life sequential decision problems. Denote by c the loss of waiting for the next groom. Including this parameter into Bellman equations one obtains

$$u_{N-1}(s) = \max_d (du_N(s) + (1-d)(u_N - c)), \tag{23}$$

$$d_{N-1}(s) = arg \max_d (du_N(s) + (1-d)(u_N - c)). \tag{24}$$

In a similar way one defines the expected utility if the impression of the $(N-n)$-th groom is s and the bride is making the optimal decision $d_{N-n}(s)$

$$u_{N-n}(s) = \max_d (du_N(s) + (1-d)(u_{N-n+1} - c)), \tag{25}$$

$$d_{N-n}(s) = arg \max_d (du_N(s) + (1-d)(u_{N-n+1} - c)). \tag{26}$$

The other expressions remains the same.

5.4 Non-Linear Case

Expression (12) was defined assuming the linear bride's utility function. It was supposed that bride's utility is equal to the goodness of groom $u(\omega) = \omega$. In the real life, utility functions are nonlinear [Moc00] and non-convex, as usual. Then expression (12) is replaced by this integral

$$u_N(s) = \int_{-\infty}^{\infty} u(\omega) p(\omega|s) d\omega. \tag{27}$$

5.5 Software of "Single-Marriage"

Considering software we refere to parts of web-sites with corresponding applets or servlets. Applets implementing examples of this chapter are in "Discrete Optimization". The "Single-Marriage" example is in
"Optimal Marriage Time Problem":
examples of optimal sequential statistical decisions using dynamic programming
Bride-1, Java1
Figure 1 shows the input window of the "Sigle-Marriage" example, the simulated numeric results are in Fig. 3 and the optimal decision function defining how the critical "goodness" of groom depends on time is in Fig. 4.

The time is defined as a groom number. That means "uniform arrival" of grooms.

The Fig. 2 shows the Multi-Marriage example for comparison.

Condition (8) implics that brides marry and marry only once. No divorce is permitted. That is not realistic assumption made just to simplify the Bellman equations.

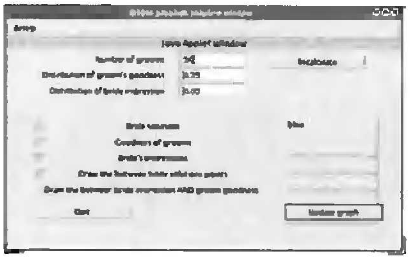


Fig. 1. Input window of "Single-Marriage"

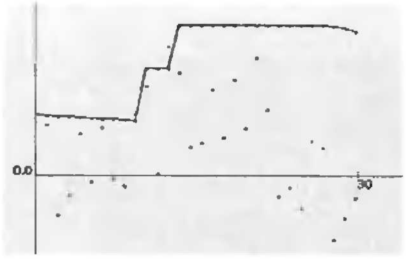


Fig. 2. Decision function of "Multi-Marriage"

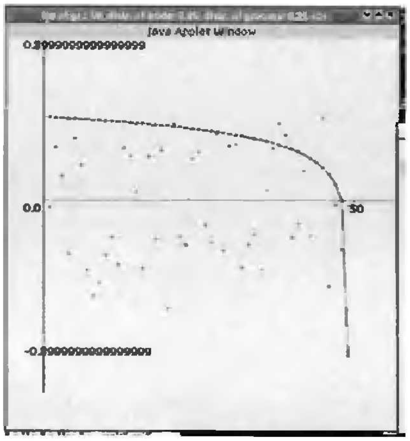


Fig. 3. Decision function of "Single-Marriage"

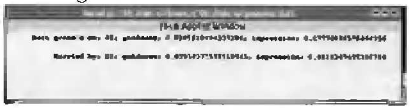


Fig. 4. Numeric results of "Single-Marriage"

6 "Multi-Marriage"

There are several sofware examples of the "Multi-Marriage" problem. The most accurate is
"the well tested example of optimal sequential statistical decisions using dynamic programming"
Bride-Multi, Java2
The screenshots of this example are in Figs. 5 and 6.

Number of grooms
Precision
Average
Distribution of groom's goodness
Distribution of bride's impression
Single stage
Calculate
Draw bride solution points
Draw line between bride impression AND groom goodness

Fig. 5. The output window of the "Grooms" problem

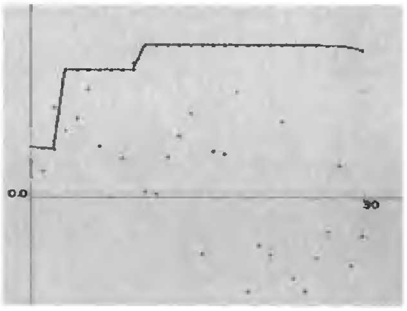


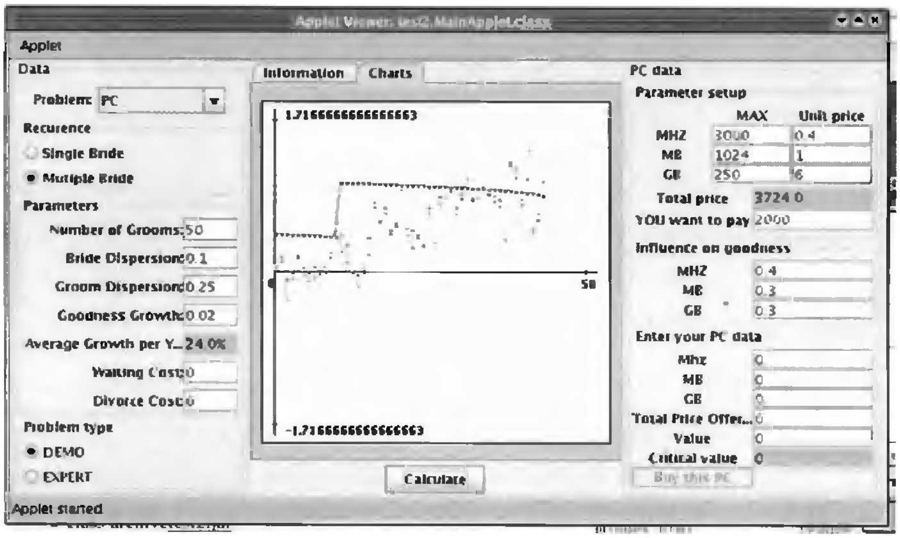
Fig. 6. The decision function of "Grooms" problem

6.1 "Buy-a-PC" Example

A good illustration of Bellman equations describing the "Multi-marriage" problem is the "Buy-a-PC" example.
"Buy-a-PC"
example of optimal sequential statistical decisions using dynamic programming in Java2
The rapid improvement of PC defining considerable positive trend (growth of expected gooness of PC).

Define PC parameters by a vector $g = (g_1, g_2, g_3)$. Here g_1 is the speed of CPU in MHz, g_2 is the volume of RAM in MB, and g_3 is the volume of HD in GB. Express a subjective utility of PC by the weighted sum $\omega = a_1 g_1 + a_2 g_2 + a_3 g_3$. Here a_i are expressed in $ per unit.

Considering real-life examples we need two GUI (Graphic User Interface) versions. We call them "Demo" and "Expert". In the Demo version Fig. 7 the "state of nature" (for example MB, GB, and GHz) is defined by some random number generator. In the Expert versions Figs. 8, 10 these values are entered by users, represents the real date and reflects subjective opinions. The final decision is defined by users and can be different from the computer suggestions.



Data
Problem: PC
Recurence
Single Bride
[✓ Multiple Bride]
Parameters
Number of Grooms: 50
Bride Dispersion: 0.1
Groom Dispersion: 0.25
Goodness Growth: 0.02
Average Growth per Y... 24.0%
Waiting Cost: 0
Divorce Cost: 0
Problem type
[✓ DEMO]
EXPERT

PC data
Parameter setup

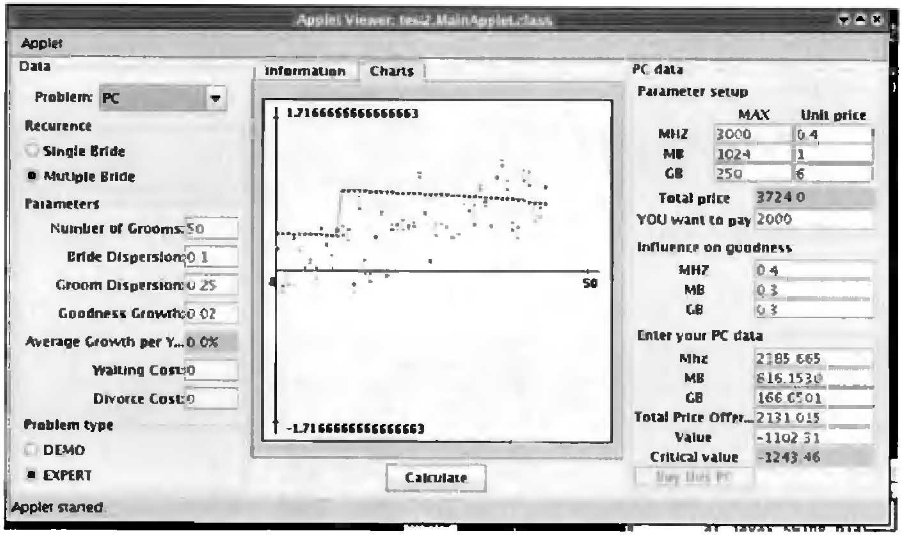
	MAX	Unit price
MHZ	3000	0.4
MB	1024	1
GB	250	6

Total price 3724.0
YOU want to pay 2000
Influence on goodness
MHZ 0.4
MB 0.3
GB 0.3
Enter your PC data
Mhz 0
MB 0
GB 0
Total Price Offer... 0
Value 0
Critical value 0

Fig. 7. The window of the demo "Buy-a-PC"

Assume that a prior probability density of PC utilities is Gaussian

$$p_t(\omega) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-1/2(\frac{\omega - \alpha_t}{\sigma_0})^2}. \tag{28}$$



Applet

Data

Problem: PC

Recurence
- Single Bride
- [✓ Mutiple Bride]

Parameters

Number of Grooms:	50
Bride Dispersion:	0.1
Groom Dispersion:	0.25
Goodness Growth:	0.02
Average Growth per Y...	0.0%
Waiting Cost:	0
Divorce Cost:	0

Problem type
- DEMO
- [✓ EXPERT]

Applet started

PC data

Parameter setup

	MAX	Unit price
MHZ	3000	0.4
MB	1024	1
GB	250	6

Total price	3724.0
YOU want to pay	2000

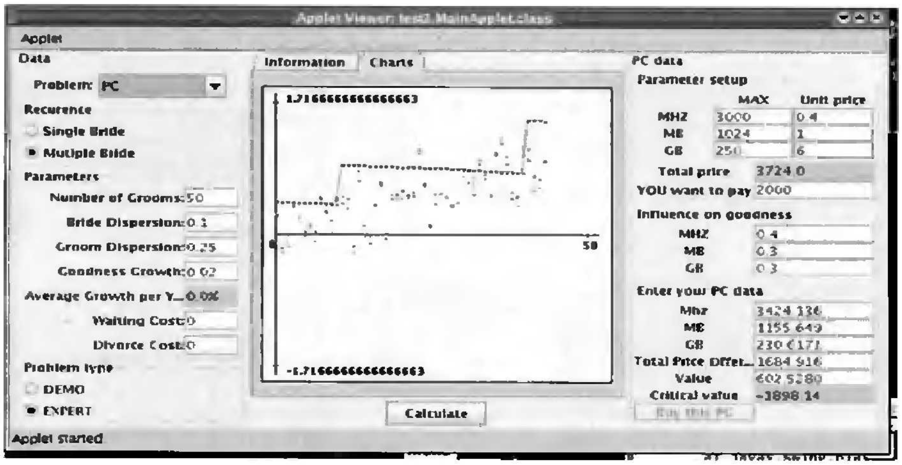
Influence on goodness

MHZ	0.4
MB	0.3
GB	0.3

Enter your PC data

Mhz	2185.665
MB	816.1530
GB	166.0501
Total Price Offer...	2131.015
Value	-1102.31
Critical value	-1243.46

Fig. 8. The input window of the expert "Buy-a-PC"



Applet

Data

Problem: PC

Recurence
- Single Bride
- [✓ Mutiple Bride]

Parameters

Number of Grooms:	50
Bride Dispersion:	0.1
Groom Dispersion:	0.25
Goodness Growth:	0.02
Average Growth per Y...	0.0%
Waiting Cost:	0
Divorce Cost:	0

Problem type
- DEMO
- [✓ EXPERT]

Applet started

PC data

Parameter setup

	MAX	Unit price
MHZ	3000	0.4
MB	1024	1
GB	250	6

Total price	3724.0
YOU want to pay	2000

Influence on goodness

MHZ	0.4
MB	0.3
GB	0.3

Enter your PC data

Mhz	3424.336
MB	1155.649
GB	230.0171
Total Price Offer...	1684.916
Value	602.5380
Critical value	-1898.14

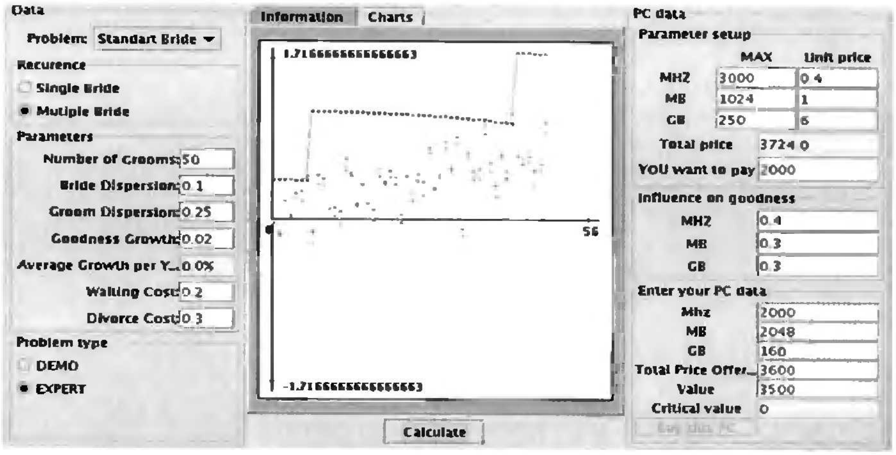
Fig. 9. The output window of the expert rejecting "Buy-a-PC" recomendations

Here σ_0 is a prior variance and α_t is a prior mean of PC utilities. Suppose that $\sigma_0 = constant$ and that $\alpha_t = \alpha_0 + \alpha_1 t$. This means that expected PC utility is increasing linearly. The expected diversity remains the same.

The PC price in \$ is denoted by l. Suppose that the price of PC depends linearly on the weighted sum

$$l = b_1 g_1 + b_2 g_2 + b_3 g_3. \tag{29}$$

Here parameters b_i are defined in \$ per unit and reflect the market prices of CPU, RAM, and HD. Expressing the price l as a function of utility ω:



Data

Problem: Standart Bride

Recurence
( ) Single Bride
(●) Mutiple Bride

Parameters

Number of Grooms:	50
Bride Dispersion:	0.1
Groom Dispersion:	0.25
Goodness Growth:	0.02
Average Growth per Y...	0.0%
Waiting Cost:	0.2
Divorce Cost:	0.3

Problem type
( ) DEMO
(●) EXPERT

PC data

Parameter setup

	MAX	Unit price
MHZ	3000	0.4
MB	1024	1
GB	250	6

Total price	3724.0
YOU want to pay	2000

Influence on goodness

MHZ	0.4
MB	0.3
GB	0.3

Enter your PC data

Mhz	2000
MB	2048
GB	160
Total Price Offer...	3600
Value	3500
Critical value	0

Fig. 10. The decision function

$$l = h_0 \omega. \tag{30}$$

Here

$$h_0 = \frac{b_1 g_1 + b_2 g_2 + b_3 g_3}{a_1 g_1 + a_2 g_2 + a_3 g_3}. \tag{31}$$

Assume that one observes the utility ω exactly. This means that the impression $s = \omega$ and that impression errors $\sigma = 0$ in expression (2). That simplifies the problem.

6.2 "Buy-a-Car"

The "Buy-a-Car" example Figs. 11 and 12 applies modified "Multi-Marriage" software. There are three options:
- autos "Buy-a-Car".
- workers "Employ-a-person",
- grooms "Marry-aGroom".

The implementation of Bellman equations is just a first approximation. For example, improving the Buy-a-Car model we should

- express desirable car properties using conditions of Pareto optimality [Moc89b],
- extend the property list by including properties that are not directly observable, such as the reliability of a car,
- assume the "No-Return" rule.

"No-Return" means that one cannot return to a situation which he or she rejected previously. The "No-Return" rule is essential in the dynamic programming. One considers "Return" cases in the framework of discrete or non-linear programming. That requires more computing, as usual.

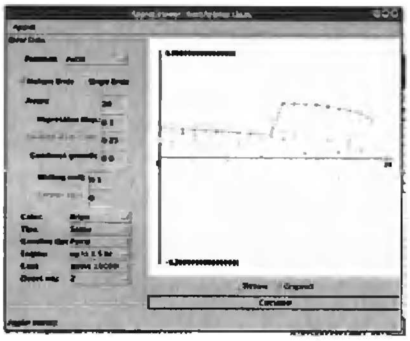


Fig. 11. The decision function of "Buy-a-Car"

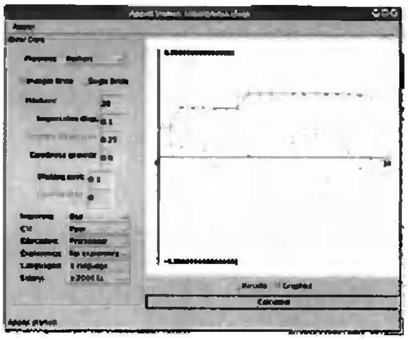


Fig. 12. The decision function of "Employ-a-Person"

6.3 "Employ-a-Person"

Another "Multi-Marriage" example is "Employ-a-Person" Fig. 12 (demo version). The comments about the "Buy-a-Car" are true in the "Employ-a-Person", too.

6.4 "Buy-a-Flat"

A "Buy-a-Flat" example implements "Multi-Marriage" model as an Java servlet. In the servlet mode all the calculations are performed at the server side. The users provide data and regard results Figs. 13 (Demo mode) and 14 (Expert mode). The user interface performs the scalarization of the vector objective in a "friendly" way.

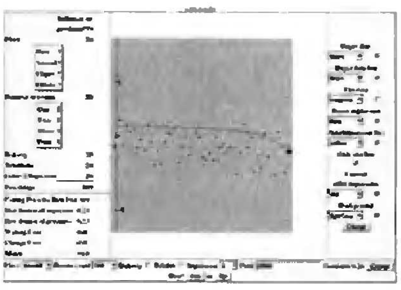


Fig. 13. The initial window of "Buy-a-Flat"

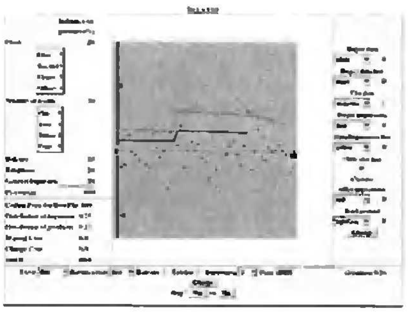


Fig. 14. The next window of "Buy-a-Flat"

6.5 Bellman's Equations

The expected utility function is $u(\omega, q)$. Here ω is the utility of a new PC. q is the utility of the old PC, to be replaced by the new one. Consider a "horizon" of N years. During a year one can change PC only once, if one wishes.

Denote by $u_N(\omega, q)$ the maximal expected utility in the year N

$$u_N(\omega, q) = \max_d(d\omega + (1-d)(q_N - c_N)). \tag{32}$$

There are two possible decisions $d = \{0, 1\}$. The decision $d = 1$ means to buy a new PC. The utility of the new PC is ω. The utility of the old one is q.

The utility of the decision $d = 0$, to keep the old PC in the last year N, is $q_N + c_N$. Here $c_N = \tau_N - l_N$ is the price of refusing to buy a new PC. This includes the waiting losses τ minus the price l_N of new PC in the year N defined by (30). It is assumed that we abandon the old PC as soon as we obtain the new one. Therefore, one "wins" the price l of the new PC by using the old PC. The optimal decision in the last year N

$$d_N^* = \begin{cases} 1, & \text{if } \omega_N > q_N - c_N, \\ 0, & \text{if } \omega_N \leq q_N - c_N. \end{cases} \tag{33}$$

Denote by $u_{N-1}(\omega, q)$ the maximal expected utility in the year $N-1$.

$$u_{N-1}(\omega, q) = \max_d(d\omega + (1-d)(u_N(q_N) - c_{N-1})). \tag{34}$$

Here $u_N(q)$ is the maximal expected utility in the year N, if the utility of the old PC is q.

$$u_N(q) = \int_{-\infty}^{\infty} u_N(\omega, q) p_N(\omega) d\omega. \tag{35}$$

$p_N(\omega)$ is a prior probability density of ω defined by expression (28) at a time $t = N$.

$$q_N = \begin{cases} \omega_{N-1}, & \text{if } q_N < q_{N-1}^*, \\ q_{N-1}, & \text{if } q_N \geq q_{N-1}^*. \end{cases} \tag{36}$$

Here q_{N-1}^* is obtained from this equation

$$\omega_{N-1} = u_N(q_{N-1}^*) - c_{N-1}. \tag{37}$$

The optimal decision in the year $N-1$

$$d_{N-1}^* = \begin{cases} 1, & \text{if } \omega_{N-1} > u_N(q_N) - c_{N-1}, \\ 0, & \text{if } \omega_{N-1} \leq u_N(q_N) - c_{N-1}. \end{cases} \tag{38}$$

Denote by $u_{N-i}(\omega, q)$ the maximal expected utility in the year $N - i$, $i = 1, ..., N - 1$. Then

$$u_{N-i}(\omega, q) = \max_d (d\omega + (1-d)(u_{N-i+1}(q_{N-i+1}) - c_{N-i})). \tag{39}$$

Here $u_{N-i+1}(q)$ is the maximal expected utility in the year $N - i + 1$, if the utility of the old PC is q.

$$u_{N-i+1}(q) = \int_{-\infty}^{\infty} u_{N-i+1}(\omega, q) p_{N-i+1}(\omega) d\omega. \tag{40}$$

$p_{N-i+1}(\omega)$ is a prior probability density of ω defined by expression (28) at a time $t = N - i + 1$.

$$q_{N-i+1} = \begin{cases} \omega_{N-i}, & \text{if } q_{N-i+1} < q^*_{N-i}, \\ q_{N-i}, & \text{if } q_{N-i+1} \geq q^*_{N-i}. \end{cases} \tag{41}$$

q^*_{N-i} is obtained from the equation

$$\omega_{N-i} = u_{N-i+1}(q^*_{N-i}) - c_{N-i}. \tag{42}$$

The optimal decision in the year $N - i$

$$d^*_{N-i} = \begin{cases} 1, & \text{if } \omega_{N-i} > u_{N-i+1}(q_N - i + 1) - c_{N-i}, \\ 0, & \text{if } \omega_{N-i} \leq u_{N-i+1}(q_{N-i+1}) - c_{N-i}. \end{cases} \tag{43}$$

Here the decision functions d^*_{N-i} depend on two variables ω, q defining the corresponding utilities of the new and the old computer. That is difficult for both the calculations and grafical representations.

One applies the discrete approximation such as in Sect. 5.2. The optimal decision functions depend on two variables ω and q. Therefore, one needs K times more calculations to obtain the same accuracy as in the single marriage case (see Sect. 5.2).

To solve real life problems the "directly unobservable" factors should be included into the "Buy-a-PC" model. For example, one can estimate the expected life-span $T(s)$ of PC, given the impression s, by the Bayesian formula

$$T = \int_{-\infty}^{\infty} \tau p(\tau|s) d\tau, \tag{44}$$

$$p(\tau|s) = \frac{p(s|\tau)p(\tau)}{p_s(s)}. \tag{45}$$

Here τ is the life-span of PC and s is the user's impression about its reliability. The impression s depends on the warranty, the reputation of the manufacturer and on some subjective factors, too.

7 Optimal Ordering of Supplies

The important application a of sequential statistical decisions is optimal ordering of supplies. Consider, for example, same computer shop that orders d units of RAM (Random Access Memory) at fixed times $t = 1, ..., N$ dividing the time into N stages of unit length each.

The recommended decision is in Fig. 15. A set of decision functions are in Fig. 16.

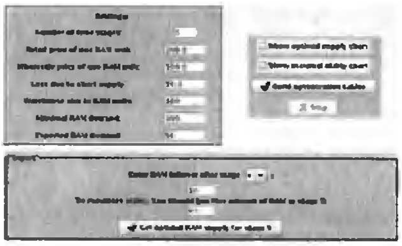


Fig. 15. The recommended decision of "Order-Ram"

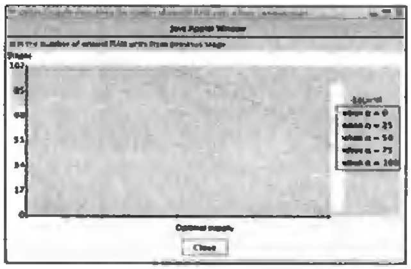
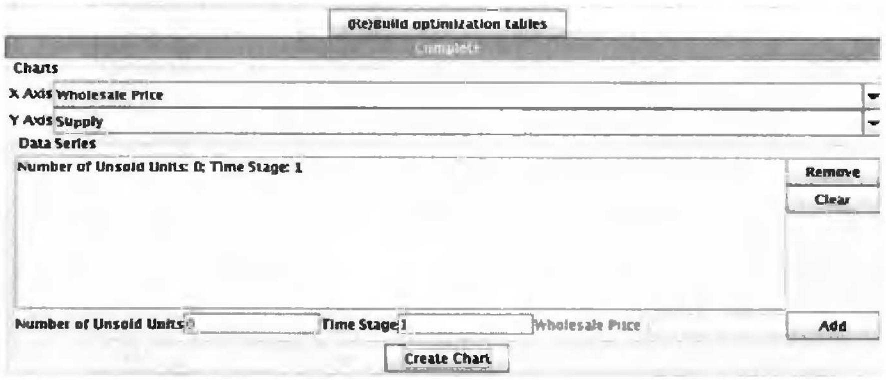


Fig. 16. A set of decision functions of "Order-Ram"

In this example the wholesale price and the demand of RAM units is fixed. Usually we have just some statistical data about the distribution of wholesale prices. The demand is defined by market elasticity. The elasticity shows how the demand depends on the retail prices. The updated "genOrder-RAM" model regards both: the uncertainty of wholesale prices and the elasticity of market. The results are in Figs. 17, 18, and 19. Figures 20 and 21 show how the utility function and the optimal supply depends on the wholesale price.



(Re)Build optimization tables

Complete

Charts

X Axis Wholesale Price

Y Axis Supply

Data Series

Number of Unsold Units: 0; Time Stage: 1

Number of Unsold Units 0 Time Stage 1 Wholesale Price

Fig. 17. Initial data of "genOrder-RAM"

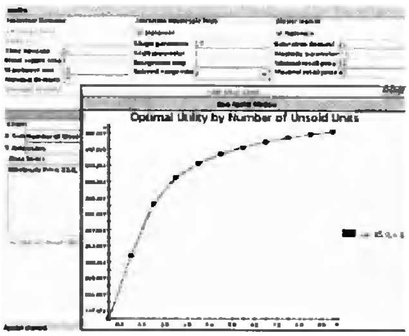


Fig. 18. Utility-to-unsold units, "genOrder-RAM"

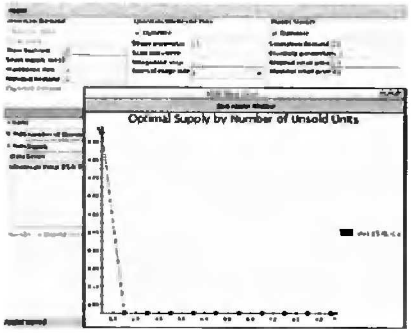


Fig. 19. Supply-to-unsold units, "genOrder-RAM"

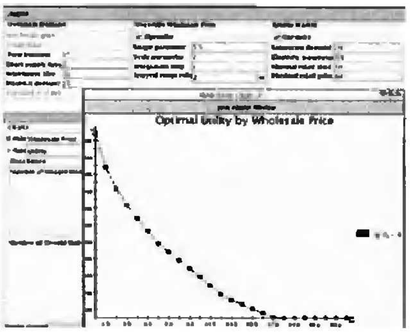


Fig. 20. Utility-to-wholesale price, "genOrder-RAM"

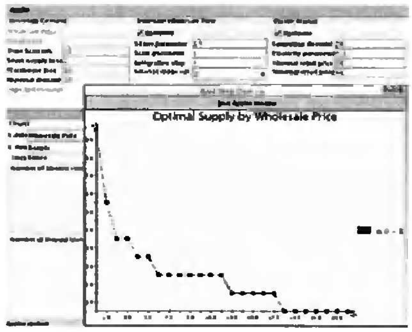


Fig. 21. Supply-to-wholesale price, "genOrder-RAM"

7.1 Uncertain Demand

One optimizes the supply d when the demand ω is unknown. Denote by ω a number of RAM units demanded during one stage. Denote the retail price of one RAM unit by c_r and the wholesale price by $c_w \leq c_r$. The utility function-the profit-at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b-\omega), & \text{if } b < \omega. \end{cases} \tag{46}$$

Here $c = c_r - c_w$ and $b = d + q$, q is the number of unsold units. L is the loss due to short supply including fines for breaking supply contracts and/or damages to reputation as an reliable retailer. All these parameters correspond to the stage $N-i$, $i = 0, 1, ..., N-1$, meaning that $\omega = \omega_{N-i}$, $d = d_{N-i}, q = q_{N-i}$ where ω_{N-i} is the demand at the stage $N-i$, $d = d_{N-i} \in D$ is the supply belonging to a feasible set $D = D_{N-i}$ at this stage, and $q = q_{N-i-1} = d_{N-i-1} + q_{N-i-2} - \omega_{N-i-1}$ is the number of unsold RAM units, left from the previous stage $N-i-1$. Here and later we omit the lower indices to make expressions shorter.

Denote by $p_{N-i}(\omega)$ the probability of demand $\omega_{N-i} = \omega$ at the stage $N-i$. Usually the demand $\omega = 0, 1, 2, ...$ during the time τ is described by the Poisson distribution [Tij94]

$$p_{N-i}(\omega) = e^{-\lambda\tau} \frac{(\lambda\tau)^\omega}{\omega!}. \tag{47}$$

Considering stages of unit time $\tau = 1$ and

$$p_{N-i}(\omega) = e^{-\lambda} \frac{\lambda^\omega}{\omega!}, \tag{48}$$

where $\lambda = \lambda_{N-i}$ is the expected demand at the stage $N-i$. Then the expected utility at the stage $N-i$, $i = 0, ..., N-1$

$$u_{N-i}(d,q) = \sum_\omega v_{N-i}(\omega, d, q)\ p_{N-i}(\omega). \tag{49}$$

The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N} \sum_\omega v_N(\omega, d, q)\ p_N(\omega), \tag{50}$$

where $q = q_{N-1}$ is a number of unsold RAM units available at the beginning of the N-th stage[4], $d = d_N$ is a number of RAM units ordered at the beginning of the N-th stage, $\omega = \omega_N$ is the demand at the stage N, D_N is a set of feasible supplies at the stage N.

Denote by $d_N(q)$ the optimal decision function maximizing the expected utility (50) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N-1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}} (u_{N-1}(d,q) + \sum_\omega u_N(q)\ p_N(\omega)). \tag{51}$$

Note, that the meaning of parameter q depends on the index of the function u, for example, $q = q_{N-1}$, if the index is N, and $q = q_{N-2}$, if this index is $N-1$, etc.

The maximum of (51) is obtained by the optimal decision function $d_{N-1}(q)$ where $q = q_{N-2}$.

Extending (51) to $i = 0, ..., N-1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}} (u_{N-i}(d,q) + \sum_\omega u_N(q)\ p_{N-i+1}(\omega)). \tag{52}$$

[4] The meaning of the argument q of the function $u_N(q)$ is defined by the index N, for example, $q = q_{N-1}$, if this index is N, etc.

Solving recurrent equations (50)-(52) one defines the sequence of optimal decision functions $d_{N-i}(q)$, $i = 0, ..., N-1$ and the optimal expected utility functions $u_{N-i}(q)$. This is done for all possible remaining RAM units q and for all stages $i = 1, ..., N-1$.

In the software terms that means calculating recurrently the sequence $i = 0, 1, ..., N-1$ of arrays that shows how the optimal supply $d = d_{N-i}(q)$ and the maximal profit $u = u_{N-i}(q)$ depend on the unsold RAM units q, left from the previous stages. This way one reduces computing time considerably, because array access operations need much less time as compared to multidimensional optimization.

7.2 Uncertain Wholesale Price

Denote by $p^c_{N-i}(c_w) = P\{c_w(N-i) = c_w\}$ a probability that the wholesale price $c_w(N-i)$ at the stage $N-i$ is c_w. Denote by $C_w(N-i)$ the expected wholesale price c_w at the stage $N-i$. Assuming the Gaussian distribution of the wholesale price logarithm, one obtains the lognormal density function [Tij94]

$$p^c_{N-i}(c_w) = \frac{1}{\alpha c_w \sqrt{2\pi}} e^{-\frac{(ln(c_w)-\gamma)^2}{2\alpha^2}}, \tag{53}$$

where c_w means $c_w(N-i)$, $\alpha > 0$ is the shape parameter, and γ is the scale parameter. The expected value $C_w(N-i)$ and the standard $S_w(N-i)$ of the lognormal distribution

$$C_w(N-i) = e^{\gamma+\alpha^2/2}, \tag{54}$$

$$S_w(N-i) = e^{\alpha^2} - 1. \tag{55}$$

The lognormal density is unimodal with maximum at $c_w = exp(\gamma - \alpha^2)$.

In the case of random wholesale prices $c_w = c_w(N-i)$, the profit at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b-\omega), & \text{if } b < \omega. \end{cases} \tag{56}$$

Here $c = c_r - c_w(N-i)$, $q = q_w(N-i-1)$ is the number of unsold units. Then the expected utility at the stage $N-i$, $i = 0, ..., N-1$

$$u_{N-i}(d, q) = \sum_\omega \int_0^\infty v_{N-i}(\omega, d, q)\ p^c_{N-i}(c_w)dc_w\ p_{N-i}(\omega). \tag{57}$$

The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N} \sum_\omega \int_0^\infty v_N(\omega, d, q)\ p^c_N(c_w)dc_\omega\ p_N(\omega), \tag{58}$$

where $q = q_{N-1}$ is a number of unsold RAM units left at the beginning of N-th stage, $c_w = c_w(N)$ is the wholesale price at the stage N.

Denote by $d_N(q)$ the optimal decision function maximizing the expected utility (58) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N-1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}} (u_{N-1}(d, q) +$$
$$\sum_{\omega} \int_0^{\infty} u_N(q)\ p_N^c(c_w) dc_w\ p_N(\omega)). \tag{59}$$

That is achieved by the optimal decision function $d_{N-1}(q)$ where $q = q_{N-2}$.

Extending to $i = 0, ..., N-1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}} (u_{N-i}(d, q) +$$
$$\sum_{\omega} \int_0^{\infty} u_N(q)\ p_{N-i+1}^c(c_w) dc_w\ p_{N-i+1}(\omega)). \tag{60}$$

Solving recurrent equations (58)-(60) one defines the sequence of optimal decision functions $d_{N-i}(q)$, $i = 0, ..., N-1$ and the optimal expected utilities $u_{N-i}(q)$. This is done for all possible numbers of RAM units q available at the beginning of stages $i = 1, ..., N-1$.

7.3 Market Elasticity

We call by Market Elasticity the relation $\Omega = \Omega(c_r)$ of expected demand Ω to the retail price $c_r \in C$, where C is a set of feasible retail prices. A simple approximation is the exponential function

$$\Omega(c_r) = be^{-\beta c_r}, \tag{61}$$

where $b = b_{N-i}$ is the saturation demand, $\beta = \beta_{N-i}$ is the elasticity parameter, $c_r = c_r(N-i) \in C_{N-i}$ is the retail price, and C_{N-i} is a set of feasible retail prices, all at the stage $N-i$.

In this case the profit function at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q, c_r) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b - \omega), & \text{if } b < \omega. \end{cases} \tag{62}$$

Here c denotes the difference $c = c_r(N-i) - c_w(N-i)$ of retail and wholesale prices at the stage $N-i$. The expected utility at the stage $N-i$, $i = 0, ..., N-1$

$$u_{N-i}(d, q, c_r) = \sum_{\omega} \int_0^{\infty} v_{N-i}(\omega, d, q)\ p_{N-i}^c(c_w) dc_w\ p_{N-i}(\omega), \tag{63}$$

where $c_r = c_r(N-i)$. The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N, c_r \in C_N} \sum_{\omega} \int_0^{\infty} v_N(\omega, d, q, c_r)\; p_N^c(c_w) dc_w \; p_N(\omega). \tag{64}$$

Denote by $d_N(q) = (d_N(q), c_N(q))$, where q denotes q_{N-1}, the two-dimensional optimal decision function maximizing the expected utility (64) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N-1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}, c_r \in C_{N-1}} (u_{N-1}(d, q, c_r) +$$
$$\int_0^{\infty} \sum_{\omega} u_N(q)\; p_N^c(c_w) dc_w \; p_N(\omega)). \tag{65}$$

That is achieved by the two-dimensional optimal decision function $d_{N-1}(q) = (d_{N-1}(q), c_{N-1}(q))$ where q means q_{N-2}.

Extending (65) to $i = 0, ..., N-1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}, c_r \in C_{N-i}} (u_{N-i}(d, c_r, q) +$$
$$\sum_{\omega} \int_0^{\infty} u_N(q)\; p_{N-i+1}^c(c_w) dc_w \; p_{N-i+1}(\omega)). \tag{66}$$

Solving recurrent equations (64)-(66) one defines the sequence of decision functions $d_{N-i}(q) = (d_{N-i}(q), c_{N-i}(q))$ $i = 0, ..., N-1$ and the optimal expected profits $u_{N-i}(q)$. This is done for all possible numbers q of RAM units q available at the beginning of stages $i = 1, ..., N-1$.

In the case of market elasticity, to represent maximal profits $u_{N-i}(q)$ and two-dimensional optimal decisions $d_{N-i}(q) = (d_{N-i}(q), c_{N-i}(q))$ $i = 0, ..., N-1$ as functions of remaining RAM units q one needs to calculate a sequence of N arrays defining the sequence of two-dimensional decision functions. This is difficult, therefore, one starts from the simplest case of uncertain demand (see Sect. 7.1), as usual.

Note that in the actual calculations of expressions (58)-(66) the integrals are replaced by sums and the densities by probabilities using expressions similar to those in the Sect. 5.2.

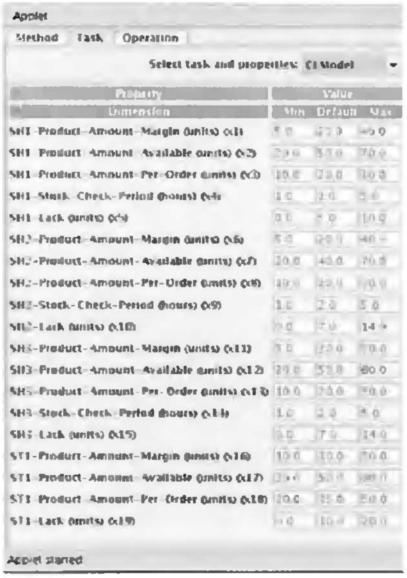
In the demonstration version the "horizon" N means the estimated end of business. In the expert version one uses the "moving horizon" $N_o < N$, repeating the optimization after each stage. This way one simplifies calculations and updates data.

The recurrent equations (50)-(52) include Bride problems as special cases with only two feasible decisions $D = \{0, 1\}$ and different utility functions (46).

8 "Logistic Model"

In the Logistic Model the heuristic decisions are regarded. The search for the optimal parameters of these heuristics are performed using the Bayesian

Heuristic Approach [Moc00]. The results are illustrated by a set of figures. Figure 22 shows the input data. Figure 23 shows the results of "Logistic Model" parameters optimization. Figure 24 shows Step 1. Initialize "Logistic Model". Figure 25 shows Step 2. Modify Data of "Logistic Model". Figure 26 shows Step 3. Simulation of "Logistic Model". Figure 27 shows Step 4. Analyzis of "Logistic Model". Figure 29 shows the contents of "Help" file.

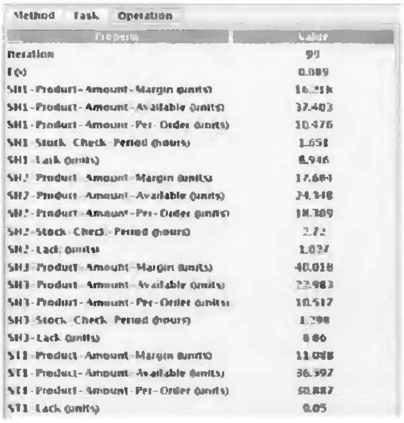


Select task and properties: CI Model

Property	Value		
Dimension	Min	Default	Max
SH1-Product-Amount-Margin (units) (x1)	5.0	[illegible]	[illegible]
SH1-Product-Amount-Available (units) (x2)	[illegible]	51.0	70.0
SH1-Product-Amount-Per-Order (units) (x3)	10.0	20.0	30.0
SH1-Stock-Check-Period (hours) (x4)	1.0	2.0	5.0
SH1-Lack (units) (x5)	0.0	5.0	10.0
SH2-Product-Amount-Margin (units) (x6)	5.0	20.0	40.0
SH2-Product-Amount-Available (units) (x7)	20.0	45.0	70.0
SH2-Product-Amount-Per-Order (units) (x8)	10.0	20.0	30.0
SH2-Stock-Check-Period (hours) (x9)	1.0	2.0	5.0
SH2-Lack (units) (x10)	0.0	7.0	14.0
SH3-Product-Amount-Margin (units) (x11)	5.0	25.0	70.0
SH3-Product-Amount-Available (units) (x12)	[illegible]	55.0	80.0
SH3-Product-Amount-Per-Order (units) (x13)	10.0	20.0	30.0
SH3-Stock-Check-Period (hours) (x14)	1.0	2.0	4.0
SH3-Lack (units) (x15)	0.0	7.0	14.0
ST1-Product-Amount-Margin (units) (x16)	10.0	30.0	70.0
ST1-Product-Amount-Available (units) (x17)	[illegible]	[illegible]	[illegible]
ST1-Product-Amount-Per-Order (units) (x18)	10.0	15.0	30.0
ST1-Lack (units) (x19)	0.0	10.0	20.0

Applet started

Fig. 22. Input window, "Logistic Model"



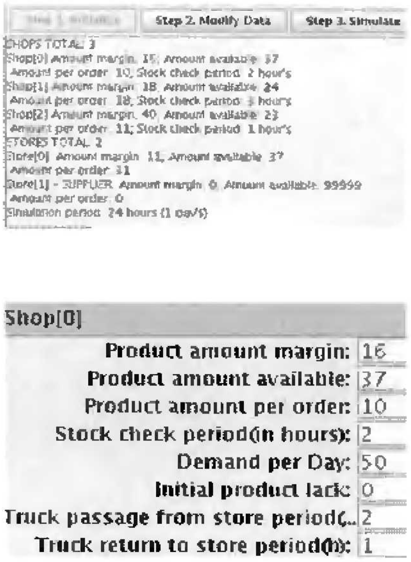
Property	Value
Iteration	93
f(x)	0.889
SH1-Product-Amount-Margin (units)	[illegible]
SH1-Product-Amount-Available (units)	37.403
SH1-Product-Amount-Per-Order (units)	10.476
SH1-Stock-Check-Period (hours)	1.651
SH1-Lack (units)	6.946
SH2-Product-Amount-Margin (units)	[illegible]
SH2-Product-Amount-Available (units)	[illegible]
SH2-Product-Amount-Per-Order (units)	[illegible]
SH2-Stock-Check-Period (hours)	[illegible]
SH2-Lack (units)	1.027
SH3-Product-Amount-Margin (units)	40.018
SH3-Product-Amount-Available (units)	22.983
SH3-Product-Amount-Per-Order (units)	10.517
SH3-Stock-Check-Period (hours)	1.298
SH3-Lack (units)	[illegible]
ST1-Product-Amount-Margin (units)	11.088
ST1-Product-Amount-Available (units)	36.397
ST1-Product-Amount-Per-Order (units)	[illegible]
ST1-Lack (units)	0.05

9 "Buy-a-Flat" by *C#*

All the examples are by Java. The reason is that Java software can run by different OS both in the form of applets (using users computer) and servlets (using server resources). The nearest alternative is *C#* (C-sharp). The C-sharp is a propriatory Microsoft software and runs by Windows. C-sharp can run by Unix, too, using "Mono" system. Hovewer there are no "applets" in C-sharp. Figure 28 shows the "Buy-a-Flat" example implemented by *C#* and "Mono".

10 Distance Studies

For the graduate level distance and class-room studies of the theory of optimal statistical decisions in the Internet environment a set of examples of

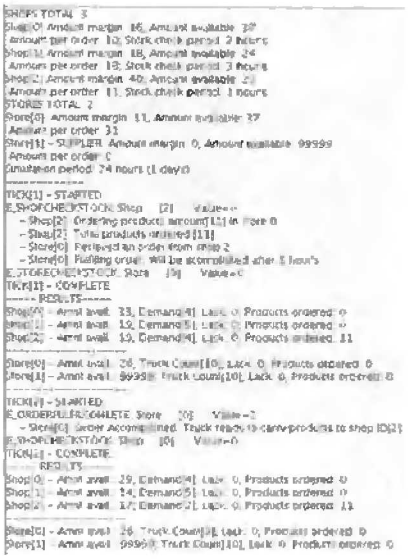


```
SHOPS TOTAL 3
Shop[0] Amount margin 15; Amount available 37
 Amount per order 10; Stock check period 2 hours
Shop[1] Amount margin 18; Amount available 24
 Amount per order 18; Stock check period 3 hours
Shop[2] Amount margin 40; Amount available 23
 Amount per order 11; Stock check period 1 hours
STORES TOTAL 2
Store[0] Amount margin 11; Amount available 37
 Amount per order 31
Store[1] - SUPPLIER Amount margin 0; Amount available 99999
 Amount per order 0
Simulation period 24 hours (1 days)
-------------
```

Shop[0]

Product amount margin:	16
Product amount available:	37
Product amount per order:	10
Stock check period(in hours):	2
Demand per Day:	50
Initial product lack:	0
Truck passage from store period(...	2
Truck return to store period(h):	1

Fig. 25. Step 2. Modify, "Logistic Model"



```
SHOPS TOTAL 3
Shop[0] Amount margin 16; Amount available 37
 Amount per order 10; Stock check period 2 hours
Shop[1] Amount margin 18; Amount available 24
 Amount per order 18; Stock check period 3 hours
Shop[2] Amount margin 40; Amount available 23
 Amount per order 11; Stock check period 1 hours
STORES TOTAL 2
Store[0] Amount margin 11; Amount available 37
 Amount per order 31
Store[1] - SUPPLIER Amount margin 0; Amount available 99999
 Amount per order 0
Simulation period 24 hours (1 days)
-------------
TICK[1] - STARTED
E_SHOPCHECKSTOCK Shop    [2]     Value=-
   - Shop[2] Ordering products amount[11] in Store 0
   - Shop[2] Total products ordered [11]
   - Store[0] Recieved an order from shop 2
   - Store[0] Fulfilling order. Will be accomplished after 1 hours
E_STORECHECKSTOCK Store    [0]     Value=0
TICK[1] - COMPLETE
------ RESULTS------
Shop[0] - Amnt avail 33, Demand[4], Lack 0, Products ordered 0
Shop[1] - Amnt avail 19, Demand[5], Lack 0, Products ordered 0
Shop[2] - Amnt avail 19, Demand[4], Lack 0, Products ordered 11
--------------------
Store[0] - Amnt avail 26, Truck Count[10], Lack 0, Products ordered 0
Store[1] - Amnt avail 99999, Truck Count[10], Lack 0, Products ordered 0
--------------------
TICK[2] - STARTED
E_ORDERFULFILCOMPLETE Store    [0]     Value=2
   - Store[0] Order Accomplished. Truck ready to carry products to shop ID[2]
E_SHOPCHECKSTOCK Shop    [0]     Value=0
TICK[2] - COMPLETE
------ RESULTS------
Shop[0] - Amnt avail 29, Demand[4], Lack 0, Products ordered 0
Shop[1] - Amnt avail 14, Demand[5], Lack 0, Products ordered 0
Shop[2] - Amnt avail 17, Demand[2], Lack 0, Products ordered 11
--------------------
Store[0] - Amnt avail 26, Truck Count[9], Lack 0, Products ordered 0
Store[1] - Amnt avail 99999, Truck Count[10], Lack 0, Products ordered 0
```

Fig. 26. Step 3. Simulate, "Logistic Model"

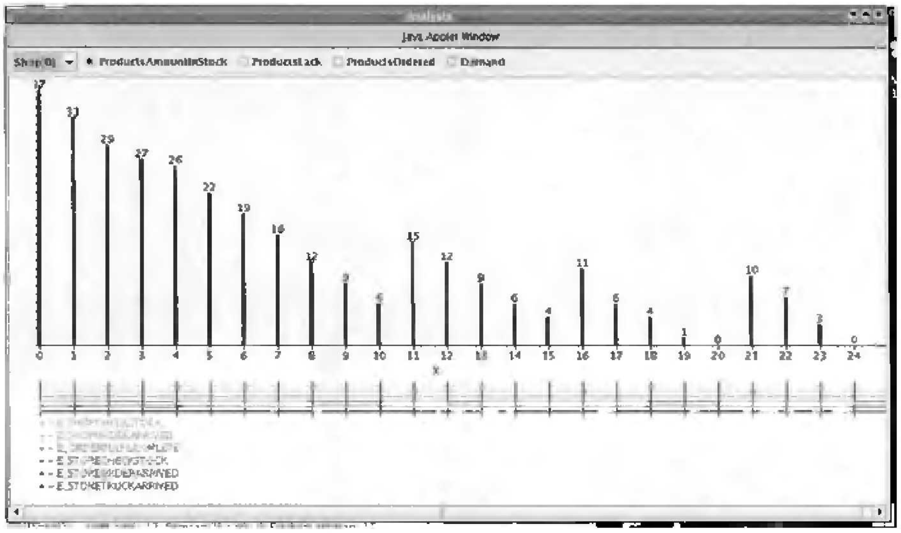
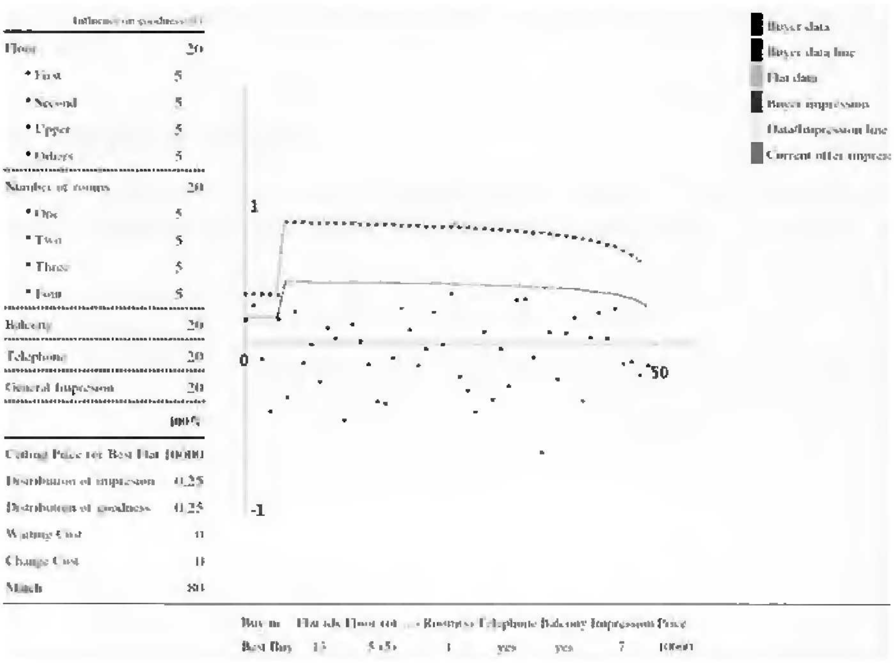


Fig. 27. Step 4. Analyze, "Logistic Model"

corresponding optimization models was implemented by platform independent Java applets and servlets. Here is the list of web-sites:
http : //pilis.if.ktlu.lt/˜jmockus
http : //optimum2.mii.lt/˜jonas2



Influence on goodness (%)	
Floor	20
• First	5
• Second	5
• Upper	5
• Others	5
Number of rooms	20
• One	5
• Two	5
• Three	5
• Four	5
Balcony	20
Telephone	20
General Impression	20
	100%
Ceiling Price for Best Flat	10000
Distribution of impression	0.25
Distribution of goodness	0.25
Waiting Cost	[illegible]
Change Cost	[illegible]
Match	[illegible]

Buy in	Flat ids	Floor (st)	Rooms	Telephone	Balcony	Impression	Price
Best Buy	15	5 (5)	1	yes	yes	7	10000

Fig. 28. "Buy-a-Flat" by C-sharp

http : //eta.ktl.mii.lt/˜mockus
http : //proin.ktu.lt/˜mockus
http : //mockus.us/optimum
The theoretical background and the complete description of the software is in the file 'stud2.pdf'. Examples are described in the section: "Discrete Optimization" for discrete optimization and applications of linear and dynamic programming.

All the results for international users are in English. Examples intended for Lithuanian universities are described in Lithuanian.

11 Conclusions

1. The growing power of internet presents new problems and opens new possibilities for distant scientific collaboration and graduate studies. Therefore some nontraditional ways for presentation of scientific results should be defined.
2. The chapter is a try to start a specific style designed for encouragement of new approaches to presentation of scientific results.
3. The objective of the chapter is to start the scientific collaboration with colleagues sharing similar ideas.

4. The examples of decision making models illustrate of two important well-known theretical topics: the Sequential Statistical Decisions, and the Bayesian Approach.
5. The results of optimization show the possibilities of some nontraditional ways of graduate studies in both the distance and the classroom cases.

References

[Bay83] Bayes, T.: An essay towards solving a problem in the doctrine of chances. Phil. Transactions of Royal Society, **53**, 370–418 (1783)

[Bel57] Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)

[DeG70] DeGroot, M.: Optimal Statistical Decisions. McGraw-Hill, New York (1970)

[Fin73] Fine, T.L.: Theories of Probability. Academic Press, New York (1973)

[Lin71] Lindley, D.: Bayesian Statistics: A Review. SIAM, Philadelphia (1971)

[Moc89a] Mockus, J.: Bayesian Approach to Global Optimization. Kluwer Academic Publishers, Dordrecht-London-Boston (1989)

[Moc89b] Mockus, J.: Bayesian approach to global optimization and application to constrained and multi-objective problems. In: Abstracts, the Fifth International Conference on Stochastic Programming in Ann Arbor, August 13-18, 1989, (1989)

[Moc00] Mockus, J.: A Set of Examples of Global and Discrete Optimization: Application of Bayesian Heuristic Approach. Kluwer Academic Publishers (2000)

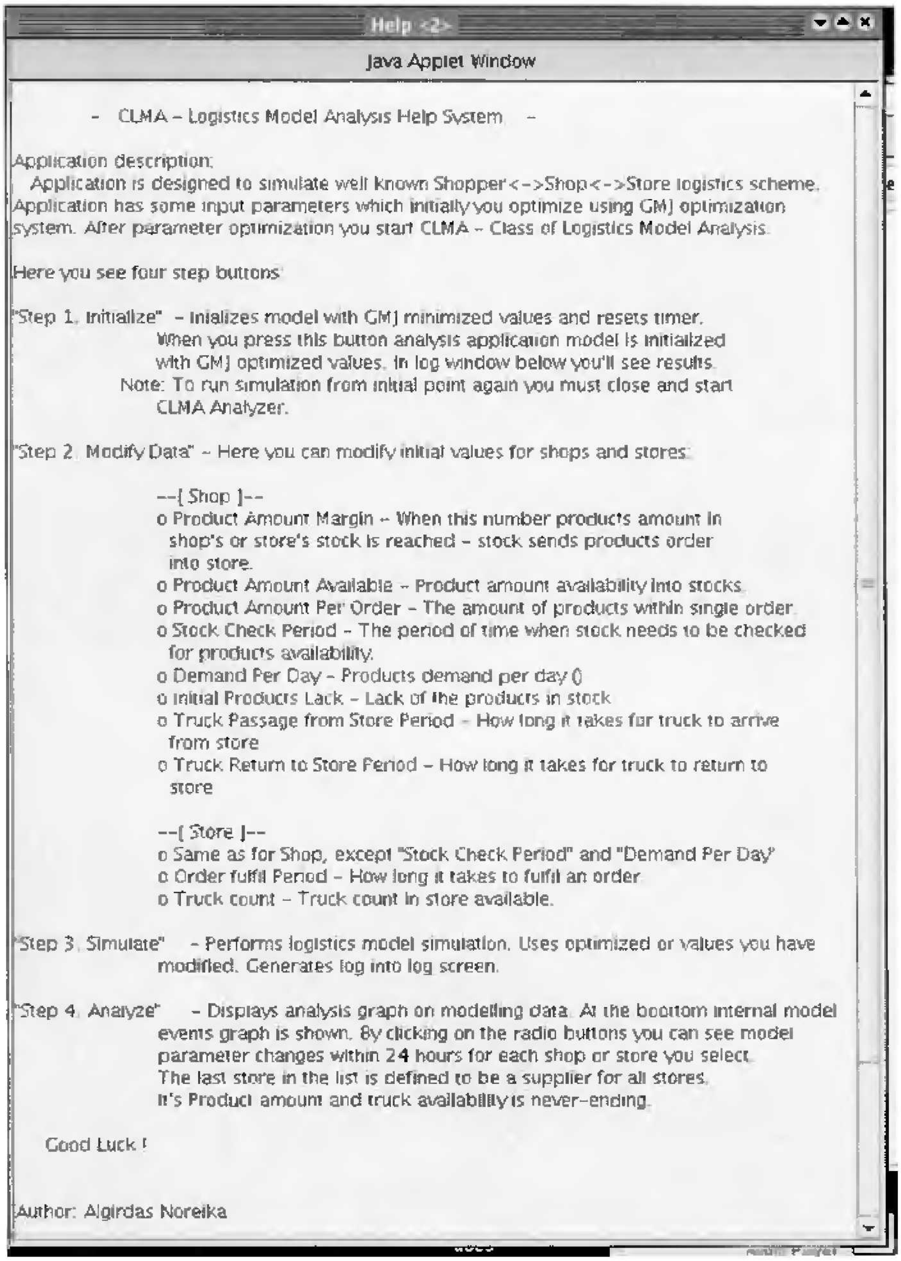
[Sav54] Savage, L.: Foundations of Statistics. Wiley, New York (1954)

[Tij94] Tijms, H.: Stochastic Models. An Algorithmic Approach. Wiley, New York (1994)

[Wal47] Wald, A.: Sequential Analysis. J. Wiley, New York (1947)

[Wal50] Wald, A.: Statistical Decision Functions. J. Wiley, New York (1950)

[Zil86] Zilinskas, A.: Global Optimization: Axiomatic of Atatistical Models, Algorithms and Their Applications. Mokslas, Vilnius, Lithuania (1986) in Russian



- CLMA – Logistics Model Analysis Help System -

Application description:
Application is designed to simulate well known Shopper<->Shop<->Store logistics scheme. Application has some input parameters which initially you optimize using GMJ optimization system. After parameter optimization you start CLMA – Class of Logistics Model Analysis.

Here you see four step buttons:

"Step 1. Initialize" – inializes model with GMJ minimized values and resets timer. When you press this button analysis application model is initialized with GMJ optimized values. In log window below you'll see results.
Note: To run simulation from initial point again you must close and start CLMA Analyzer.

"Step 2. Modify Data" – Here you can modify initial values for shops and stores:

--[Shop]--
o Product Amount Margin – When this number products amount in shop's or store's stock is reached – stock sends products order into store.
o Product Amount Available – Product amount availability into stocks.
o Product Amount Per Order – The amount of products within single order.
o Stock Check Period – The period of time when stock needs to be checked for products availability.
o Demand Per Day – Products demand per day ()
o Initial Products Lack – Lack of the products in stock
o Truck Passage from Store Period – How long it takes for truck to arrive from store
o Truck Return to Store Period – How long it takes for truck to return to store

--[Store]--
o Same as for Shop, except "Stock Check Period" and "Demand Per Day"
o Order fulfil Period – How long it takes to fulfil an order
o Truck count – Truck count in store available.

"Step 3. Simulate" – Performs logistics model simulation. Uses optimized or values you have modified. Generates log into log screen.

"Step 4. Analyze" – Displays analysis graph on modelling data. At the bootom internal model events graph is shown. By clicking on the radio buttons you can see model parameter changes within 24 hours for each shop or store you select. The last store in the list is defined to be a supplier for all stores. It's Product amount and truck availability is never-ending.

Good Luck !

Author: Algirdas Noreika

Fig. 29. Help, "Logistic Model"

The Problem of Visual Analysis of Multidimensional Medical Data

Jolita Bernatavičienė, Gintautas Dzemyda, Olga Kurasova, Virginijus Marcinkevičius, and Viktor Medvedev

Institute of Mathematics and Informatics, Akademijos 4, Vilnius 08663, Lithuania
{JolitaB, Dzemyda, Kurasova, VirgisM, ViktorM}@ktl.mii.lt

Summary. We consider the problem of visual analysis of the multidimensional medical data. A frequent problem in medicine is an assignment of a health state to one of the known classes (for example, healthy or sick persons). A particularity of medical data classification is the fact that the transit from the normal state to diseased one is often not so conspicuous. From the table of the parametric medical multidimensional data, it is difficult to notice which objects are similar, which ones are different, i.e., which class they belong to. Therefore it is necessary to classify the multidimensional data by various classification methods. However, classification errors are inevitable and the results of classification in medicine must be as correct as possible. That is why it is advisable to use different types of data analysis methods, for example, in addition to visualize the multidimensional data (to project to a plane). A visual analysis allows us to estimate similarities and differences of objects, a partial assignment to one or another class in simple visual way. However, the shortcoming of this analysis is the fact that while projecting multidimensional data to a plane, a part of the information is inevitably lost. Thus, one of the agreeable methods is a combination of classification and visualization methods. This synthesis lets us to obtain a more objective conclusions on the analysed data. The results, obtained by the integrated method, proposed in this chapter, can help medics to preliminary diagnose successfully or have some doubt on the former diagnosis.

1 Introduction

A frequent problem in medicine is an assignment of a health state to one of the known classes (for example, healthy or sick persons). Such an assignment is made by medics as usual. Various data mining methods and software for the decision support in medicine are developed [JLPB02, LKZ97, Adl03]. A particularity of medical data classification is the fact that the transit from the normal state to diseased one is often inconspicuous. It is of utmost importance to determine the bound of the transit from one state to another. In the general case, it does not suffice to determine the bounds by a function, which separates classes. We propose the way to find a certain area of parameter values, which correspond to the patients that should be thoroughly examined. Our method allows to present this area in two-dimensional form, that is preferable for visual decision on the state of health of the patients.

An array of the parameters $x_1, x_2, \ldots, x_n$ characterises a disease (health state), therefore it is possible to form n-dimensional vectors $X = (x_1, x_2, \ldots, x_n)$, which correspond to patients. Let a set of the analysed data consists of m objects: $X^1, X^2, \ldots, X^m$, where $X^i = (x_1^i, x_2^i, \ldots, x_n^i)$, $i = 1, \ldots, m$. The problem of classification of multidimensional data should be solved. Let there are two classes C_1 and C_2. The goal is to assign the vectors $X^1, X^2, \ldots, X^m$ to one of the classes either C_1 or C_2. In the general case, the classification to several classes $(C_1, C_2, \ldots, C_v)$ may be investigated. The classification errors are inevitable. It is a important problem in medicine, because, if a sick person is assigned to the class of healthy persons, the outcome can be fatal. The visual analysis of such data is possible as well as using classifiers [SIL01, OM94]. There are a lot of classical visualization methods (see surveys [Kas97, GHP02, HG02, KW03]) and classification methods (see surveys [Dun03, MST94]). We propose here a method, where the results of classification of multidimensional data are integrated into the visualization of the analysed data.

This chapter is organized as follows. In Sect. 2, sets of analysed medical data are described. In Sect. 3, the classification and visualization methods are surveyed, and the quantitative criteria of mapping are presented. The experimental results are presented in Sect. 4. The final section draws conclusions.

2 Description of the Medical Data Sets

Two medical data groups have been analysed:

- physiological fractal dimension data set,
- ophthalmologic data set.

Physiological fractal dimension data. This data set is developed in Kaunas Medical University in cooperation with Kaunas University of Technology. This set consists of three groups: ischemic heart diseased patients (1) (61 items), healthy persons (not going in for sports) (2) (110 items), sportsmen (3) (161 items). Total number of items is equal to 332. The aim of the analysis of fractal dimensions is to estimate the relation among three functional elements of the human organism – periphery, regulation, and supplying systems. The following parameters that describe the human functional state, were measured: heart rate (HR), interval in electrocardiogram from point J to the end T of the wave (JT interval), systolic blood pressure (SBP), diastolic blood pressure (DBP), and ratios between some parameters (SBP-DBP)/SBP, JT/RR (RR=60/HR). The data set is prepared as follows:

1. These six numeric parameters are fixed from a provocative incremental bicycle ergometry stress test at certain time moments.

2. The set of the obtained numeric data on each parameter are interpolated by cubic spline. Thus, we get a graphical chart for each of six parameters.
3. Fractal dimensions (capacity, information and correlation) are computed for these graphical charts in a way below:
   - let analyze a regular grid (ε is size of grid cell), superimposed on the graphical charts;
   - the number n_i of "occupied" pixels, is counted for each grid cell; it is divided by the total number N of pixels, thus, we have $P_i(\varepsilon) = n_i/N$;
   - let us describe the information function $I \equiv -\sum_{i=1}^{N_\varepsilon} P_i(\varepsilon)\log[P_i(\varepsilon)]$, where N_ε is the number of "occupied" cells. Thus, the information dimension is $d_{\text{inf}} \equiv -\lim\limits_{\varepsilon\to 0}\frac{I}{\log(\varepsilon)} = \lim\limits_{\varepsilon\to 0}\sum_{i=1}^{N_\varepsilon}\frac{P_i(\varepsilon)\log[P_i(\varepsilon)]}{\log(\varepsilon)}$; if $I = \log N_\varepsilon$, the capacity dimension is obtained, if $I = \log\sum_i(P_i(\varepsilon))^2$, the correlation dimension is obtained.

Thus, the total number of the parameters is 18: 6 parameters for each fractal dimension. In the paper [BBA+05], it has been shown that the information dimension is the most informative: the fractal dimension data were classified for each dimension separately using the feedforward neural network, trained by the back-propagation algorithm; the best classification results (87%) were obtained when the vectors of the information dimension parameters were analysed; when the vectors of other dimension (capacity and correlation) parameters were classified, the classification accuracy was worse (only 70% in the best case). Therefore, the vectors of the information dimension parameters are analysed in our investigation. The dimensionality n of the data is equal to 6.

Ophthalmologic data. The base for research of ophthalmologic data is the project "Information technologies for human health – clinical decision support (e-Health), IT Health", supported the Lithuanian State Science and Studies Foundation.

The analysed object is the image of fundus of eyes. The fundus of eye is presented in Fig. 1.

27 parameters of the fundi of eyes have been measured. There are four groups of parameters:

(a) parameters of optic nerve discs (OND): major axis of OND ellipse (x_1), minor axis (x_2), semimajor axis (x_3), semiminor axis (x_4), horizontal diameter (x_5), vertical diameter (x_6), area (x_7), eccentricity (x_8), and perimeter (x_9);
(b) parameters of excavation (EKS) (excavation is a degenerated part of OND): major axis of EKS ellipse (x_{10}), minor axis (x_{11}), semimajor axis (x_{12}), semiminor axis (x_{13}), horizontal diameter (x_{14}), vertical diameter (x_{15}), area (x_{16}), eccentricity (x_{17}), and perimeter (x_{18});
(c) ratios between various OND, EKS, NRK parameters (neuroretinal rim (NRK) is an OND part that is not degenerated): ratio of EKS and OND horizontal diameters (x_{19}), ratio of EKS and OND vertical diameters

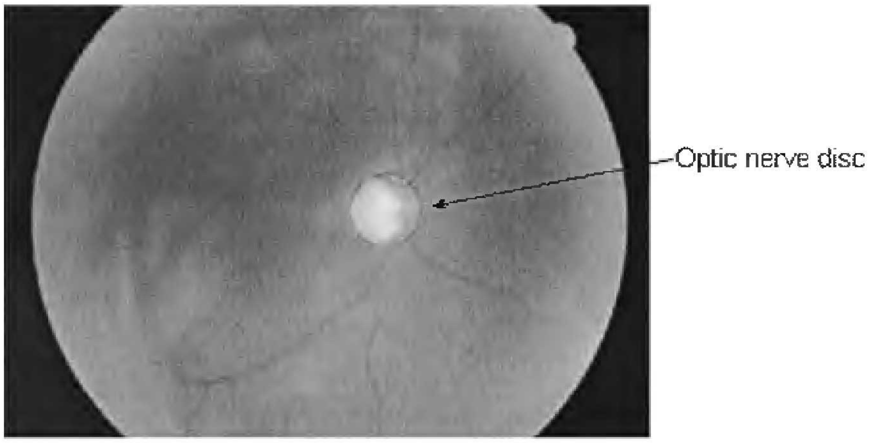


Fig. 1. Fundus of eye

(x_{20}), NRK area (x_{21}), ratio of NRK and OND areas (x_{22}), ratio of EKS and OND (x_{23});

(d) thickness of NRK parts: inferior disc sector (x_{24}), superior disc sector (x_{25}), nasal disc sector (x_{26}) and temporal disc sector (x_{27}).

Three groups of items are investigated: vectors, corresponding to the healthy eyes (18 items); vectors, corresponding to the eyes, damaged by glaucoma (24 items).

3 Data Mining Methods

In this chapter, some groups of data mining methods are used: (1) classification methods, (2) visualization (dimensional reduction) methods.

3.1 Classification Methods

Some classification methods are used: Naïve Bayes [RS03], Classification trees [BFOS84], kNN classifier [Dun03], and Support Vector Machine classifier [CS03].

Naïve Bayes classifier is based on the Bayes probability rule. Suppose we have items, assigned to some classes C_j, $j = 1, \ldots, \nu$ (here ν is the number of classes). Our task is to classify new items as they arrive, i.e., to decide to which class label they belong, based on the currently exiting objects. Suppose that item X has n independent attribute values $\{x_1, x_2, \ldots, x_n\}$. We need to calculate the posterior probability $P(C_j|X) = \frac{P(X|C_j)P(C_j)}{P(X)}$, i.e., the probability that X belongs to C_j. Here $P(C_j)$ is the prior probability of class C_j; $P(X|C_j)$ is the conditional probability, which can be rewritten as

$P(X|C_j) = \prod_{k=1}^{n} P(x_k|C_j)$; $P(X)$ is the likelihood that X is in each class, this can be done by finding the likelihood that the item is in each class and adding all these values. The posterior probability $P(C_j|X)$ is found for each class. We label a new item with a class C_j that achieves the highest posterior probability. This technique does not handle continuous data, but dividing the continuous values into ranges could be used to solve this problem [RS03].

Using a classification tree technique, a tree is constructed to model the classification process. Classification tree methods are a good choice when the data mining task is classification or prediction of outcomes and the goal is to generate rules that can be easily understood and explained. Once the tree is built, it is applied to each item in the data set and results in a classification for that item. A classification tree describes a tree structure wherein leaves represent classifications and branches represent conjunctions of features that lead to those classifications. A classification tree can be learned by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner. The recursion is completed when splitting is either non-feasible, or a singular classification can be applied to each element of the derived subset [BFOS84].

The k nearest neighbours (kNN) technique assumes that the entire training set includes not only the data in the set but also the desired classification for each item. The training data become a model. When a classification is to be made for a new item, its distance to each item in the training set must be determined. Only the k closest entries in the training set are considered further. The new item is then placed in the class that contains most items from this set of k closest items. The kNN technique is extremely sensitive to the value of k [Dun03].

Support Vector Machines (SVM) are a set of related supervised learning methods, applicable to both classification and regression. When used for classification, the SVM algorithm creates a hyperplane that separates the data into two classes with the maximum-margin. Given training examples, labelled either "yes" or "no", the maximum-margin hyperplane splits the "yes" and "no" training examples, so that the distance from the closest examples (the margin) to the hyperplane is maximized [CS03].

3.2 Visualization – Projection Methods

Visualization is one of the ways of presenting the multidimensional data in a visual form, allowing the human being to gain insight into the data, draw conclusions, and directly interact with the data. Various methods may be used for the visualization of the multidimensional data. They may be grouped into such categories: direct visualization methods, projection methods, and cluster methods. In the direct visualization methods (such as scatter plots, parallel coordinates, grand tour, Chernoff faces, star glyphs, dimensional stacking, etc.), each parameter of a multidimensional object is presented in a visual form. There are not any formal mathematical criteria, that must be optimised with a view to obtain an optimal solution. The drawback of these methods is that, using these methods, it is complicated to comprehend the data structure, it is almost impossible, when large data sets or data of large dimensions are analysed. It is easier to comprehend and to interpret the results, obtained by the projection methods, where multidimensional data are presented onto a lower dimension space so that certain properties of the structure of the data set were preserved as faithfully as possible. The projection can be used to visualise the data set if a sufficiently small output dimensionality is chosen. Here, it is necessary to optimise a certain criterion. There are linear and nonlinear projection methods. The principal component analysis [Tay03], and projection pursuit [FT74] are linear projection methods; while multidimensional scaling [BG97], and principal curves [HS89] are nonlinear ones. A more precise data structure is preserved using nonlinear projection methods. Nevertheless, the projection errors are inevitable. It is necessary to look for the ways of minimizing these projection errors. At the beginning of the large data set analysis, it is necessary to cluster data, followed afterwards by the visualization of a member of the cluster. Artificial neural networks (ANN) may be used to visualise the multidimensional data, too. They are often combined with the classical projection methods.

Multidimensional scaling (MDS) refers to a group of methods that are widely used. The starting point of MDS is a matrix consisting of pairwise dissimilarities of the entities. In general, the dissimilarities need not be distances in the mathematically strict sense. There exists a multitude of variants of MDS with slightly different cost functions and optimisation algorithms.

Let us have vectors $X^i = (x_1^i, x_2^i, \dots, x_n^i)$, $i = 1, \dots, m (X^i \in R^n)$. The pending problem is to get the projection of these n-dimensional vectors X^i, $i = 1, \dots, m$ onto the plane R^2. Two-dimensional vectors $Y^1, Y^2, \dots, Y^m \in R^2$ correspond to them. Here $Y^i = (y_1^i, y_2^i)$, $i = 1, \dots, m$. Denote the distance between the vectors X^i and X^j by d_{ij}^*, and the distance between the corresponding vectors in the projected space (Y^i and Y^j) by d_{ij}. In our case, the initial dimensionality is n, and the resulting one is 2.

The goal of projection in the metric multidimensional scaling (MDS) is to optimise the projection so that the distances between the items in the lower-dimensional space would be as close to the original distances as possible. If the square-error cost is used, the objective function (stress) to be minimized can be written as

$$E_{MDS} = \sum_{i,j=1, i<j}^{m} w_{ij} (d_{ij}^* - d_{ij}). \tag{1}$$

Weights w_{ij} can be as follows:

$$w_{ij} = \frac{1}{\sum_{k,l=1,k<l}^{m} (d_{kl}^*)^2}, \quad w_{ij} = \frac{1}{d_{ij}^* \sum_{k,l=1,k<l}^{m} d_{kl}^*}, \quad w_{ij} = \frac{1}{m d_{ij}^*}.$$

Various types of minimization of the stress function are possible. One of simple ways is a gradient descent method. Then the coordinates $y_k^i, i =$

$1, \ldots, m$, $k = 1, 2$, of the two-dimensional vectors $Y^i \in R^2$ can be computed by the iteration formula: $y_k^i(t+1) = y_k^i(t) - \eta \frac{\partial E_{MDS}(t)}{\partial y_k^i(t)}$. Here t denotes the iteration number, $\eta$is the parameter, influenced optimisation step. However, other ways of optimisation are also possible: SMACOF algorithm (Scaling by majorization of a complicated function) [BG97], conjugate gradient, quasi-Newton method, simulated annealing, combination of genetic algorithm and quasi-Newton's descent algorithm [MZ93, Pod03]. Minimization of Stress functions are complex problems for multimodal criteria [ZP03].

In this chapter, we use the SMACOF algorithm for stress function minimization. This algorithm for MDS can be summarized as follows (see details in [BG97]:

1. Set the initial values of Y, set$t = 0$.
2. Compute the value of the stress function $E_{MDS}(Y(t))$; here the two-dimensional vectors Y are set in $t = 0$.
3. Increase the iteration counter t by one.
4. Compute the Guttman transform $Y(t)$ by (2).
5. Compute $E_{MDS}(Y(t))$.
6. If $E_{MDS}(Y(t-1)) - E_{MDS}(Y(t)) < \varepsilon$ or $t =$ maximum number of iterations, then stop ($\varepsilon$is as small positive constant), else go to Step 3.

Formula (2) is called the Guttman transform.

$$Y(t+1) = m^{-1} B(Y(t)) Y(t), \tag{2}$$

where $B(Y(t))$ has the elements

$$b_{ij} = \begin{cases} -\dfrac{d_{ij}^*}{d_{ij}}, & \text{for } i \neq j \text{ and } d_{ij} \neq 0, \\ 0, & \text{for } i \neq j \text{ and } d_{ij} = 0, \end{cases}$$

$$b_{ii} = -\sum_{j=1, j\neq i}^{m} b_{ij}. \tag{3}$$

Sammon's mapping [Sam69] is one of the MDS group methods. The cost function of Sammon's mapping is the following distortion of projection:

$$E_S = \frac{1}{\sum_{i,j=1,i<j}^{m} d_{ij}^*} \sum_{i,j=1,i<j}^{m} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*}. \tag{4}$$

The function E_S is coincident with function $E_{MDS}(1)$, when $w_{ij} = \frac{1}{d_{ij}^* \sum_{k,l=1,k<l}^{m} d_{kl}^*}$.

Because of the normalization by the distance in the original space, the preservation of small distances will be emphasized [Kas97].

In the steepest-descent minimization, the coordinates $y_k^i, i = 1, \ldots, m$, $k = 1, 2$, of the two-dimensional vectors $Y^i \in R^2$ are computed by the iteration formula:

$$y_k^i(t+1) = y_k^i(t) - \eta \frac{\frac{\partial E_S(t)}{\partial y_k^i(t)}}{\left|\frac{\partial^2 E_S(t)}{\partial (y_k^i)^2(t)}\right|}. \tag{5}$$

Here t denotes the iteration order number; $\eta$is a parameter, which influences the optimisation step. Various realizations of the algorithm are possible. For example, it is advisable to apply the coordinate descent method in Sammon's mapping. In solving linear equation systems and in optimisation, it is called Seidel's method. The coordinates of the two dimensional vectors Y^i are recalculated, taking into consideration not only the coordinates, obtained in the previous iteration, as in classical Sammon's algorithm, but also the new coordinates, obtained in the current iteration, i.e., the coordinates $y_k^j(t+1)$, if $j = 1, \ldots, i-1$, and $y_k^j(t)$, if $j = i+1, \ldots, m$ [BDKM05, DBKM05]. This realization of Sammon's algorithm was used in the experiments in this chapter. Another way of Sammon's error minimization is based on an artificial neuron network (SAMMAN).

SAMANN algorithm. J. Mao and K. Jain [MJ95] have suggested a neural network implementation of Sammon's mapping. A specific backpropagation-like learning rule has been developed to allow a normal feedforward artificial neural network to learn Sammon's mapping in an unsupervised way, called SAMANN. The SAMANN network offers the generalization ability of projecting new data, which is missing in the original Sammon's projection algorithm. SAMANN is a feedforward neural network where the number of input units is set to be the feature space dimension n, and the number of output units is specified as the extracted feature space dimension d. They have derived a weight updating rule for the multilayer perceptron neural network that minimizes Sammon's stress, based on the gradient descent method. The general updating rule for all the hidden layers ($l = 1, \ldots, L-1$) and for the output layer ($l = L$) is

$$\Delta\omega_{jk}^{(l)} = -\eta \frac{\partial E_{\mu\nu}}{\partial \omega_{jk}^{(l)}} = -\eta\big(\Delta_{jk}^{(l)}(\mu) y_j^{l-1}(\mu) - \Delta_{jk}^{(l)}(\nu) y_j^{l-1}(\nu)\big),$$

where ω_{jk} is the weight between the unit j in the layer $l-1$ and the unit k in the layer l, η is the learning rate, $y_j^{(l)}$ is the output of the jth unit in the layer l, and μ and ν are two patterns. The $\Delta_{jk}^{(l)}$ are the errors accumulated in each layer and backpropagated to a preceding layer, similarly to the standard backpropagation. The sigmoid activation function whose range is $(0.0, 1.0)$ is used for each unit. However, in the neural network implementation of Sammon's mapping the errors in the output layer are functions of the interpattern distances. In each learning step, the artificial neural network is shown by two points. The outputs of each neuron are stored for both points. The distance between the neural network output vectors can be calculated and an error measure can be defined in terms of this distance and the distance between the points in the input space. From this error measure a weight update rule can be derived. Since no output examples are necessary, this is an unsupervised algorithm.

As an alternative to SAMANN's unsupervised learning rule, one could also train a standard feedforward artificial neural network, using supervised backpropagation on a previously calculated Sammon's mapping. Although it requires much more computation, as it involves two learning phases (one for Sammon's mapping, one for the neural network), it should perform at least as well as SAMANN [RD97].

When projecting data, it is of great importance to achieve good results in a short time interval. In the consideration of the SAMANN network, it has been observed that the projection error depends on different parameters. The latest investigations have revealed that, in order to achieve good results, one needs to correctly select the learning rate η. J. Mao and K. Jain used $\eta = 0.7$ in their study. In that case, however, the network training is very slow. One of the possible reasons is that, in the case of the SAMANN network, the interval $(0, 1)$ is not the best one. Thus, it is reasonable to look for the optimal value of the learning parameter that may not necessarily be within the interval $(0, 1)$ (see [MD06]).

The self-organizing map (SOM) [Koh01] is a class of neural networks that are trained in an unsupervised manner, using competitive learning. Let $X^1, X^2, \ldots, X^m \in R^n$ be a set of n-dimensional vectors for mapping, m is the number of the vectors. The rectangular SOM is a two-dimensional array of neurons $\{m_{ij}, i = 1, \ldots, k_x, j = 1, \ldots, k_y\}$. Here k_x is the number of rows, and k_y is the number of columns. Each neuron is an n-dimensional vector$m_{ij} = (m_1^{ij}, m_2^{ij}, \ldots, m_n^{ij}) \in R^n$. The learning starts from the vectors m_{ij} initialised at random or by another rule. At each learning step, the input vector X is passed to the neural network. The Euclidean distance from this input vector to each vector m_{ij} is calculated and the vector (neuron) m_c with the minimal Euclidean distance to X is designated as a winner. The components of m_{ij} are adapted according to the rule: $m_{ij} \leftarrow m_{ij} + h_{ij}^c(X - m_{ij})$. In our case, $h_{ij}^c = \beta/(\beta\eta_{ij}^c + 1)$, $\beta = \max((e + 1 - \hat{e})/e, 0.01)$; η_{ij}^c is the neighbourhood order between the neurons m_{ij} and m_c; e is the number of training epochs; $\hat{e}$ is the order number of the current epoch. A training epoch consists of m steps: the input vectors from X^1 to X^m are passed to the neural network. The vector m_{ij} is recalculated if $\eta_{ij}^c \leq \max[\beta \max(k_x, k_y), 1]$. After a large number of training steps, the network has been reorganized and n-dimensional input vectors $X^1, \ldots, X^m$ have been mapped – each input vector is related to the nearest neuron, i.e., the vectors are distributed among the elements of the map during training. The neurons related with the input vectors are called neurons-winners. The training quality of the neural network can be evaluated by the following formula:

$$E_{SOM} = \frac{1}{m} \sum_{i=1}^{m} ||X^i - m_c^i||. \tag{6}$$

Here m is the number of visualised vectors X^i, $i = 1, \ldots, m$, m_c^i is the codebook vector-winner, corresponding to the vectorX^i. In the case of the rectangular topology, we can draw a simple table with cells corresponding to the neurons. However, the table and its properties do not answer the question, how much the vectors of the neighboring cells are close in the n-dimensional space. The answer may be found, by using additional visualization of SOM, for example, graphic display, called the U-matrix (Unified distance matrix) [Koh01, DK02], and component planes [Koh02].

Sometimes all the ways, discussed above, of the SOM visualization are not sufficient to comprehend the analysis of the data. Therefore, it is necessary to use other ways of visual presentation or their combinations, for example, combinations of the SOM and some nonlinear projection mappings (Sammon's algorithm or MDS).

The combination and integrated use of data analysis methods of different nature are under a rapid development. The combination of different methods can be applied to make a data analysis, while minimizing the shortcomings of individual methods.

Combination of the SOM and Sammon's mapping (SOM_Sammon). Sammon's mapping is quite a common tool for the multidimensional data visualization, though it has its own shortcomings: the obtained projection error depends on the initial values of two-dimensional points, and the parameterη, which influences the optimisation step; Sammon's algorithm complexity is $O(m^2)$ for a fixed number of iterations (m is the total number of analysed data). The SOM is able not only to visualize the multidimensional data, but also to cluster them. However, the SOM does not answer the question, how much the vectors of the neighbouring cells are close in the n-dimensional space. Thus, when both these methods are joined, it is possible to expect some alleviation or even elimination of the above-mentioned shortcomings: data clustering would be performed; the SOM would be visualised informatively.

There are several ways of combining the SOM and Sammon's mapping:

- *consecutive combination*, where all input vectors $X^1, X^2, \ldots, X^m$ first are processed, using the SOM, afterwards all the neurons or only the vectors-winners are displayed using Sammon's mapping; such a combination has been used in various applications [Kas97, Dze01, Kon00].
- *integrated combination*, proposed and analysed in [Kur05, DK02], where multidimensional data are projected onto a plane by using Sammon's mapping, taking into account the learning flow of the SOM.

The integrated combination is superior to the consecutive one because of a more faithful projection. In the paper [BDKM05], we examined the combination of the SOM not only with Sammon's mapping, but also with the MDS.

3.3 Quantitative Criteria of Mapping

The problem of objective comparison of the mapping results arises when the multidimensional data are visualized using various methods that optimise different criteria of the mapping quality. It is necessary to select a set of universal criteria that describe the projection quality and may be common for different methods. Three criteria of this kind are presented below.

Metric topology preserving (MTP). Given spaces $R^n(\forall X \in R^n)$ and $R^d(\forall Y \in R^d)$, where $d < n$, a map M: $X \to Y$ is called *topology preserving*, if

$$\forall i, j, k, l \quad d^*_{ij} < d^*_{kl} \Rightarrow d_{ij} \leq d_{kl}.$$

Here d^*_{ij} is the distance between points X^i and X^j, $(X^i, X^j \in R^n)$; d_{ij} is the distance between points Y^i and Y^j, $(Y^i, Y^j \in R^d)$, $Y = M(X)$. The larger value of MTP is better. In the experiments below we scaled the value of MTP its the largest possible value became equal to 1.

Assume that, for each space R^n and R^d, there is a similarity function, which, for any given pair of points in the space, specifies by a non-negative scalar value how similar (or dissimilar) they are. Denote for R^n function by F and for R^d by G(in some cases, F and G can be Euclidean distances). Then we define the cost functional C as follows:

$$C = \sum_{i=1}^{m} \sum_{j<i} F(i,j) G\big(M(i), M(j)\big),$$

where i and j denote the points in R^n and $M(i)$ and $M(j)$ are their respective images in R^d.

Minimal wiring (MW). In minimal wiring, a good mapping is defined to be the one that maps points, that are nearest in space R^n, as close in space R^d, where the closeness in R^n is measured by, for instance, the Euclidean distance [GS96]. Minimal wiring is equivalent to the C measure for

$$F(i,j) = \begin{cases} 1, & i,j \text{ neighbouring}, \\ 0, & \text{otherwise}, \end{cases}$$

$$G\big(M(i), M(j)\big) = \big\|M(i) - M(j)\big\|^2.$$

The lower values of the criterion MW correspond to a better map.

Spearmen coefficient. Since preservation of distance is an important characteristic of the MTP transformation, we relabel the distance in $D^* = [d^*_{ij}]$ as $[d^*_k]$ where $d^*_k = d^*_{ij}$, $k = (i-1)((2m-i)/2) + (j-i)$. Relabel the distances in $D = [d_{ij}]$ in the same way. Let the rank of the kth elements in D and D^* be $r(k)$ and $r^*(k)$, respectively, and let r and r^* be the corresponding vectors of ranks [BP95].

Deviation from MTP can be measured by a rank correlation coefficient. Specifically, Spearman's ρ_{Sp} seems applicable.

$$\rho_{Sp}(r^*, r) = 1 - \frac{6\sum_{k=1}^{T}(r^*(k) - r(k))^2}{T^3 - T},$$

where $T = m(m-1)/2$. As usual, $-1 \leq \rho_{Sp} \leq 1$. The best value of the Spearmen coefficient is equal to one.

4 Experimental Results

4.1 Analysis of the Physiological Fractal Dimension Data

A physiological fractal dimension data set has been analysed. The vectors, consisting of parameters of the information dimension, are analysed in this investigation. The dimension of these vectors is equal to 6.

At first the analysed data were classified by Naïve Baeys, classification tree, and Support vector machine classifiers, using the data analysis system "Orange" [DZL04]. Given a classification problem, no single classification technique yields the best results. Therefore, it is necessary to use several classification methods. Some ways of choosing the best classifiers are possible: one technique is to choose a classifier that has the best accuracy in a data set sample; another way is simple voting: assign the item to the class to which the majority of the classifiers has been assigned [Dun03]. Second way is acceptable as well, because the classifier with the best accuracy can be so adjusted to the training data that it is not able to classify the new data well.

We show that these physiological fractal dimension data do not consist of three classes exactly, though the parameters have been measured for three groups of patients. First the classifiers have been trained using the data of all three groups. The accuracy of classifiers is presented in Table 1. Here the accuracy of classifiers is obtained using the cross validation strategy, the number of folds being equal to 5.

Table 1 shows that the accuracy of classifiers is insufficient. Therefore we decided to eliminate the data of Group 2 (healthy persons) from the training set. The classifiers have been trained using only the data of Groups 1 and 3. The obtained results are presented in Table 2.

When the data of Groups 1 and 3 are classified, the accuracy of classification increases to 88-89%. Thus we can draw a conclusion, that Group 2 (healthy persons) is not a class. An idea arises to observe in what way the objects of Group 2 are assigned to Class 1 or 3, i.e., which healthy persons are similar to sportsmen, which ones are similar to ischemic heart diseased patients in terms of the measured parameters.

Number of the correctly classified data of Group 1 and 3 is presented in Table 3. The results of kNN classifier are not presented in Table 3, because it

Table 1. Accuracy of classifiers (three data groups (1, 2, and 3) have been classified)

Classifiers	Accuracy, %
Naïve Bayes	66
Classification tree	61
KNN	58
SVM	64

Table 2. Accuracy of classifiers (two data groups (1 and 3) have been classified)

Classifiers	Accuracy, %
Naïve Bayes	89.19
Classification tree	88.25
kNN	87.83
SVM	89.63

Table 3. Accuracy of classifiers for the training set of the physiological fractal dimension data

Classifiers	Number of the correctly classified data		
	(1) ischemics (total 61)	(3) sportsmen (total 161)	both groups (total 222)
Naïve Bayes	55 (90%)	146 (91%)	201 (90.5%)
Classification tree	55 (90%)	160 (99%)	215 (97%)
SVM	52 (85%)	155 (96%)	207 (93%)
Majority	55 (90%)	155 (96%)	210 (95%)

is an unsupervised classifier, therefore its accuracy is 100% for the training set. The number of correct classification by the majority of classifiers is presented in the last row of Table 3.

Such investigation have been performed:

- Classifiers are formed using the training set that consists of the data of Group 1 and 3 (the accuracy of classifiers is presented in Table 2).
- Using the created classifiers, we define whether the data of Group 1 and 3 were correctly classified (the accuracy of classifiers is presented in Table 3).
- Using the created classifiers, we determine to which class the objects of Group 2 are assigned.

The results, presented in Table 3, show that the accuracy of classification is insufficient. Since the medical data are investigated, the results of classification of these data must be especially precise.

An alternative of classical classification is visual analysis of multidimensional data: projections of data into a plane are calculated, and visual groups of data are observed. In our investigation, some projections (dimension reduction) methods have been used. The main problem of such a visualization (projection) is assessment of the quality of mapping. One of the simple ways is to estimate the error, obtained by minimizing criteria of the method itself, and to select map, corresponding to the lowest minimization error, obtained by the algorithms. When 6-dimensional data are projected to a plane, some projections are obtained:

Table 4. Values of quantitative criteria of mappings of physiological fractal dimension data

	SAMANN	Sammon			MDS		
Initialisation type	Rand	by PCA	Rand		by PCA	Rand	
Error of algorithm	0.10003	0.034063	0.033946	0.034062	0.010816	0.009263	0.00964
MW	**19.216196**	25.468488	25.888919	25.636505	25.693917	24.869168	25.467240
MTP	0.888781	0.906650	0.906942	0.906685	0.902036	**0.907421**	0.906163
SC	0.924129	0.943163	0.943514	0.943163	0.934586	**0.943804**	0.940781

- Seidel type realization of the Sammon's mapping and the multidimensional scaling (SMACOF algorithm) have been used, when the initial coordinates of two-dimensional vectors are equal to two principal components of the multidimensional data.
- After 100 experiments by the Sammon and MDS methods, the initial components of two-dimensional vectors being random number in interval (0, 1), two maps for each methods have been selected, the projection errors of which were lowest.
- Projections, obtained by the SAMANN algorithm.

The problem of objective comparison of the mapping results arises when the multidimensional data are visualised using various methods that optimise different criteria of the mapping quality. The way out is to estimate mapping using criteria, that do not depend on the method. In this investigation, three criteria are used: minimal wiring (MW), metric topology preserving (MTP), spearmen coefficient (SC). The joint use of quantitative criteria of mapping allows us to choose the best visualization result from some possible projections. We estimate seven mappings by these criteria. The values of all the criteria are presented in Table 4. When choosing the best mapping, such a strategy is used: select such a mapping, that the values of majority of estimated criteria are best.

In Table 4, the bold font indicates a better result. Table 4 shows that the quality of maps, obtained by MDS algorithm ($E_{MDS} = 0.009263$), is better as compared with those maps, obtained by other methods, in many cases.

Figure 2 illustrates the projection of the data of Group 1 and 3, obtained by MDS (SMACOF algorithm). Some groups are indicated:

- ischemic heart diseased patients (assigned to ischemics by medics and by most classifiers);
- sportsmen, whom the majority of classifiers assigned to ischemics;
- sportsmen (assigned to sportsmen by medics and by most classifiers)
- ischemics, whom the majority of classifiers assigned to sportsmen.

As seen in Fig. 2, the majority of points corresponding to ischemics are on the one side of the picture, while the points, corresponding to sportsmen are on the opposite side. However, these groups partly overlap. Therefore, it is reasonable to isolate the area of those intermix points of groups. One of the

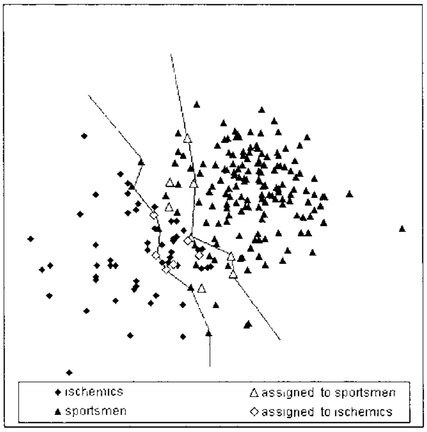


Fig. 2. Projection of the physiological fractal dimension data (Groups 1 and 3)

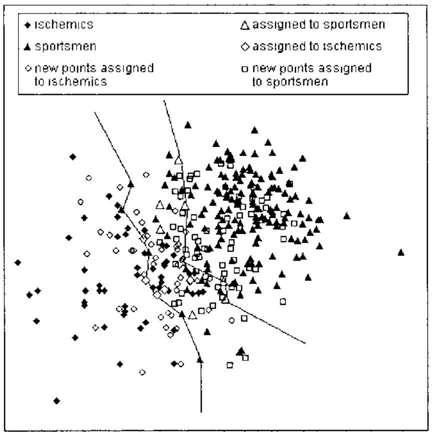


Fig. 3. Projection of the physiological fractal dimension data (all 3 groups)

simplest ways is to connect the closest points, classified incorrectly or assigned to the opposite class by a broken line.

When calculating the projections, the points of all three groups have been used, however, in order to more exactly define the area, in which the points of the investigated classes are intermixed, only the points of Groups 1 and 2 are demonstrated in Fig. 2. It is difficult to tell by the visual image which class the points between the broken lines should be assigned to and which of them correspond to a particular patient under investigation. These patients should be studied and observed more in detail.

Figure 3 illustrates the points of all three groups. When visualizing the data, the points of Group 2 are denoted with reference to the class they have been assigned to by most of the classifiers, i.e., it is indicated:

- healthy persons, assigned to Class 1 by the classifiers (circles);
- healthy persons, assigned to Class 3 by the classifiers (squares).

From the results of classification and visual analysis we can draw the following conclusions:

1. Persons not going in for sports assigned by the classifiers to the persons going in for sports and the points corresponding to them visually are arranged in the area of points corresponding to the persons who go in for sports (on the right in Fig. 3), are healthy and can go in for sports, because their measured parameters do not differ at all from the persons going in for sports.
2. Persons not going in for sports assigned by the classifiers to the persons sick with ischemic heart disease and the points corresponding to them visually are arranged in the area of points corresponding to ischemic patients (on the left in Fig. 3) may possibly have some health problems and are

not allowed to practice exercises without a thorough examination because their measured parameters do not differ at all from ischemics patients.
3. It is also worthwhile observing and examining the sportsmen assigned by classifiers to ischemic patients, and the visual analysis also shows that their points are arranged close to the points corresponding to ischemic patients, because there is a possibility for them to have some health disturbances.

In the visualization of multidimensional data there may be used other strategies, e.g., SAMANN algorithm, the advantage of which against the other algorithms of the MDS group is that it is not necessary to look for projections of all the points again then mapping the points of a new set.

In the analysis of the visualization of multidimensional data, a particular case of the SAMANN network was considered: a feedforward artificial neural network with one hidden layer and two outputs. In each case, the same number of neurons (20) of the hidden layer was taken and the set of initial weights was fixed in advance. To visualize the initial dataset, the following parameters were employed: the number of iterations 100000, the training parameter$\eta = 1$. One iteration means showing all pairs of samples to the neural network once. The SAMANN network is trained by two groups (ischemics and sportsmen), using the standard backpropagation algorithm with a learning rate of 1.0 and the momentum value of 0.3 for 100 000 iterations. A new set of SAMANN network outputs has been found for two groups (ischemics and sportsmen). The set of SAMANN network weights ω has been calculated. Now it is possible to decide where to place the third group data (healthy persons) in the final 2-dimensional configuration created by SAMANN. The third group shown to the network is mapped to the plane very fast and quite exactly without any additional calculations.

The data visualization results are presented in Fig. 4 (two groups are shown: ischemics and sportsmen) and Fig. 5 (three groups: ischemics, sportsmen, and healthy persons). The points assigned by classifiers to different classes then those assigned by medics are also isolated. Fig. 5 shows the points classified in Group 2 (healthy persons).

The results obtained both by the MDS and SAMANN methods are rather similar visually. However, the projection obtained by the MDS method is a little bit more exact (see Table 4), but it there is a necessity to map the points of the new data set by the SAMANN algorithm, there is no need to calculate projections of all the points a new contrary to the case using the MDS methods. In this case, the mutual arrangement of the points of the first two groups does not change as well after placing the points of the third group on the plane.

4.2 Analysis of the Ophthalmologic Data Set

First at all the ophthalmologic data were classified by some classifiers. The training data set consists of vectors, corresponding to healthy eyes and eyes, damaged by glaucoma. The dimensionality of these vectors is equal to 27.

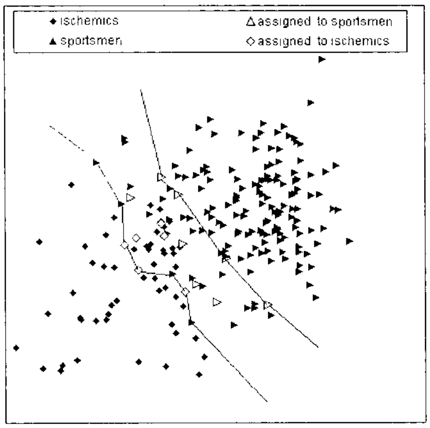


Fig. 4. Projection of the training physiological fractal dimension data (Groups 1 and 3), obtained by SAMANN algorithm

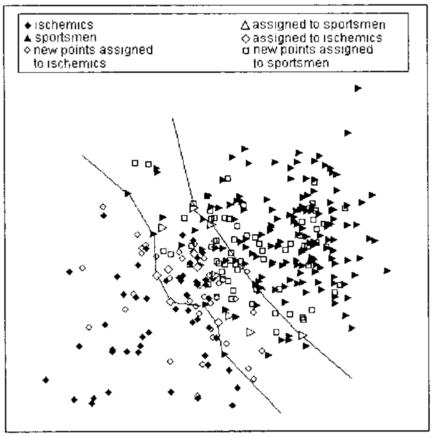


Fig. 5. Projection of the training physiological fractal dimension data (all 3 groups), obtained by SAMANN algorithm

Table 5. Accuracy of classifiers

Classifiers	**Accuracy, %**
Naïve Bayes	64
Classification tree	56
KNN	57.7
SVM	58

Table 6. Accuracy of classifiers for training set of ophthalmologic data

	Number of the correctly classified data		
Classifiers	Norm (total 18)	Glaucoma (total 24)	Total 42
Naïve Bayes	17 (94%)	17 (71%)	34 (81%)
Classification tree	17 (94%)	23 (96%)	40 (95%)
SVM	16 (89%)	19 (80%)	35 (83%)
Majority	17 (94%)	19 (80%)	36 (86%)

The results of classification are presented in Table 5. Number of the correctly classified data of the ophthalmologic data is presented in Table 6.

The obtained results show that the accuracy of classification is insufficient. It is necessary to find out, how the points, corresponding to the ophthalmologic data, are located on the plane. With a view to select the best projection we have used the same strategy as that when analysing physiological data of fractal dimensions. The values of the evaluated criteria are presented in Table 7. It shows that the best map is obtained by the MDS method. In addi-

Table 7. Values of quantitative criteria of mappings of ophthalmologic data

	Sammon			**MDS**		
Initialisation type	by PCA	Rand		by PCA	Rand	
Error of algorithm	0.002272	0.002263	0.002296	0.002296	0.0022958	0.002302
MW	26.809435	26.597401	26.756312	25.618042	25.61042	**25.58507**
MTP	0.982044	0.982100	0.982241	0.987100	0.987100	**0.987189**
SC	0.998001	0.997981	0.998025	0.998867	0.998867	**0.99888**

tion, note that, according to the estimation strategy, the best map not always corresponds to the lowest value of the function optimised by the algorithm.

In Fig. 6, a part of points corresponding to glaucoma intermix among the points corresponding to the norm. One of the reasons may be the fact that when projecting 27-dimensional vectors into a two-dimensional space, there appear noticeable distortions. As mentioned above when analysing the physiological data, frequently the classification results alone just like the visualization alone are not sufficient, therefore it is reasonable to combine both these results in the mapping as well.

In Fig. 7, the points, which were assigned by classifiers not to the same class as that assigned by medics, are separated as well. All the three classifiers assigned one point, corresponding to the norm, to glaucoma. In the projection map, this point (dark triangle) is closer to the points, corresponding to glaucoma. One point, corresponding to glaucoma, was assigned by all the classifiers to the norm. In the picture, this point (empty rhombus) is closer to the points corresponding to the norm. Several points, corresponding to glaucoma (empty circles), present in a "cloud" of points, corresponding to the norm, were assigned to glaucoma by one classifier, and to the norm by two classifiers. There is no strict watershed in this picture between the points corresponding to the norm, and that corresponding to glaucoma, however, it is possible to discern certain groups. Based on the classification and visualization results, we have a set of shady points, and the patients corresponding to them are recommended to be repeatedly examined for specifying their diagnosis.

Another visualization strategy applied in the research is based on self-organizing neural networks (SOM), as the SOM is combined with one of the MDS group methods – the Sammon's algorithm (SOM_Sammon). Figure 8 presents the result of the integrated combination SOM_Sammon. Since the SOM also plays a clusterisation role, some points of the analysed set can get to one cell of the SOM (Fig 9).

Therefore it is more informative to isolate by colours not the points themselves, but the vector number corresponding to them (1–18 points correspond to the norm, (brighter points), 19–42 to glaucoma (dark points)). The point, assigned by classifiers note to the same classes as that by medics, are marked in

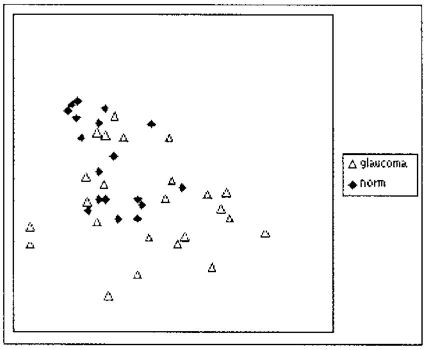


Fig. 6. Projection of the ophthalmologic data

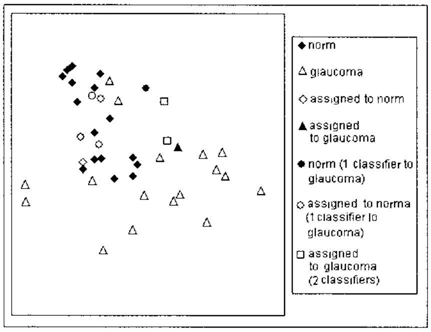


Fig. 7. Projection of the ophthalmologic data (taking into account the classification results)

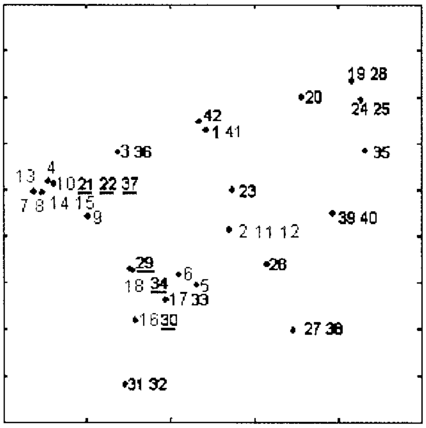


Fig. 8. Projection, obtained by integrated SOM_Sammon mapping

19 26	24 25	35	39 40		27 38
20				28	
42	1 41	23	2 11 12		5
3 36				6	17 33
4	10 21 22 37	9	29	16 30	
7 8 14 15	13		18 34		31 32

Fig. 9. SOM table [6×6]

the same way: the numbers of points corresponding to the norm but assigned to glaucoma by classifiers, are marked in bold, and the underline numbers correspond to glaucoma, but were assigned to the norm by classifiers.

Applying various visualization and classification strategies, rather similar results have been obtained, while the set of shady points are the same.

5 Conclusions

We have analysed here two sets of multidimensional medical data (physiological fractal dimension data set and ophthalmologic data set) using various data mining methods and their combinations. The topical problem of analysis of the medical data is their classification, i.e., assignment of the health state of patients to one of the known classes (for example, healthy or sick person). A particularity of the medical data classification is the fact that often the transit from the normal state to diseased one is inconspicuous, for example, healthy/glaucoma cases. Therefore, it is of utmost importance to estimate bounds of that transit. It is not enough to determine a bound by a function, that separates classes. It is proposed here the way of finding the area of the points, which correspond to the patients that should be taken into consideration - the wrong diagnosis for these patients is possible. This way integrates the results of the classification and visualization of multidimensional data. The visual analysis (arrangement of the points, corresponding to the data of the patients on the plane and visual presentation of the results of classification in the same map) enables us to make a prompt decision on the analysed data. In this chapter, a wide scale of data mining means was used. The comparative results of some classification methods (Naïve Bayes, Classification trees, kNN classifier, and Support Vector Machine classifier) are presented. To obtain more objective results, the data were visualized using some projection methods (MDS, Sammon, SAMANN). Some projections of the data are obtained. These projections were estimated using some criteria (Metric topology preserving, Minimal wiring and Spearmen coefficient) that are independent of the projection methods.

Evidently, the faithful automated diagnosis is impossible, however the results, obtained by the method, proposed in this chapter, can be of use to medics for a preliminary diagnosis: healthy, unclear or sick persons. This conclusion follows from the analysis of the physiological fractal dimension data. Meanwhile, the analysis the ophthalmologic data does not yield a preliminary diagnosis. One of the reasons is that the system of parameters is insufficient to characterise a disease (as a rule, a medic makes a decision when investigating the image of the eye fundus). However, we found the patients, whose diagnosis is the same as the results of classification, but it is at variance with the results of visualization. Namely, these are the data to be paid a particular attention: for medics to have doubts to the diagnosis, for developers of the system of parameters, maybe other parameters should be calculated, and for developers of data mining methods to further improve them.

Acknowledgement. The authors would like to thank Prof. A. Vainoras and his colleagues from Kaunas Medical University and Prof. Z. Navickas and his colleagues from Kaunas University of Technology for the physiological fractal dimension data set. The base of research of ophthalmologic data is the project "Information technologies for human health – clinical decision support (e-Health). IT Health (No. C-03013)", supported by the Lithuanian State Science and Studies Foundation.

References

[Adl03] Adlassnig, K.P. (ed): Artificial intelligence in medicine. Elsevier (2003)

[BBA$^+$05] Bernataviciene, J., Berskiene, K., Aseriskyte, D., Dzemyda, G., Vainoras, A.: Analysis of biomedical informativity of the fractal dimensions. In: IX-th International Conference "Biomedical Engineering". Kaunas University of Technology, October 27–28, 2005, 27–31 (2005)

[BDKM05] Bernataviciene, J., Dzemyda, G., Kurasova, O., Marcinkevicius, V.: Optimal decisions in combining the som with nonlinear projection methods. European Journal of Operational Research (in press) (2005)

[BFOS84] Breiman, L., Friedman, J., Olshen, R., Stone, S.: Classification and Regression Trees. Chapman and Hall, New York (1984)

[BG97] Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer, New York (1997)

[BP95] Bezdek, J.C., Pal, N.R.: Index of topological preservation for feature extraction. Pattern Recognition, **28**(3), 381–391 (1995)

[CS03] Cristianini, N., Shawe-Taylor, J.: Support vector and kernel methods. In: Berthold, M., Hand, D.J. (eds) Intelligent Data Analysis: An Introduction. Springer, 169–197 (2003)

[DBKM05] Dzemyda, G., Bernataviciene, J., Kurasova, O., Marcinkevicius, V.: Minimization of the mapping error using coordinate descent. In: WSCG'2005 Short Papers Proceedings, 169–172 (2005)

[DK02] Dzemyda, G., Kurasova, O.: Comparative analysis of the graphical result presentation in the SOM software. Informatica, **13**(3), 275–286 (2002)

[DK05] Dzemyda, G., Kurasova, O.: Heuristic approach for minimizing the projection error in the integrated mapping. European Journal of Operational Research (in press) (2005)

[Dun03] Dunham, M.H.: Data Mining Introductory and Advanced Topics. Pearson Education, Inc. (Prentice Hall) (2003)

[Dze01] Dzemyda, G.: Visualization of a set of parameters characterized by their correlation matrix. Computational Statistics and Data Analysis, **36**(10), 15–30 (2001)

[DZL04] Demsar, J., Zupan, B., Leban, G.: Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper, Faculty of Computer and Information Science, University of Ljubljana (2004) `http://www.ailab.si/orange`

[FT74] Friedman, J.H., Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput., **23**(9), 881–890 (1974)

[GHP02] Grinstein, G.G., Hoffman, P.E., Picket, R.M.: Benchmark development for the evaluation of visualization for data mining. In: Fayyad, U., Grinstein, G.G., Wierse, A. (eds) Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers, San Francisco (2002)

[GS96] Goodhill, G.J., Sejnowski, T.: Quantifying neighbourhood preservation in topographic mappings. In: Proceedings of the 3rd Joint Symposium on Neural Computation, University of California, 61–82 (1996)

[HG02] Hoffman, P.E., Grinstein, G.G.: A survey of visualizations for high-dimensional data mining. In: Fayyad, U., Grinstein, G.G., Wierse, A. (eds) Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers, San Francisco (2002)

[HS89] Hastie, T., Stuetzle, W.: Principal curves. Journal of the American Statistical Association, **84**, 502–516 (1989)

[JLPB02] Jegelevicius, D., Lukosevicius, A., Paunksnis, A., Barzdziukas, V.: Application of data mining technique for diagnosis of posterior uveal melanoma. Informatica, **13**(4), 455–464 (2002)

[Kas97] Kaski, S.: Data Exploration Using Self-Organizing Maps. PhD thesis, Department of Computer Science and Engineering, Helsinki University of Technology (1997) `http://www.cis.hut.fi/~sami/thesis/`

[Koh01] Kohonen, T.: Self-Organizing Maps. 3rd ed. Springer Series in Information Sciences, vol. 30, Springer (2001)

[Koh02] Kohonen, T.: Overture. In: Seifert, U., Jain, L.C. (eds) Self-Organizing Neural Networks. Springer, New York, 1–10 (2002)

[Kon00] Konig, A.: Interactive visualization and analysis of hierarchical neural projections for data mining. IEEE Trans. Neural Networks, **11**(3) (2000)

[Kur05] Kurasova, O.: Visual Analysis of Multidimensional Data Using the Self-Organizing Maps. Doctoral dissertation, Vilnius (2005)

[KW03] Keim, D.A., Ward, M.: Visualization. In: Berthold, M., Hand, D.J. (eds) Intelligent Data Analysis: an Introduction. Springer, 403–427 (2003)

[LKZ97] Lavrac, N., Keravnou, E., Zupan, B. (eds): Intelligent Data Analysis in Medicine and Pharmacology. Springer (1997)

[MD06] Medvedev, V., Dzemyda, G.: Retraining of the SAMANN network. In: SOFSEM 2006 Proceedings. MatFyz Press (to appear) (2006)

[MJ95] Mao, J., Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks, **6**, 296–317 (1995)

[MST94] Michie, D., Spiegelhalter, D.J., Taylor, C.C. (eds): Machine Learning, Neural and Statistical Classification. Ellis Horwood (1994)

[MZ93] Mathar, R., Zilinskas, A.: On global optimization in two-dimensional scaling. Acta Aplicandae Mathematicae, **33**, 109–118 (1993)

[OM94] Ohno-Machadom, L., Musen, M.A.: Hierarchical neural networks for partial diagnosis in medicine. In: World Congress on Neural Networks. San Diego, CA, I-291 to I-296 (1994)

[Pod03] Podlipskyte, A.: Visualization of Multidimensional Data and its Application to Biomedical Data Analysis. Doctoral dissertation (2003)

[RD97] De Ridder, D., Duin, R.P.W.: Sammon's mapping using neural networks: A comparison. Pattern Recognition Letters, **18**, 1307–1316 (1997)

[RS03] Ramoni, M., Sebastiani, P.: Bayesian methods. In: Berthold, M., Hand, D.J. (eds) Intelligent Data Analysis: An Introduction. Springer, 131–168 (2003)

[Sam69] Sammon, J.W.: A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, **18**, 401–409 (1969)

[SIL01] Sierra, B., Inza, I., Larrañaga, P.: On applying supervised classification techniques in medicine. Medical Data Analysis. Lecture Notes on Computer Sciences, Vol. 2199, Springer, 14-19 (2001)

[Tay03] Taylor, P.: Statistical methods. In: Berthold, M., Hand, D.J. (eds) Intelligent Data Analysis: An Introduction. Springer, 69–129 (2003)

[ZP03] Zilinskas, A., Podlipskyte, A.: On multimodality of the SSTRESS criterion for metric multidimensional scaling. Informatica, **14**(1), 121–130 (2003)

On Global Minimization in Mathematical Modelling of Engineering Applications

Raimondas Čiegis

Vilnius Gediminas Technical University, Saulėtekio av. 11, LT-10223 Vilnius, Lithuania rc@fm.vtu.lt

1 Introduction

Many problems in engineering, physics, economic and other subjects may be formulated as optimization problems, where the minimum value of an objective function should be found. Mathematically the problem is formulated as follows

$$f^* = \min_{X \in D} f(X), \tag{1}$$

where $f(X)$ is an objective function, X are decision variables, and D is a search space. Besides of the minimum f^*, one or all minimizers $X^* : f(X^*) = f^*$ should be found.

In this chapter we describe two applications where global minimization techniques are the essential part of the mathematical modelling cycle.

The first problem deals with modelling a moisture motion in wood products and, more generally, the wood drying process. Development of accurate mathematical models requires to use quite complicated models, which depend on a number of free parameters. These parameters should be selected by using experimental data. In general such problem is ill-posed, i.e. the set F of possible solutions is non-compact and there is no continuous dependence of the solution on data of the problem. Even more, the right-hand side of the equation is known only approximately due to various errors and it can not belong to the set AF, thus the given problem does not have a solution with such a right-hand side function. Tichonov proposed a special regularization principle, which gives a rule how to select a compact set defining an approximate solution. It is based on minimization of special functionals, thus global minimization algorithms are important part of such a methodology [TA86].

Identification of parameters for wood drying model is an example of such applied problem. It is described in Sect. 2. The presented analysis is based on our paper [CSS04].

The second examples deals with one civil engineering problem. The goal of mathematical modelling is to optimise pile placement schemes. A grillage consists of separate beams, which may be supported by piles, or may reside on other beams, or a mixed scheme may be the case. The optimal placement of grillage should possess, depending on given carrying capacities of piles, the minimum possible number of piles. This problem is reduced to the formulation of global optimization problem (1).

The main difficulty arises due to the fact that the objective function is not given analytically and the properties of this function (e.g. the Lipschitz constant) can not be obtained apriori. The value of the objective function for fixed values of its arguments is obtained by solving a complicated nonlinear PDE problem and it requires a large amount of CPU time to get this value. Thus we should use so called *black box* global minimization algorithms. Such situation is very typical for many applications in engineering and technology. Algorithms for optimal placement of grillages are described in Sect. 3, where also some results of numerical experiments are presented.

2 Identification of Parameters for Wood Drying Model

Moisture exists in the wood as a bound water within the walls of the cells, free water in liquid form and water vapour in gas form in the voids of wood. The main difficulty in modelling moisture motion in wood is the fact that more than one mechanism may contribute to the total flow and the relative contribution of different mechanisms may change as the drying process proceeds [CS02]. Modelling is also complicated because wood is an anisotropic porous medium.

However, the development of more accurate and general models becomes more and more popular over the last decades [PSO85, PM95]. In these models a wood is considered as porous medium with multiphase flow and heat transfer taking place. Equations are derived from fundamental mathematical models describing multiphase flow and heat transfer phenomena in porous media, which are based on principles of conservation [CS02]. Allowing for more accurate modelling in general, the use of such wood drying models becomes inevitable for some applications.

The main difficulty in applying the multiphase modelling approach for moisture motion in wood are the availability and reliability of constitutive relationships, which are the closure conditions of multiphase models. Thus identification of unknown parameters (or even coefficients) of the model becomes very important part of model calibration with respect to existing experimental data.

We use a moisture diffusion equation [SL97]. The main attention is paid to two key moments. Firstly, we investigate algorithms for solving the formulated optimization problem and try to answer the question what kind of the minimum points are determined in some previous publications. Secondly, we analyse the physical meaning of obtained coefficients and discuss their reliability.

2.1 Mathematical Model and Finite-Difference Schemes

We consider the nonstationary one-dimensional model in $[0, L] \times [0, T]$, which is proposed in [SL97]. It is assumed that different mechanisms of moisture movement can be approximated accurately using nonlinear diffusion coefficient:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial}{\partial x}\Big(d(C,u)\dfrac{\partial u}{\partial x} - \nu u\Big) = 0, \\ u(x,0) = u^0(x), \\ -\big(d(C,u)\dfrac{\partial u}{\partial x} - \nu u\big)|_{x=0} = \mu(u_e - u|_{x=0}), \quad \dfrac{\partial u}{\partial x}\Big|_{x=L/2} = 0. \end{cases} \tag{2}$$

Here $u(x,t)$ is the moisture content, μ is the diffusion rate on the boundaries described by the Newton law, u_e is the equilibrium moisture content in the outside region. The convection coefficient ν and the source coefficient $\mu > 0$ are assumed to be constant. The boundary condition on $x = L/2$ describes the symmetry condition.

Simpson and Liu[SL97], and Baronas *et al.* [BIS99] have used this model without convection (i.e. $\nu = 0$) and with the nonlinear diffusion coefficient:

$$d(C,u) = \begin{cases} Ae^{-5280/\theta}e^{Bu/100}, & \text{for } u \le u_{fsp}, \\ d(C, u_{fsp}), & \text{for } u > u_{fsp}, \end{cases}$$

where θ is the temperature, $C = (A, B, \mu, \nu)$ are free parameters. They solved this boundary value problem using the explicit finite-difference scheme

$$\begin{cases} \overline{\partial}_t U - \delta(d(C,\check{U})\delta\check{U}) = 0, \\ U^0 = u^0\,, \\ -d(C,\check{U})\delta\check{U}_0 = \mu(u_e - \check{U}_0), \quad U_{n-1} = U_{n+1}\,. \end{cases} \tag{3}$$

Here we use standard notation of the finite-difference method

$$U_i^n = U = U(t^n, x_i), \quad \check{U} = U^{n-1}, \quad \overline{\partial}_t U = \frac{U - \check{U}}{\tau}, \quad \delta U = \frac{U_{i+1/2} - U_{i-1/2}}{h}.$$

The well-known stability condition $\alpha = 2d\tau/h^2 < 1$ of the explicit Euler scheme can be very restrictive, since integration in time must be done with a very small time step.

We approximate the differential problem with the following semi-implicit scheme

$$\begin{cases} \overline{\partial}_t U - \delta\big(d(C,\check{U})\delta U - \nu U\big) = 0, \\ U^0 = u^0\,, \\ -d(C,\check{U})\delta U - \nu U|_{i=0} = \mu(u_e - U_0)\,, \quad U_{n-1} = U_{n+1}\,, \end{cases} \tag{4}$$

which is unconditionally stable.

2.2 Identification of Parameters

A parameter identification problem is given by the task to find parameters $C = (A, B, \mu, \nu)$ out of measured data. Experimental data is taken from [SL97].

We use the least squares technique and minimize three different functionals, which measure the absolute, relative and weighted errors, respectively:

$$F_d = \sum_{j=1}^{K} \big(< U >_j - u_{exp}(t_j) \big)^2, \quad F_r = \sum_{j=1}^{K} \Big(\frac{< U >_j}{u_{exp}(t_j)} - 1 \Big)^2,$$

$$F_i = \sum_{j=1}^{K} \big(< U >_j - u_{exp}(t_j) \big)^2 \frac{t_{j+1} - t_{j-1}}{2},$$

where $u_{exp}(t_j)$ are experimental moisture content values at time moments t_j, $j = 1, \dots, K$ and $< U >_j = \sum_{i=1}^{n} U_i^j h_{i+1/2}$ is the average value of U.

When the parameter identification algorithm is used for simple mathematical models, which are not obtained from fundamental heat and mass transfer equations, we do not have good initial approximations of parameters. In such situations it is important to use global minimization algorithms.

A simple two-step global minimization algorithm is applied for the parameter identification.

1. A set of initial approximations is generated randomly in the specified *apriori* region of parameters.
2. The nonlinear Simplex procedure is used to find local minimum points starting the search from each point defined in the previous step. The global minimum point is defined from the set of local minimizers.

Our hypothesis is that the parameters which are obtained in [SL97, BIS99] are not global (and sometimes even not local) minimum points. We have used these values of parameters as additional initial approximations for the Simplex procedure.

The results of computations are presented in Table 1. In the case of linear diffusion ($B = 0$) the Simplex method founds two local minimums for the functional F_d:

$$(A, \mu, \nu, F_d) = (9.624,\ 0.6104e-5,\ 8.063e-3,\ 902.08),$$
$$(A, \mu, \nu, F_d) = (1.00,\ 6.08e-5,\ -5.12e-6,\ 5.749).$$

Different functionals F_d, F_i, F_r lead to different optimal values of parameters C_j (see Table 2) and such sensitivity of these parameters also shows that models based on the diffusion equation have only a limited area of applications.

It follows from the results presented in Table 1 that the increase in the number of parameters always decreases the value of the minimized functional, but this fact can not be used as a proof that convection process is really important in the wood drying under the given conditions. Since we are using the

Table 1. Values of parameters which minimize F_d

	$v = 1.5$		$v = 5.1$	
A	82.62	1.001	205.05	30.87
B	12.39	25.28	9.78	14.95
μ	0.65	0.607	1.09	1.095
ν	0.	-5.12e-6	0.	-4.468e-6
F_d	10.17	5.75	2.808	1.76

Table 2. Optimization of different functionals $F_j, j = d, i, r$

	C_d	C_i	C_r
F_d	10.17	14.984	10.683
F_i	428532	192713	283854
F_r	0.1718	0.2001	0.1601

averaged values of moisture content, many black box models can be proposed to mimic the experimental data (including simple ODE models). A general rule should be to take a minimal number of parameters, which are sufficient to approximate the given experimental measurements.

Table 3 gives the values of parameters A, B, μ ($\nu = 0$) which are found in [SL97, BIS99]. We see that all parameters depend strongly on the air velocity v and therefore the model can not be used to predict the drying process results for the other values of v.

Table 3. Parameters for 1D[SL97] and 2D[BIS99] models

	$v = 1.5$		$v = 5.1$	
	1D	2D	1D	2D
A	1290	1290	1480	1390
B	2.32	2.41	2.48	2.55
μ	9.27e-5	4.83e-5	1.51e-4	7.86e-5

Table 4. F_d and $\alpha = 2d\tau/h^2$ values for various FDS

scheme	τ	F_d	α
(3)	30	30.07	0.075–0.1
(4)	30	30.04	0.075–0.1
(4)	360	29.87	0.9–1.2
(4)	3600	28.37	9.0–12.0

In order to compare the explicit and implicit schemes, we have investigated the dependence of F_d on the time step τ. The results are presented in Table 4 for $v = 1.5$, the parameters A, B, μ are the same as in Table 3.

3 Global Optimization of Foundation Schemes in Civil Engineering

This section describes results obtained in application of global optimization for determination of optimal pile placement schemes. A detail description of this problem is given in [BVM02, CBB04].

A grillage consists of separate beams, which may be supported by piles, or may reside on other beams, or a mixed scheme may be the case. The optimal placement of grillage should possess, depending on given carrying capacities of piles, the minimum possible number of piles.

In the case of simple geometries, simple loadings and limited number of design parameters designer can propose a good approximation of optimal pile placement schema by implementing the well known engineering tests algorithms. Then local optimization methods can be used (see [BBC02, BVM02, KLCL04], where the proposed techniques enabled the authors to find local minimum solutions). But it is evident that in more general situations the global minimum solution is desirable since it can improve significantly the quality of obtained grillage scheme.

Our main goal is to use global optimization techniques in order to find optimal placement of piles. Here we should take into account the fact that the objective function is not given analytically and the properties of this function (e.g. the Lipschitz constant) can not be obtained apriori. The value of the objective function for fixed values of its arguments is obtained by solving a nonlinear PDE problem. Thus we should use so called *black box* global minimization algorithms.

3.1 Mathematical Model

We consider the global optimization problem

$$P(\mathrm{x}^*) = \min_{\text{s.t. } \mathrm{x} \in D} P(\mathrm{x}) , \tag{5}$$

where P is the objective function, D is the feasible shape of structure, which is defined by the type of certain supports, the given number and layout of different cross-sections, and different materials in the structure, and x are design parameters. In our problem P is defined by the maximum difference between vertical reactive force at a support and allowable reaction for this support, thus allowing us to achieve different reactions at supports on different beams, or even at particular supports on the same beam:

$$P(\mathrm{x}) = \max_{1 \le i \le N_s} |R_i - f_i \, R_{allowable}| ,$$

here N_s denotes the number of supports, $R_{allowable}$ is allowable reaction, f_i are factors to this reaction and R_i are reactive forces in each support and x are nodal co-ordinates of all (or a chosen set of) supports.

Further, the minimum–maximum problem is converted to a pure minimum problem with constraints by treating P_{max} as additional unknown parameter, subject to constraints that P_{max} limits the magnitudes of P everywhere in the structure and for all load cases when design changes Δx_i are performed:

$$P(\mathrm{x}) + \sum_{i=1}^{N_s} \frac{\partial P(\mathrm{x})}{\partial x_i} \Delta x_i - P_{max} \le 0 . \tag{6}$$

All derivatives $\frac{\partial P(\mathrm{x})}{\partial x_i}$ are computed numerically by using central finite difference approximations. A beam length constraint is also included into formulation of the problem:

$$L(\mathrm{x}) + \sum_{i=1}^{N_s} \frac{\partial L(\mathrm{x})}{\partial x_i} \Delta x_i - L_0 \leq 0\,, \tag{7}$$

where L_0 is the initial length of the model.

Several possibilities exist for the choice of design parameters x_i. We use nodal coordinates of all (or a chosen set of) supports, since from the engineering point of view they are the most evident parameters.

We use a two-step hybrid global minimization algorithm. The *branch and bound* method is applied for a global search and a very efficient specialized local minimization algorithm is used to find local minimum points.

3.2 Local Optimization Algorithm

The formulated problem is strongly non-linear, thus it is solved iteratively. At each iteration the current shape is changed to a better neighbouring one. The solution requires three steps:

1. Finite element analysis.
2. Sensitivity analysis with regard to design parameters.
3. Optimal re-design with linear programming.

Two-node beam element with four degrees of freedom has been implemented in analysis, therefore the analytical sensitivity analysis with regard to nodal co-ordinates was feasible. A special *Move limit* technique (see, [Ped89]) relates the design parameters' alterations per one iteration of the linear Simplex procedure to the objective function and assures adjustment of those alterations to the extent of problem non-linearity.

Optimization of separate beams has to be included into a general iterative algorithm for optimization of the whole system of piles, because pile placement scheme of one beam influences reactions distribution in remaining beams. The following algorithm is employed:

```
Algorithm 1
Initialization:
        (1) Set stiffnesses at fictitious immovable supports of upper
        beams simulating joints with lower beams.
        (2) Set accuracy tolerance.
        (3) Set stop ← false.
while ( stop = false ) do
        (4) Optimize the upper beams using defined in the last
            iteration stiffnesses of fictitious immovable supports.
        (5) Optimize the lower beams in addition to specified
            loadings taking into account concentrated loads
            coming from the upper beams.
        (6) if ( stiffnesses of the upper and lower beams at joints
              match (with specified accuracy) ) do
              (7) Set $stop \leftarrow true$.
            end if
    end while
    Filtering results to exclude matching supports at joints of beams.
```

The linear Simplex method is used to solve optimization subproblems in steps (4) and (5).

3.3 Global Optimization

The most simple global optimization algorithm is obtained by starting local search algorithm from many different trial points. Such strategy can be easily implemented in parallel (see, [BBC02]). In some cases it gives sufficiently good solutions. The quality of the obtained solution depends on the initial trial points, and some heuristic for the selection of such initial approximations should be given (see [BVM02]).

In general the probability of finding the global minimum can be increased by using a big number of starting points of local searches and it approaches one only when the number of trial points approaches the infinity (or when we have some apriori information about the distribution of possible solutions).

Here we have applied a more sophisticated method, which is based on branch and bound (BB) algorithms. Any BB algorithm consists of two main steps:

a) branching rule, i.e. we select a subdomain from D and divide it into two or more smaller parts,

b) computation of lower bounds on the objective function value for each new subdomain.

If the lower bound of the objective function is lager than the best known approximation of the value of this function, then such a subdomain is taken out from further searches.

The starting point of our method is the algorithm developed by Žilinskas [Zil01]. This method can be interpreted as a strategy for managing local searches in a search for global minimizers. Once a local minimizer has been found by the local search method described above, a domain around it is taken out from farther searches. We use the branch strategy from [Zil01].

Main steps of the BB algorithm

1. The monotonicity test, which tries to prove that no local minimum exists in the tested subdomain;

2. Once a local minimizer has been found, a domain around it is taken out from father searches;
3. A subdivision step, where the estimation of the lower bound is computed, if needed.

We note that the lower bound of the objective function can be computed only approximately, since the objective function is not given explicitly. Thus we can not guarantee that the exact global minimum solution is obtained. But in real world applications we are mostly interested in finding a sufficiently good solution, since the problem is too large to be solved exactly.

3.4 Numerical Examples

To illustrate the proposed technology, two support placement schemes for relatively simple grillages were investigated. It should be noted, that some grillage schemes expose extreme sensitivity to the positions of supports. Small changes of supports' coordinates may lead to significant perturbations in reactive forces. The two examples below belong just to this category of schemes. Earlier, using local optimization methods, we even did not obtain a reasonable solutions for these problems. As to the results of examples, there the global solution, i.e., an even distribution of reactive forces across the scheme was not achieved, because the design program stops optimization after the allowable reaction is reached.

Example 1.

Grillage of rectangular shape is loaded with two sets of distributed vertical loadings (Fig. 1a). Construction of grillage consists of standard prefab reinforced concrete girders. The most important data for support scheme is given by the maximum allowable vertical reaction, the minimum allowable distance between two adjacent supports, and the vertical stiffness of support, and it is fixed to 200, 0.210, and 1.e15, accordingly. Theoretical number of supports is 10. All reactive forces for supports, ordered as shown in Fig. 1, are the following:

$$-206.3, -170.6, -204.5, -189.0, -94.96,$$
$$-196.8, -168.6, -207.5, -192.1, -207.4.$$

Example 2.

Grillage consists of two rectangular frames under distributed loadings (see Fig. 2a). Theoretical number of supports for initial data on limiting factors 150, 0.10, 1.e10 is 15. The obtained reactions are the following:

$$-156.4, -158.1, -147.4, -161.2, -153.6, -118.8, -130.6,$$
$$-161.3, -161.5, -139.7, -161.2, -144.0, -94.37, -121.1, -136.4.$$

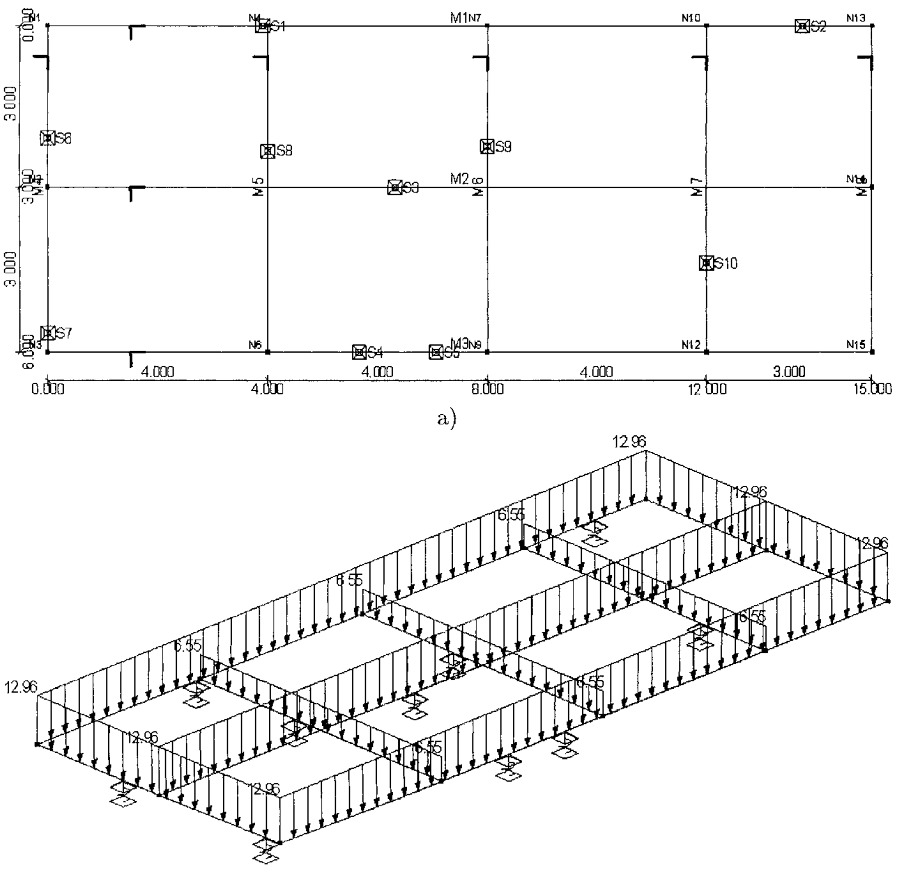


Fig. 1. Example 1: a) geometry of the grillage, b) loadings and the obtained optimized pile placement scheme

Computational experiments were performed on a PC cluster of Vilnius Gediminas technical university by using the template library of generalized parallel BB algorithms (see [BCZ05]).

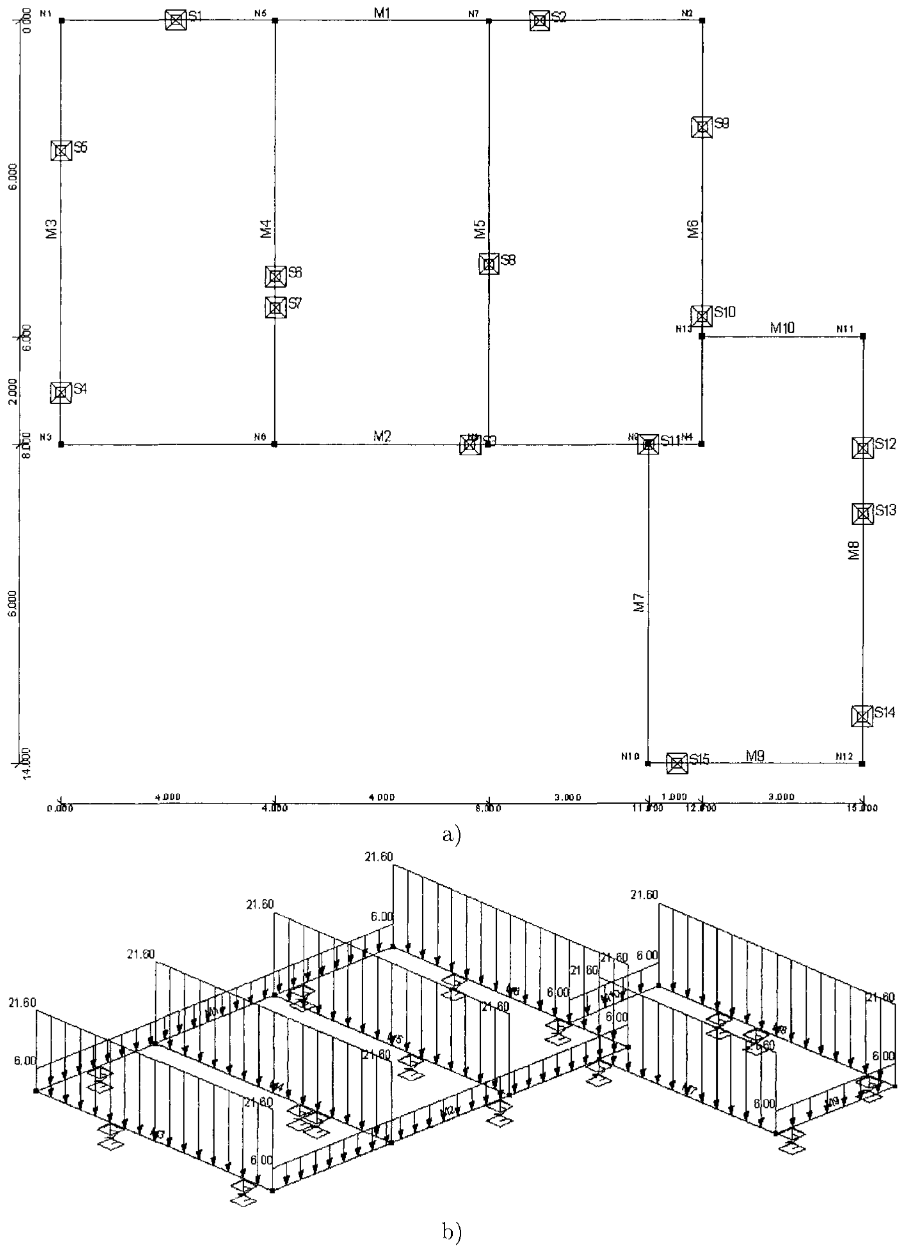


Fig. 2. Example 2: a) geometry of the grillage, b) loadings and the obtained optimized pile placement scheme

References

[BBC02] Baravykaitė, M., Belevičius, R., Čiegis, R.: One application of the parallelization tool of Master-Slave algorithms. Informatica, **13**(4), 393–404 (2002)

[BCZ05] Baravykaitė, M., Čiegis, R., Žilinskas, J.: Template realization of generalized Branch and Bound algorithm. Matematical Modelling and Analysis, **10**, 217–236 (2005)

[BIS99] Baronas, R., Ivanauskas, F., Sapagovas, M.: Modelling of wood drying and an influence of lumber geometry on drying dynamics. Nonlinear Analysis: Modelling and Control, **4**, 11–21 (1999)

[BVM02] Belevičius, R., Valentinavičius, S., Michnevič, E.: Multilevel optimization of grillages. Journal of Civil Engineering and Management, **8**(1), 98–103 (2002)

[CBB04] Čiegis, R., Baravykaitė, M., Belevičius, R.: Parallel global optimization of foundation schemes in civil engineering. In: Dongarra, J., Madsen, K, Wasniewski, J. (eds) PARA04, Workshop on State of the Art in Scientific Computing. Lecture Notes in Computer Science, Vol. **3732**, Springer, 305–312 (2005)

[CS02] Čiegis, R., Starikovičius, V.: Mathematical modeling of wood drying process. Mathematical Modelling and Analysis, **7**(2), 177–190 (2002)

[CSS04] Čiegis, R., Starikovičius, V., Štikonas, A.: Parameters identification algorithms for wood drying modeling. In: Buikis, A., Čiegis, R., Fitt, A.D. (eds) Progress in Industrial Mathematics at ECMI2002. Mathematics in Industry-ECMI Subseries, Vol. **5**, Springer, Berlin Heidelberg New York, 107–112 (2004)

[KLCL04] Kim, K., Lee, S., Chung, C., Lee, H.: Optimal pile placement for minimizing differential settlements in piled raft foundations. http://strana.snu.ac.kr/laboratory/publications (2004)

[Ped89] Pedersen, P.: Design for minimum stress concentration – some practical aspects. In: Structural Optimization. Kluwer Academic, 225–232 (1989)

[PM95] Perre, P., Mosnier, S.: Vacuum drying with radiative heating. Vacuum Drying of Wood, **95**, (1995)

[PSO85] Plumb, O., Spolek, G., Olmstead, B.: Heat and mass transfer in wood during drying. Int. J. Heat Mass Transfer, **28**(9), 1669–1678 (1985)

[SL97] Simpson, W.T., Liu, J.T.: An optimization technique to determine red oak surface and internal moisture transfer coefficients during drying. Wood Fiber Sci., **29**(4), 312–318 (1997)

[TA86] Tichonov, A., Arsenin, V.: Methods for Solution of Ill-posed Problems. Nauka, Moscow (1986)

[Zil01] Žilinskas, J.: Black box global optimization inspired by interval methods. Information Technology and Control, **21**(4), 53–60 (2001)

A Two Step Hybrid Optimization Procedure for the Design of Optimal Water Distribution Networks

Eric S. Fraga[1] and Lazaros G. Papageorgiou[2]

[1] Centre for Process Systems Engineering, Department of Chemical Engineering, UCL (University College London), UK e.fraga@ucl.ac.uk
[2] Centre for Process Systems Engineering, Department of Chemical Engineering, UCL (University College London), UK l.papageorgiou@ucl.ac.uk

1 Introduction

The design of a water distribution network [AS77] involves identifying the optimal pipe network, the head pressures of the individual supply and demand nodes, and the flows between the nodes, including both the amount and the direction of flow. The objective is to find the minimum cost network which meets the demands specified. Despite the objective function often being simple, consisting of a linear combination of pipe diameters and lengths, the water distribution network design problem poses challenges for optimization tools due to the tight nonlinear constraints imposed by the modelling of the relationship between node heads, water flow in a pipe, and the pipe diameter.

The optimization problem, as described above, can be modelled as a mixed-integer nonlinear programme (MINLP). This problem has been tackled through a variety of methods including mathematical programming [AS77, GM85] and stochastic procedures [CS99, GGK99, SW97]. Wu & Simpson [WS02] categorise methods for the water distribution network optimization problem into those that make use of a hydraulic network solver, such as the EPANET software, originally developed by L. Rossman,[3] and those that do not. The latter have been typically solved using a two-stage approach with a non-linear programming (NLP) outer stage and a linear programming (LP) inner stage. The methods that use an embedded hydraulic network solver are based on stochastic methods (cf. for instance, [SW97, CS99, MSZ+03] and references cited therein).

Due to the multi-modal nature of the problem, mathematical programming tools will typically only identify a local optimum. The actual local optimum found will depend on the initial point used for the optimization solver.

[3] http://www.epa.gov/ORD/NRMRL/wswrd/epanet.html

More importantly, many optimizers may fail if no reasonable initial guess is provided. Table 1 shows how the quality of solution obtained can vary for different solvers for a specific initial condition (all flows set to 100). In fact, all of these methods failed to find any solution at all if no initial condition was specified.

Stochastic methods have therefore been a popular choice to overcome the multimodal nature of the solution space. However, even the stochastic methods cited above require an embedded solution of systems of nonlinear algebraic equations leading to difficulties in initialization and handling convexity.

Table 1. Comparison of solutions obtained with different combinations NLP solvers and MINLP solution approaches with a given initial condition for the Hanoi problem (Example 2 below) [SW97]

NLP Solver	DICOPT	SBB
MINOS	Fails	Fails
CONOPT	6.295	6.272
CONOPT2	Fails	6.295
CONOPT3	6.295	6.064

In this chapter, we wish to consider a two step procedure. The first step is a targetted genetic algorithm, using discretization and graph theoretic methods, and is based on a single stage model which solves an approximation to the complete problem, choosing pipe diameters simultaneously with the determination of the node heads and flow patterns in the network. Using a stochastic method, combined with interval analysis to alleviate the effects of discretization on the search space, this first step will identify solutions which are close to the globally optimal solution. The second step consists of a deterministic mathematical programming approach, using standard solvers, such as those available through NEOS[4] [CMM98], which will be initialized with the solutions obtained in the first step and essentially acts as a fine-tuning step.

An early study into the development of the novel two-step procedure was presented in [FPS03]. This demonstrated that a simple procedure, combining user interaction via visualization with a stochastic optimizer using discretization, can improve the consistency of the results obtained by the subsequent mathematical programming step. Although promising, the procedure described in [FPS03] was limited to small problems, such as the 7 node, 8 pipe example from [AS77]. In this chapter, we extend and enhance the basic concepts of this two step approach to cater for much larger problems, such as the 32 node, 34 pipe Hanoi problem [SW97] and the 20 node, 24 pipe problem from Goulter & Morgan [GM85].

[4] `http://www-neos.mcs.anl.gov/neos/`

2 The Model

Both the first step genetic algorithm and the subsequent mathematical programming step use the following model for the water distribution network. We wish to find the minimum cost network,

$$\min \sum_{j=1}^{n_p} \sum_{k=1}^{n_d} C_k L_j y_{jk}. \tag{1}$$

There are n_p pipes in the network, indexed with j, and n_d different diameters possible for each pipe, indexed with k. C_k is the cost, per unit length, of a pipe with diameter d_k; L_j is the length of the pipe j connecting a pair of nodes; and, y_{jk} is 1 if pipe j has diameter d_k.

There is a mass balance constraint on each node i, $i = 1, \ldots, n_n$,

$$\sum_{j \in \mathcal{P}_i} \delta_{ij} Q_j = D_i, \tag{2}$$

where Q_j is the flow, positive or negative, along pipe j; D_i is the demand of water at node i; and, $\mathcal{P}_i$ is the set of pipes incident on node i. The flow, Q_j, is relative to the initially chosen direction for the pipe, as indicated by δ_{ij}, where a value of $+1$ indicates a pipe j coming into node i and -1 indicates a pipe leaving the node.

There is also an *energy* balance across each pipe j, $j = 1, \ldots, n_p$,

$$\Delta H_j = H_{i \in \mathcal{I}_j} - H_{i \in \mathcal{O}_j}, \tag{3}$$

where H_i is the node head for node i and E_i is the elevation of node i. $\mathcal{I}_j$ is the set of source nodes for pipe j and $\mathcal{O}_j$ is the set of destination nodes for pipe j. The energy balance can equivalently be expressed in terms of loops in the network. The head loss, ΔH_j, is defined by the Hazen-Williams correlation,

$$\Delta H_j = w \operatorname{sign}(Q_j) \left(\frac{|Q_j|}{C_{\mathrm{HW}}} \right)^{\beta} L_j \sum_{k=1}^{n_d} d_k^{-\gamma} y_{jk} \tag{4}$$

and the head at each node must meet the minimum heat requirements,

$$H_i \geq H_i^{\min} + E_i. \tag{5}$$

Finally, only one pipe diameter can be chosen for each pipe,

$$\sum_{k=1}^{n_d} y_{jk} = 1, \tag{6}$$

where $k = 1, \ldots, n_d$ is the index into the set of pipe diameters.

Values for various parameters, such as C_{HW}, w, β and γ, are those used in the literature for the specific case studies considered.

3 A Hybrid Optimization Procedure

The hybrid two-step method proposed consists of a genetic algorithm (GA) step which solves a version of the problem in discrete space. The solution from this step provides an initial guess for the second step which is formulated as a mixed integer nonlinear programme (MINLP) and solved using the NEOS server. The GA implementation is described in detail below (§3.1 and §3.2). The mathematical programming model, implemented in GAMS[5], is described in the following section (§3.3). The transfer of the solution from the GA to NEOS is achieved using an automated procedure which creates the GAMS initialization from the solution proposed by the GA, combines the initialization with the GAMS model for the water distribution network problem and finally emails the combined GAMS model and initialization information to the NEOS server. The NEOS server solves the model and emails the output of the GAMS system back. The final results are collated from the email responses. The procedure is illustrated in Fig. 1. All but the final collation step are fully automated. The novel contributions of the work presented here are not only the individual optimization steps but also the combination of the two to create a hybrid globally optimal and robust procedure for tackling the water distribution network problem.

Genetic algorithms are stochastic methods based on the analogy of survival of the fitness in evolution [Smi02, Gol89]. Developing a genetic algorithm for a specific problem requires the definition of the following aspects: the encoding used to represent alternative solutions, the genetic operators required to create new solutions and the selection procedure to use for selecting the solutions to manipulate with the genetic operators. These aspects, as they have been defined for the water distribution network problem, are described below.

Previous use of stochastic procedures [AS77, CS99, GM85, GGK99, MSZ+03, SW97, WS02] have addressed solely the combinatorial aspects of the problem: the choice of discrete pipe diameters. The actual node heads have been determined with an embedded simulation system, typically the EPANET system. Our approach, however, addresses the discrete and continuous variables simultaneously. Therefore, the genetic encoding used must represent both aspects.

3.1 A Discrete Spanning Tree Encoding

The water distribution network optimization problem is modelled by a directed graph with nodes representing either a source of water or a demand for water and edges representing the possible connections of nodes with pipes. The optimization decision variables for our approach are the n_n node heads and the n_p pipe diameters. We use discretization to map the continuous node heads, H_i, $i = 1, \ldots, n_n$, to discrete space. In order to provide support for

[5] `http://www.gams.com/`

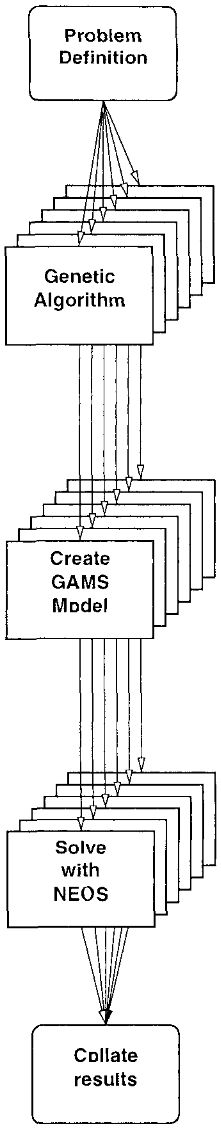


Fig. 1. Two step hybrid optimization procedure

visualization (not necessary but quite a useful bonus), the discretization parameters are chosen to be suitable for mapping to a pixel index on a display, essentially limiting the range of values allowed as a typical display will have on the order of a thousand pixels in each dimension. As the diameter selection for each pipe is already a discrete variable, one genetic encoding could consist of $n_n + n_p$ integer values, the first n_n being the node heads and the remaining n_p values being the diameter selection for each of the pipes. However, this encoding alone is neither effective nor efficient. There are two issues:

1. The use of discretization means that it is highly unlikely that the values of H chosen from the discrete space will satisfy the equality constraint of the model (mass balance).
2. The domain for the discrete values of H does not take into account the connectivity of the network and hence the search space encoded is potentially much larger than the feasible space.

The first issue is addressed by using *interval* arithmetic and the second through graph theory.

Interval Arithmetic for Estimating Mass Balance Constraint

The key equation in the water network model is the mass balance around each node, $\sum_{j \in P_i} \delta_{ij} Q_j = D_i$, $i = 1, \ldots, n_n$. The mass balance requires the calculation of Q_j for each pipe. Q_j is a function of the node heads for the nodes the pipe connects. In discrete space, the node heads are restricted to a set of discrete values so the mass balance is unlikely to be exactly satisfied. However, as the aim of the first step, the stochastic optimization, is to provide *good* initial points for the subsequent mathematical programming step, we are content if the mass balance (the first equation in the constraints for (1)) is *close* to satisfaction. The definition of *close* we use is

$$0 \in \sum_{j \in \mathcal{P}_i} \delta_{ij} Q_j - D_i \qquad i = 1, \ldots, n_n \tag{7}$$

where the Q_j values have been calculated using interval arithmetic by

$$Q_j = \frac{1}{C_{HW}} \left(\frac{\Delta H_j}{\omega L_j \sum_k d_k^{-\gamma} y_{jk}} \right)^{\frac{1}{\beta}}$$

with ΔH_j the difference in node heads for the source and destination ends of pipe j, using interval values defined by the discretization of the head domain. Equation (7) is satisfied if there is some combination of values of Q_j and D_i within their respective intervals which balance.

Spanning Tree Enumeration for Dynamic Node Head Domain

A common feature of water distribution networks is low connectivity with most nodes having at most two or three edges. Graph theoretic approaches may be suitable for manipulating these networks, making beneficial use of their special structure.

A key property of any solution for the water distribution network problem is the flow direction in each edge. Our proposal is that the alternative flow configurations can be characterised by the set of spanning trees of the graph representing the water distribution network connectivity. Each spanning tree is a directed graph starting at source nodes and therefore defines the relative positions of all the nodes in the network graph in terms of the head pressure at each node. The spanning tree specifies explicitly the flow direction across a sub-set of the pipes in the network; the flow direction in each of the remaining pipes, those not in the spanning tree, is defined implicitly subsequently by the pressure at each node. Therefore, the full network is used for optimization but the spanning tree defines the direction of flow. Therefore, we have defined a genetic algorithm encoding which includes the specification of the spanning tree defining the flow pattern.

The list of all spanning trees for a network graph are generated using a simple algorithm, described in Appendix A, suitable for graphs of the size and type typically encountered in water distribution problems.

Decoding a Chromosome

The proposed encoding is

$$\begin{array}{|c|c|c|c|c|c|c|}\hline t & h_1 & \dots & h_{n_n} & d_1 & \dots & d_{n_p} \\ \hline\end{array}$$

where t is the index into a list of all the spanning trees for the network, h_i are integer values in$[0, n_l - 1]$, for a given number of discrete levels l, and d_j are integer values in $[1, n_d]$, with n_d the number of discrete pipe diameters allowed. This chromosome is decoded by first selecting spanning tree t from the set of spanning trees generated during the initialization procedure. The spanning tree is used to determine the bounds for the head of each node in the graph dynamically.

The node heads are represented using integer values due to the use of discretization. To support visualization, the value of the node head is used as the vertical placement of the graphical representation of the node (see Fig. 2 for an example and [FPS03] for a discussion on visualization for water distribution networks). Because graphical displays number pixels vertically starting at the top of the screen, the integer values for node heads increase as the node head decreases (inverse linear mapping). Therefore, the lower bound for a node head, in integer representation, is actually the upper bound for the node's placement and vice versa for the lower bound. Given this, the upper bound for a given node will be the minimum head allowed (given as part of

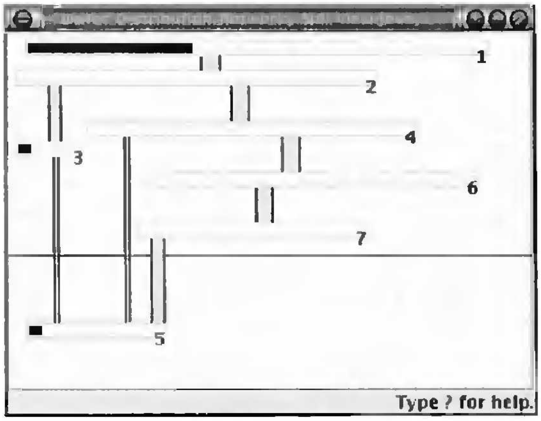


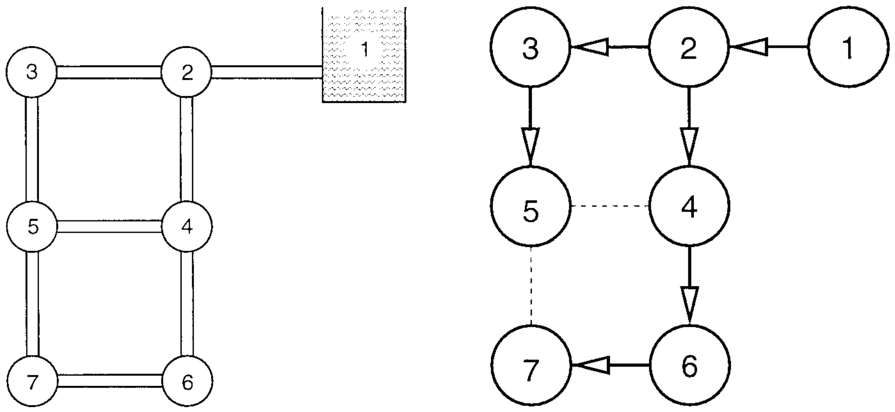
Fig. 2. Example visualization for case study 1

the problem definition) mapped to an integer value. The lower bound will be one discrete level higher than the node head integer value of the parent node. A parent node is the immediate precursor of the node in the path from the source node to this node in the spanning tree. Algorithm 8 shows how this procedure is performed.

```
Given: chromosome, $c$; lower bounds, $\mathbf{a}$; upper bounds, $\mathbf{b}$;
Outputs: Node locations, $\mathbf{y}$, based on dynamic bounds
 1: for each edge $e_i$ in tree$_{c.t}$ using depth first traversal do
 2:   $p \leftarrow e_i.parent$
 3:   if $p \geq 0$ then {if not a root node}
 4:     $\tilde{a} \leftarrow y_p + 1$ {lower bound is position just below parent node}
 5:   else
 6:     $\tilde{a} \leftarrow a_i$ {use default lower bound for node $i$}
 7:   end if
 8:   {Calculate actual node location using lower bound determined above}
 9:   $y_i \leftarrow \frac{c.h_i}{n_l}(b_i - \tilde{a}) + \tilde{a}$
10: end for
```

Algorithm 8: Definition of dynamic node head bounds from spanning tree

By way of example, consider the Alperovits & Shamir network. The network is shown in Fig. 3(a) and one possible spanning tree is shown in 3(b). The spanning tree is indicated by the solid lines with arrowheads, showing the dependency of each node on its parent. Dashed lines indicate that the relative position of the two nodes connected by this line is undefined. The spanning tree illustrated could lead to the node placements shown in Fig. 2. In this actual placement, we see that node 4 is above node 5 and node 7 is



(a) Water distribution network (b) Example spanning tree

Fig. 3. Small example from Alperovits & Shamir [AS77]

also above node 5. There are other spanning trees that could lead to the same node placement, however. An example would be converting the line from 3 to 5 in Fig. 3(b) to a dashed line and the dashed line between 5 and 7 into an arrow from 7 to 5.

3.2 GA Parameters

A genetic algorithm must be tailored for the problem tackled. The encoding used is often the key consideration, and this is described above. This section describes the other parts of the genetic algorithm, including the choice of parameters:

population A replacement approach was used where a new population is generated at each generation using selection, copying, crossover and mutation. The population size is denoted by n and varies for each case study.

selection A tournament based selection procedure with a tournament of size 2 is used. This caters easily for populations with a mix of feasible and infeasible solutions [Deb00]. The fitness function used is the objective function if the function is feasible and the constraint violation alone if not. The constraint violation is defined as

$$\sum_{j=1}^{n_n} r_j$$

where

$$r_j = \begin{cases} \frac{D_j + \sum_{k \in O_j} Q_k - \sum_{k \in I_j} Q_k}{D_j - \sum_{k \in O_j} Q_k} & D_j + \sum_{k \in O_j} Q_k - \sum_{k \in I_j} Q_k > 0 \\ 0 & \text{o.w.} \end{cases}$$

and represents a relative measure of the demand (node demand and requirements of downstream nodes) not met. To ensure that feasible flow patterns are favoured, r_j is multiplied by 100 if node j has no upstream nodes, unless it is a source node. Although the use of a spanning tree encoding is designed to avoid this situation, it minimizes the number of such occurences but does not eliminate them.

diversity Ensuring sufficient diversity in a population is often difficult. See below (§3.2) for a description of how this is tackled.

crossover A crossover rate of 0.7 is used in all cases presented below. Higher crossover rates led to reduced performance. Ten percent of the time, a single point crossover between the spanning tree index and the rest of the chromosome is used; the rest of the time, a full multi-point crossover technique is used as the order of the values in the chromosome encoding described above is completely arbitrary.

mutation A mutation rate of 0.1 was used. When a mutation is requested, 10% of the time, the mutation chooses the spanning tree index as the value to mutate. The rest of the time, a value from the rest of the chromosome is chosen arbitrarily. The value chosen is changed to a randomly selected value in the domain for the particular variable.

elite An elite set of size 1 was used to ensure that the best solution encountered was not lost.

Two other aspects of the implementation need to be discussed. These are the discretization parameters required by the encoding and the need to ensure sufficient diversity.

Discretization Parameters

There are two discretization parameters, the number of discrete values for node heads and the number of discrete values for the encoding. Node head discretization is applied to the domain of the problem and there must be sufficient levels to ensure that all potential layouts can be generated. If this value is too small, it may be impossible to place one node above or below another, limiting the search space unduly. Typically, we have used a simple rule: the number of discrete node head values is 5 times the number of nodes, allowing for a minimum, on average, of three discrete values between any two nodes when placed randomly.

The second discretization parameter must be sufficiently large to discriminate between the different positions that a node would be allowed to take, within the discrete space of node heads, but small enough to ensure the search space for the genetic algorithm is not overwhelming. The choice of this parameter is based on analysis of the search space. Given a suitable value for

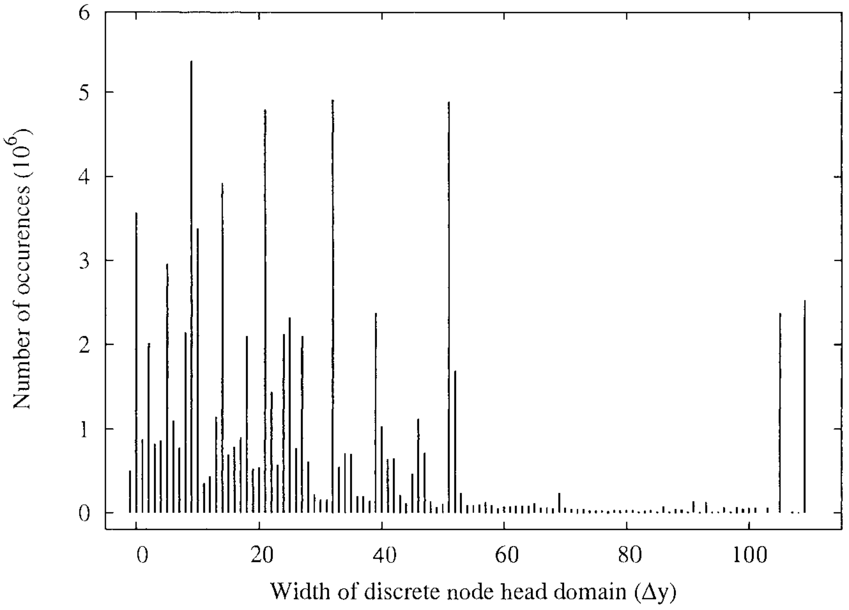


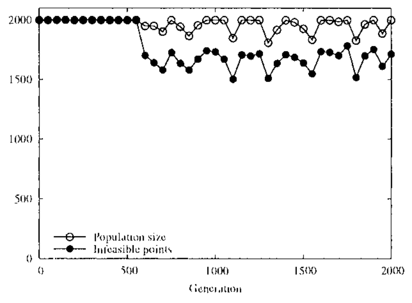
Fig. 4. Distribution of dynamic domain for individual node head values

the node head discretization, a large value for the number of levels in the encoding is used and the problem is solved once. The occurences of each discrete value in the node head domain is analyzed and a reasonable number of levels is chosen from this analysis. The results of this are shown in Fig. 4. Removing outliers, especially at the right end of the distribution, allows us to conclude that the suitable discretization for the dynamic domain should have 60 values, approximately, for this particular example.

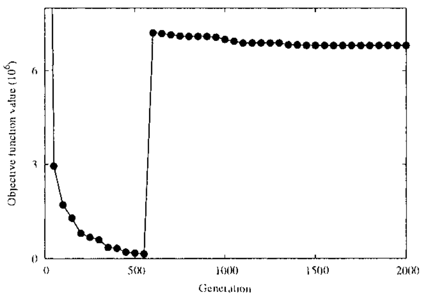
Ensuring Diversity

Ensuring diversity in a population for genetic algorithms can be key to the success of the procedure. Premature convergence will lead to the identification of a local optimum. Many techniques have been proposed for ensuring diversity; see for instance [Ron95, Zhu03]. Although potentially suitable for our procedure, a simpler approach was taken. Our experience with the water distribution network problem is that the flow layout is key, especially in the context of generating good initial guesses for a subsequent mathematical progrmaming step. Therefore, the diversity we are most interested in is the different flow patterns. These are best represented by the spanning trees in the encodings. Therefore, a diversity rule is added to the procedure for deciding whether a new solution should be added to the population being generated.

If the spanning tree in the new solution is not already present, accept the solution. If it is already present, accept the new solution if its fitness is better than the existing solution. Note, the pre-existing solution with the same spanning tree is kept in the population.



(a) Population size and number of infeasible solutions



(b) Objective function evolution

Fig. 5. Population characteristics and objective function evolution for sample run of the Hanoi problem (Example 2)

This procedure is efficient, requiring just $O(n)$ comparisons per solution to be inserted in the population. A maximum number of insertion attempts is allowed to create the full population, typically $2n$. This limit is reached for some problems; see for instance the variation in the actual population size shown in Fig. 5(a). However, it appears appropriate for a wide range of problems and can easily be increased if it is found to affect the performance of the procedure. Fig. 5(a) shows the size of each population versus generation number along with the number of infeasible solutions for each population for one of the case studies. The first feasible point is found a little after the 500*th* generation. Fig. 5(b) shows the objective function value of the best solution in the population versus generation number for the same run (objective function values have been scaled for presentation to be of the same order as the penalty function values for infeasible solutions). Thereafter, the proportion of feasible versus infeasible solutions remains essentially constant. This is due to the procedure for ensuring diversity and could potentially be relaxed once feasible solutions are found.

3.3 The GAMS Model

The overall problem is formulated as an MINLP model and is solved using standard mixed integer optimization techniques through the GAMS modelling platform. The mathematical model determines all key decision variables (i.e. pipe diameters, flowrates, flow direction, head pressures) simultaneously with a single optimization step rather than adopting a decomposition-based strategy as commonly used (see for example, Alperovits and Shamir [AS77]).

One important feature of our mathematical model is that flow direction and flowrate values are determined simultaneously using the Hazen-Williams formula:

$$\Delta H_j = w \operatorname{sign}(Q_j) \left(\frac{|Q_j|}{C_{\mathrm{HW}}} \right)^{\beta} L_j \sum_k d_k^{-\gamma} y_{jk}$$

which has been implemented as follows:

$$\Delta H_j = w Q_j \frac{\left(\sqrt{Q_j^2}\right)^{\beta-1}}{C_{\mathrm{HW}}^{\beta}} L_j \sum_k d_k^{-\gamma} y_{jk}.$$

This reformulation allows the model to work with both positive and negative values of flow, Q_j, without recourse to any complicated switching procedures.

As mentioned earlier, the initialization for the solution of the MINLP model is provided by the genetic algorithm, thus increasing the likelihood of identifying the globally optimal solution as well as enhancing the robustness of the proposed two-step solution procedure, as can easily be seen in the examples section below.

3.4 Implementation

The genetic algorithm has been written in Java, as part of the Jacaranda system for design and optimization [FSBH00]. The water distribution network model ((1) with the constraints) is directly evaluated without recourse to any other system (cf. methods based on the use of EPANET). The model is written in an interpreted interval expression language provided by Jacaranda. It should be noted that the combination of Java with a further interpreted language leads to a reduction in computational performance. However, the aim of this implementation is to investigate the behaviour of the suggested approach and not to provide an off-the-shelf tool for water distribution network design at this stage. Some details of the computational efficiency, nevertheless, will be discussed in the next section.

The mathematical programming step is handled by the NEOS email server with the model written in GAMS. The automation for the overall procedure is provided by a set of Linux shell scripts and the whole system is run under Debian Linux r3.x with Java Standard Edition 1.5 from Sun[6].

[6] `http://java.sun.com/`

4 Results

The hybrid two-step procedure described above has been applied to three example problems from the literature. The examples chosen demonstrate the applicability of the new procedure.

4.1 Example 1: 7 Nodes, 8 Pipes

The first example is a small network consisting of 1 source node, 6 demand nodes and 8 pipes [AS77] and the spanning tree enumeration returns 15 distinct trees. Table 2 summarizes results from 10 genetic algorithm runs, each running for 200 generations with a population size of 200, 32 discrete node head levels and 10 discrete levels for encoding of node heads, with other parameters as described above. Diversity is based on uniqueness of the spanning tree, also described above.

Table 2. Statistics on the behaviour of the genetic algorithm and the subsequent mathematical programming method for Example 1

	Statistical analysis for 10 runs			
	min	ave	max	std dev
Best GA solution, f $(\times 10^6)$	0.399	0.418	0.509	0.033
$n_{\text{infeasible}}$	84	110	145	19
SBB/CONOPT3 solution, f $(\times 10^6)$	0.419	0.419	0.420	0.001

The first row presents statistics on the behaviour of the best objective function value, f, found for each run. The best solution obtained in all 10 runs was 399 000 whereas the average of the best found was 418 000 and the worst was 509 000. A value of 399 000 is actually lower than the best solution known and expected for this problem. This might appear to be a contradiction but is actually due to the approximation introduced through discretization and the analysis based on interval arithmetic (see (7)). The pipe diameters and node heads identified in this solution do not actually correspond to a feasible point in continuous space; however, the aim of the genetic algorithm is to provide good starting points for a mathematical programming approach and we shall see below that this is indeed the case.

The second row in the table represents the number of solutions, in the final population, which are infeasible. We see that, on average, over half the population is infeasible at the end but that all the runs succeed in finding feasible solutions, although feasible in the discrete space defined by the discretizations and interval analysis calculations. The fact that infeasible solutions are still present at the end of each GA run is an indication that the control introduced for ensuring diversity in the populations has been successful. Further analysis of the final populations, such as the average and worst fitness values also supports this view.

The solutions identified as best in each of the 10 runs are used to generate a GAMS model with an initial solution. The GAMS model has then been solved using SBB with CONOPT3 via the NEOS email interface with the results summarized in the last row of Table 2. SBB with CONOPT3 has been used as this combination was found to be the most robust in our initial studies (see, for instance, Table 1). Six of the 10 runs result in a solution of 419 000 and 4 give 420 000. Figure 6 shows the relationship between the solution obtained by each GAMS run and the objective function value for the best solution obtained by the genetic algorithm (using interval arithmetic) used as the initial guess for that GAMS run. All initial guesses provided by the 10 runs lead to mathematical programming solutions with similar values. Points above the diagonal line indicate GAMS solutions which have a higher cost than the cost of the initial guess. However, the initial guess cost is determined using interval arithmetic and, most importantly, the constraints are satisfied only in interval space. Therefore, the cost of the initial solution is approximate and may actually represent an infeasible solution, albeit one which is "close" to feasible and ideally close to the globally optimal solution. The figure clearly demonstrates the robustness of the method in the sense that all the initial guesses lead to similar final solutions.

4.2 Example 2: The Hanoi Problem

Table 3. Statistics on the behaviour of the genetic algorithm and the subsequent mathematical programming method for Example 2

	Statistical analysis for 10 runs			
	min	ave	max	std dev
Best GA solution, f $\left(\times 10^6\right)$	6.593	6.857	7.207	0.193
$n_{\text{infeasible}}$	1551	1785	2000	136
SBB/CONOPT3 solution, f $\left(\times 10^6\right)$	6.092	6.183	6.272	0.083

This example is significantly larger than Example 1, consisting of 32 nodes and 34 pipes (see [SW97], for instance, for a full description of the problem). The spanning tree enumeration algorithm identifies 1048 spanning trees. As we have already seen in Table 1, this problem is difficult to solve using mathematical programming. Table 3 presents a summary of the results for this problem using the new procedure with a population size of 2000, for 2000 generations, with 120 node head discrete levels and 60 levels for encoding. Two of the 10 genetic algorithm runs did not succeed in generating a feasible solution. The

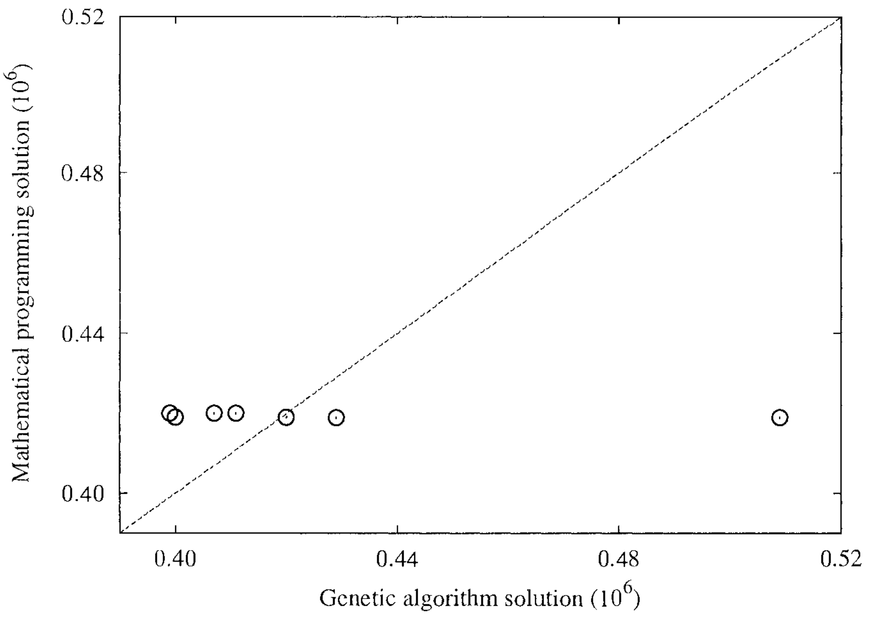


Fig. 6. Graph of the value of the mathematical programming solution value obtained versus the value of the initial guess provided by the genetic algorithm for Example 1

statistics in the first row do not include these two runs. However, the best solution found in each of those runs, having small constraint violations, have also been used as the initial guess for the mathematical programming steps and these results are included in the statistics for the last row of the table. Out of the 10 mathematical programming runs, using SBB/CONOPT3, 9 were successful. Although the unsuccessful attempt was based on a feasible initial guess (feasible in the context of discretization and interval analysis), the two initially infeasible solutions led to feasible optima. The consistency of results obtained is clear from table 3 and the results obtained are always at least as good as the results obtained our initial studies, as summarized in Table 1 for SBB/CONOPT3.

For this problem, the solution obtained in the second step always has a better objective function value than the approximate solution values obtained by the genetic algorithm. There is also clustering apparent in the distribution of results when plotting the result of the mathematical programming step versus the GA value, as shown in Fig. 7. The dashed line is the curve of $y = x$. Points below the line represent initial solutions which have a better value after solution by mathematical programming. All the points lie below this line.

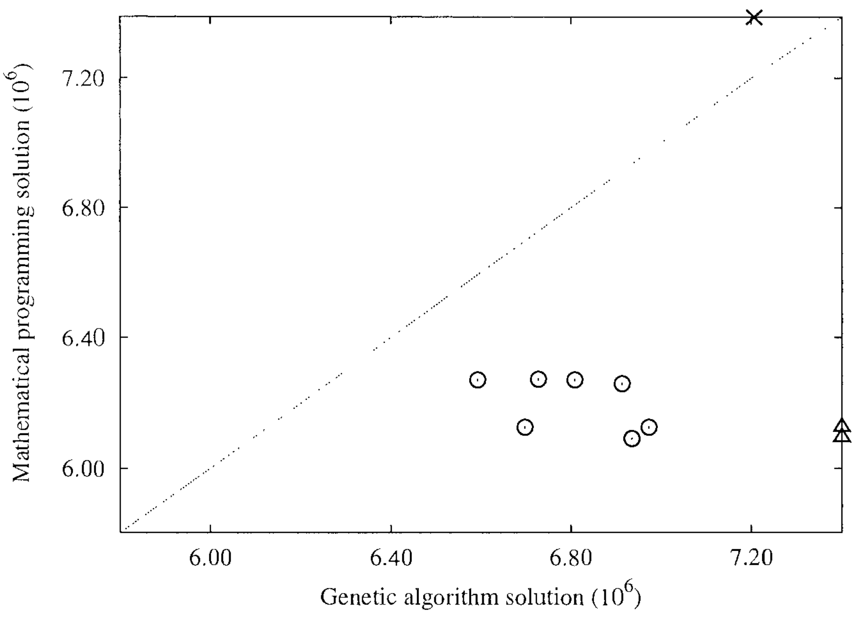


Fig. 7. Graph of the value of the mathematical programming solution value obtained versus the value of the initial guess provided by the genetic algorithm for Example 2. Triangle points ($\triangle$) indicate infeasible initial solutions (from GA runs which did not find a feasible solution) and crosses ($\times$) indicate failure in the GAMS step

4.3 Example 3: 20 Node, 24 Pipe Problem

The third example problem has been taken from the work of Goulter & Morgan [GM85]. Although not presented as a design problem by Goulter & Morgan, we have defined a design problem using the same network to provide an example with potentially different characteristics. The network has 20 nodes and 24 pipes and the pipe diameters must be chosen from a discrete set of size 11. At first glance, this problem would appear to fall between the two previous case studies with respect to size. However, the network is qualitatively different due to its connectivity. The difference is reflected in the number of spanning trees: there are 6178 for this example versus 1048 for Example 2 and 15 for the first example. The search space for the GA encoding is therefore correspondingly larger.

The results are summarized in Table 4 and Fig. 8. In this case, the GA was run using a population of 2500 for 4000 generations and with 40 discrete levels for the GA encoding. Four of the 10 GA runs led to infeasible solutions. Again, the mathematical programming step used all 10 solutions generated by the GA step, regardless of feasibility. In this case, all 10 runs led to feasible solutions in GAMS. The results in Table 4 are based on the feasible solutions only for the first row and all solutions for the subsequent rows. The figure shows that the final solutions all have a better objective function value than that found by the GA and again we see some clustering and this is supported by the statistics (small standard deviation) in the table.

Table 4. Results for Example 3

	Statistical analysis for 10 runs			
	min	ave	max	std. dev.
Best GA solution, f $(\times 10^6)$	2.07	2.12	2.22	0.05
$n_{\text{infeasible}}$	1102	1789	2501	625
SBB/CONOPT3 solution, f $(\times 10^6)$	1.92	1.95	1.98	0.03

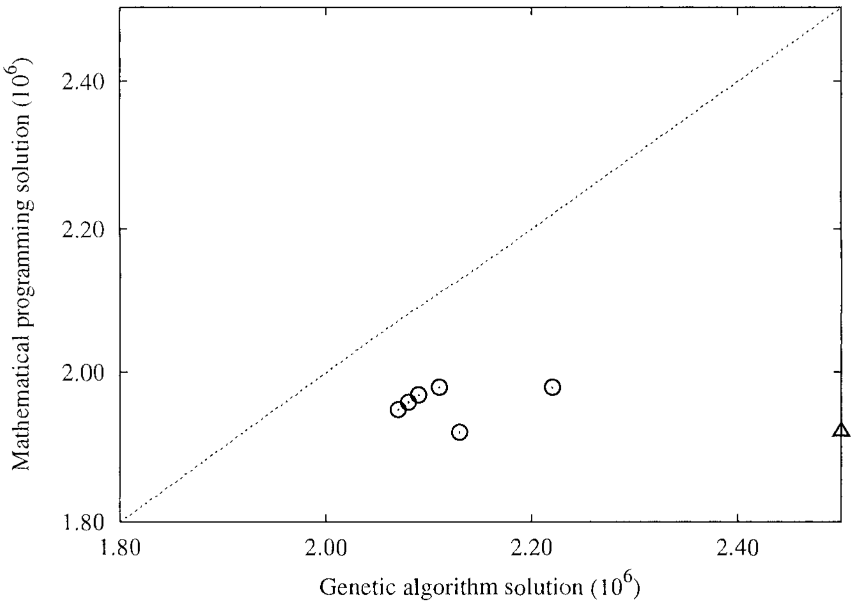


Fig. 8. Correlation between solution obtained by mathematical programming versus approximate value of initial point generated by the genetic algorithm for Example 3. Triangle points (△) indicate infeasible initial solutions (from GA runs which did not find a feasible solution)

5 Discussion

This chapter has presented a two step hybrid optimization procedure for optimizing water distribution networks. The first step consists of a targetted genetic algorithm using an encoding based on spanning trees to represent the search space and to ensure diversity in the populations created. The second step is based on deterministic mathematical programming using the NEOS server with models written in GAMS. Both steps use a single stage model including both combinatorial and hydraulic aspects simultaneously. The aim of the hybrid two-step procedure is to use the first step to generate good initial solutions for the second step. In this way, the global optimization characteristics of the first step are inherited by the second step.

Three example problems have been presented, drawn from the literature. The range of diversity of the examples is sufficient to evaluate the performance of the novel approach. The results indicate that the use of a stochastic procedure to generate initial conditions for a subsequent mathematical programming step does indeed lead to consistent results. Of all the tests performed, only once out of 30 attempts did the mathematical programming step fail to find a good solution when provided by an initial solution obtained using the genetic algorithm step. The hybrid procedure therefore provides a level of robustness not available with only the mathematical programming step. The consistency of results obtained are illustrated by the clustering of the solution values, again exhibiting behaviour better than can be achieved by either step individually.

The use of a discrete encoding supports the use of a visualization environment which can help the engineer understand the solutions obtained. Although not discussed in detail in this chapter, the visualization interface enables the engineer to manipulated solutions which, when combined with the stochastic optimizer, can lead to further improvements in the robustness of the combined system [FPS03]. In any case, the visualization interface supporst the engineer in decision making activities, providing an immediately understandable presentation of alternative solutions catering for the business analysis needs in industry.

A Generating Spanning Trees for Water Distribution Network Flow Direction

The majority of algorithms described in the literature for enumerating all the spanning have been designed for efficiency (time and/or space) when applied to large graphs (thousands of nodes or more with high connectivity). See, for instance, Shioura et al. [STU97] and Avis & Fukuda [AF96] for examples. These efficient algorithms are not required for our implementation for two reasons:

1. The network graphs for water distribution problems are small (tens of nodes and low connectivity).
2. The enumeration of all the spanning trees need only be done once.

Therefore, we have designed and implemented a simpler procedure sufficient for the needs of this work. The procedure consists of enumerating all subsets, of size $n_n - 1$, of the set of edges in the network (noting that any spanning tree of the network graph will have exactly $n_n - 1$ edges) and checking each of these sets to see if they satisfy the requirements of a spanning tree. There are two requirements to check:

1. The set of n_{n-1} edges covers all the n_n nodes in the graph.
2. A traversal of the graph, starting from any node in the graph, using only the selected edges, will visit all the nodes.

This procedure is presented in Algorithm 9 and uses Algorithm 10 to enumerate all subsets of $n_n - 1$ edges, essenially implementing an

$$\binom{n_e}{n_{n-1}}$$

operation by emulating a binary digit counter.

```
Given: $n_n$, number of nodes in the graph; $E$, set of all $n_e$ edges in the graph;
Outputs: $T$, set of spanning trees, each represented by a set of edges
1: $T \leftarrow \phi$
2: for each subset $S$ of $E$ of size $n_{n-1}$ do {See Algorithm 10}
3:   $N \leftarrow \phi$ {Will contain nodes visited by edges in $S$}
4:   for each edge $e$ in $S$ do
5:     $N \cup \{e.\text{from}, e.\text{to}\}$ {Add edge source and destination nodes to set of nodes}
6:   end for
7:   if $|N| = n_n$ then {all nodes incident}
8:     traverse graph starting from a single node, using only edges in $S$, to see if
       $S$ represents tree and not a set of disjoint graphs
9:     if tree then
10:      $T \leftarrow T \cup \{S\}$
11:    end if
12:  end if
13: end for
```

Algorithm 9: Enumeration of all the spanning trees for a water distribution network where each tree is represented by a set of $n_n - 1$ edges

References

[AF96] Avis, D., Fukuda, K.: Reverse search for enumeration. Discrete Applied Mathematics, **65**, 21–46 (1996)

```
Given: $m$, total number of elements to choose from; $n$, size of subsets to create.
Outputs: $S$, set of subsets, each containing the indices for a different choice of $n$
    elements from the $m$ available.
1: $S_0 \leftarrow \{i,\ i = 1, \ldots, n\}$ {$S$ is set of element index sets and first set consists of the
   first $n$ elements:}
2: $p_i \leftarrow \begin{cases} \textbf{true} & i = 1, ..., n \\ \textbf{false} & i = n+1, ..., m \end{cases}$ {$p$ records composition of the last set}
3: $i \leftarrow 1$
4: done $\leftarrow$ false
5: while not done do
6:   $l \leftarrow 0$ {count of elements present in current set before pair to be swapped}
7:   $j \leftarrow 0$
8:   found $\leftarrow$ false
9:   while $j < m - 1$ and not found do
10:     {Find first pair of consecutive elements where first is present in current set
   and next is not present:}
11:     if $p_j$ then {this element in current set}
12:       if not $p_{j+1}$ then {and next is not}
13:         {We swap the two elements, select the first $l$ elements present to the
   left of the swapped elements, and leave the elements to the right as they are:}
14:         $p_k \leftarrow \begin{cases} \textbf{true} & k = 1, ..., l \\ \textbf{false} & k = l+1, ..., j \end{cases}$
15:         $p_{j+1} \leftarrow$ true {Element $j+1$ added to set in place of $j$}
16:         $S_i \leftarrow \{k$ if $p_k\}$ {Create next set of $n$ elements}
17:         $i \leftarrow i + 1$ {Prepare for next set}
18:         found $\leftarrow$ true
19:       else
20:         $l \leftarrow l + 1$
21:       end if
22:     end if
23:     $j \leftarrow j + 1$
24:   end while
25:   done $\leftarrow$ not found {Finish when no more sets generated.}
26: end while
```

Algorithm 10: Choose n elements from set of size m

[AS77] Alperovits, E., Shamir, U.: Design of optimal water distribution systems. Water Resource Research, **13**(6), 885–900 (1977)

[CMM98] Czyzyk, J., Mesnier, M.P., Moré, J.J.: The NEOS server. IEEE Computing in Science and Engineering, **5**(3), 68–75 (1998)

[CS99] Cunha, M.C., Sousa, J.: Water distribution network design optimization: Simulated annealing approach. Journal of Water Resources Planning and Management, **125**(4), 215–221 (1999)

[Deb00] Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Engng., **186**, 311–338 (2000)

[FPS03] Fraga, E.S., Lazaros, G.P, Sharma, R.: Discrete model and visualization interface for water distribution network design. In: Kraslawski, A., Turunen, I. (eds) European Symposium on Computer Aided Process

Engineering – 13. Vol. 14 of Computer-Aided Chemical Engineering, pp. 119–124. Elsevier Science B.V., Amsterdam (2003)

[FSBH00] Fraga, E.S., Steffens, M.A., Bogle, I.D.L., Hind, A.K.: An object oriented framework for process synthesis and simulation. In: Malone, M.F., Trainham, J.A., Carnahan, B. (eds) Foundations of Computer-Aided Process Design. Vol. 96 of AIChE Symposium Series, pp. 446–449 (2000)

[GGK99] Gupta, I., Gupta, A., Khanna, P.: Genetic algorithm for optimization of water distribution systems. Environmental Modelling & Software, **14**, 437–446 (1999)

[GM85] Goulter, I.C., Morgan, D.R.: An integrated approach to the layout and design of water distribution networks. Civil Engineering Systems, **2**, 104–113 (1985)

[Gol89] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)

[MSZ$^+$03] Maier, H.R., Simpson, A.R., Zecchin, A.C., Foong, W.K., Phang, K.Y, Seah, H.Y, Tan, C.L.: Ant colony optimization for design of water distribution systems. Journal of Water Resources Planning and Management, **129**(3), 200–209 (2003)

[Ron95] Ronald, S.: Preventing diversity loss in a routing genetic algorithm with hash tagging. Complexity International, **2** (1995) `http://www.complexity.org.au/`

[Smi02] Smith, J.E.: Genetic algorithms. In: Pardalos, P.M., Romeijn, H.E. (eds) Handbook of Global Optimization Volume 2. Nonconvex optimization and its applications, pp. 275–362. Kluwer Academic Publishers (2002)

[STU97] Shioura, A., Tamura, A., Uno, T.: An optimal algorithm for scanning all spanning trees of undirected graphs. SIAM J. Comput., **26**(3), 678–692 (1997)

[SW97] Savic, D.A., Walters, G.A.: Genetic algorithms for least-cost design of water distribution networks. Journal of Water Resources Planning and Management, **123**(2), 67–77 (1997)

[WS02] Wu, Z.Y., Simpson, A.R.: A self-adaptive boundary search genetic algorithm and its application to water distribution systems. Journal of Hydraulic Research, **40**(2), 191–203 (2002)

[Zhu03] Zhu, K.Q.: A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In: Proceedings – 15*th* International conference on tools with artificial intelligence (ITCAI-2003). number 15, pp. 176–183, Sacramento, California, November 2003. IEEE Computer Society (2003)

Predictor-Based Self Tuning Control with Constraints

Vytautas Kaminskas

Department of Applied Informatics, Vytautas Magnus University, Vileikos 8, 44404 Kaunas, Lithuania v.kaminskas@if.vdu.lt

Summary. Design problems of predictor-based self tuning digital control systems for different types of linear and non-linear dynamical plants are discussed. Control systems based on generalized minimum variance algorithms with amplitude and introduction rate restrictions for the control signal are considered in the article.

Key words: predictor-based self tuning control, generalized minimum variance control, constraints for the control signal.

1 Introduction

Usually the design of a control system must take into account the fact that a priori information about the plant and its environment is insufficient. Control systems must also be capable of achieving the control task despite variations in the plant's dynamic and static characteristics, caused by inner and outer disturbances. Different versions of self-tuning control systems can be successfully used in order to cope with these requirements [Ise81, AW84, ML99]. In many applications, however, these nonlinearities are not known, and non-linear parameterision must be used instead. Murray-Smith and Sbarbaro [MSRG03, MS05] proposed implicit self tuning control of nonlinear systems using Gaussian process prior models, where the expected value of a quadratic cost function is minimized, without ignoring the variance of the model predictions. Richalet [Ric93], Camacho [CB95, CB98], Espinosa [EV98], Juang [JK98] focus on implementation issues for predictive control.

A self-tuning control system usually consists of two loops. The plant and the controller form the so-called main loop. The second loop may be called the tuning loop, and its aim is to change the control law to accomplish the control task. Different synthesis methods for the latter loop make it possible to group self-tuning control systems into explicit and implicit ones. An explicit self-tuning control system is based on the estimation of an explicit control plant model, while an implicit one is based on implicit estimation of the controller parameters.

This chapter considers practical issues in the implementation of a kind of explicit self-tuning control system predictor-based systems [Pet84, CMT87, Kam88]. In this case an explicit plant model is constructed in the form of an optimal predictor of the output signal.

This chapter is intended to give recommendations in design of self-tuning predictor-based control systems based on a generalized minimum variance controller with amplitude and deviation constraints on the control signals. Design problems for predictor-based self-tuning control systems for different types of control plants are discussed.

2 General Framework

Stochastic plants are considered with their output signal $y_{t+\tau+1}$ in the form of the sum of two components [Kam88]

$$y_{t+\tau+1} = y_{t+\tau+1|t}(c) + \varsigma_{t+\tau+1} \tag{1}$$

where $y_{t+\tau+1}(c)$ is the optimal $(\tau + 1)$ step prediction of the output signal; c is the vector of control plant parameters; $\varsigma_{t+\tau+1}$ is a sequence of random values.

In case of a stochastic plant it is natural to demand that the control system provide the minimum variance of the deviations of the observed sequence y_t from the reference sequence y_t^*. Sometimes it is preferable to apply a generalized minimum variance control algorithm, obtained by introducing control costing. In this case the control criterion is

$$Q_t(u_{t+1}) = M\left\{(y_{t+\tau+1} - y^*_{t+\tau+1})^2 + q_t(u_{t+1} - \tilde{u}^*_{t+1})^2\right\}, \tag{2}$$

and optimal control values can be obtained by

$$u^*_{t+1} = \arg \min_{u_{t+1}\in\Omega_u} Q_t(u_{t+1}), \tag{3}$$

where

$$\Omega_u = \{u_{t+1} : u_{\min} \le u_{t+1} \le u_{\max}, |u_{t+1} - u^*_t| < \delta_t\} \tag{4}$$

is the admissible domain for the control values; u_{min}, u_{max} are control signal boundaries; $\delta_t > 0$ are the restriction values for the control signal deviations; $y^*_{t+\tau+1}$ marks the reference trajectory for the output signal; $\tilde{u}^*_t$ marks the reference trajectory for the control signal; q_t is a weight coefficient.

Solution of the extremal problem (3) requires the knowledge of genuine plant parameters c. Since these parameters are usually unknown a priori and vary during the operation of the process, current estimates $\hat{c}_t$ can be used instead of genuine parameters. The estimates can be obtained from the identification process using the condition [Kam82]

$$\hat{c}_t : \tilde{Q}_t(c) = \frac{1}{t}\sum_{l=1}^{t} \varepsilon^2_{l|l-1}(c) \to \min_{c \in \Omega_c}, \tag{5}$$

where Ω_c is the admissible domain for the parameters, c, and is usually the same as the stability domain for the closed-loop system;

$$\varepsilon_{t+1|t}(c) = y_{t+1} - y_{t+1|t}(c) \tag{6}$$

is the error of one step prediction of the output signal; $y_{t+1|t}(c)$ is optimal one-step-prediction of the output signal.

3 Predictor-Based Self-tuning Control of Linear Plants

We consider simple linear plants with a stochastic component in their output signal. Operation of such a plant can be described by the following difference equations

$$y_t = W_0(z^{-1})u_{t-\tau} + H(z^{-1})\xi_t, \tag{7}$$

where

$$W_0(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})}, \tag{8}$$

$$A(z^{-1}) = 1 + \sum_{j=1}^{n_a} a_j z^{-j}, B(z^{-1}) = \sum_{j=0}^{n_b} b_j z^{-j}, \tag{9}$$

$$H(z^{-1}) = \frac{P(z^{-1})}{R(z^{-1})}, \tag{10}$$

$$P(z^{-1}) = 1 + \sum_{j=1}^{n_p} p_j z^{-j}, R(z^{-1}) = 1 + \sum_{j=1}^{n_r} r_j z^{-j}, \tag{11}$$

are transfer functions of the minimum-phase system with stable parts for the control signal u_t and the disturbance ξ_t, and assuming their numerator and denominator polynomials having no common roots; z^{-i} is an i-step backward-shift operator; y_t is an observable output signal; ξ_t is a sequence of independent random variables with zero mean and a finite variance σ_ξ; τ is pure delay value in the control channel.

Optimal $(\tau + 1)$-step prediction of the output signal for the plant (7) can be defined by the following equation [Kam88]

$$y_{t+\tau+1|t} = z^{\tau+1}[1 - \tilde{H}^{-1}(z^{-1})]y_t + \tilde{H}^{-1}(z^{-1})W_0(z^{-1})u_{t+1}. \tag{12}$$

Here the following relationships are true

$$\tilde{H}(z^{-1}) = E^{-1}(z^{-1})H(z^{-1}), \tag{13}$$

$$P(z^{-1}) = R(z^{-1})E(z^{-1}) + z^{-(\tau+1)}L(z^{-1}), \tag{14}$$

$$E(z^{-1}) = 1 + \sum_{i=1}^{n_e} e_i z^{-i}, L(z^{-1}) = \sum_{i=0}^{n_l} l_i z^{-i}, \tag{15}$$

$$\varsigma_t = E(z^{-1})\xi_t, \sigma_\varsigma^2 = (1 + \sum_{i=1}^{n_e} e_i)\sigma_\xi^2, \tag{16}$$

$$n_e = \begin{cases} \tau, & \text{if} \quad n_r > 0 \\ \min(\tau, n_p) & \text{if} \quad n_r = 0 \end{cases} \qquad n_l = \max\{n_r, n_p - \tau\} - 1, \tag{17}$$

$c^T = (a_1, \dots, b_0, \dots, p_1, \dots, r_1, \dots)$ is the parameter vector for the control plant (7). One-step-prediction of the output signal for the plant (7) can be defined by

$$\begin{aligned} y_{t+1|t}(c) &= z[1 - H^{-1}(z^{-1})]y_t + H^{-1}(z^{-1})W_0(z^{-1})u_{t+1-\tau} \\ &= z[H(z^{-1}) - 1]\varepsilon_{t|t-1}(c) + W_0(z^{-1})u_{t+1-\tau}. \end{aligned} \tag{18}$$

Solving (3) we get the controller equations

$$u_{t+1}^* = \begin{cases} \min\{u_{\max}, u_t^* + \delta_t, \tilde{u}_{t+1}\} & \text{if} \quad \tilde{u}_{t+1} \geq u_t^*, \\ \max\{u_{\min}, u_t^* - \delta_t, \tilde{u}_{t+1}\} & \text{if} \quad \tilde{u}_{t+1} \leq u_t^*, \end{cases} \tag{19}$$

$$\begin{aligned} \tilde{u}_{t+1} &= \left[\hat{W}_{0,t}(z^{-1}) + \alpha_t\right]^{-1} \\ &\times \left\{\tilde{y}_{t+\tau+1} + z^{\tau+1}\left[E_t(z^{-1}) - H_t(z^{-1})\right]\varepsilon_{t|t-1}(\hat{c}_{t-1})\right\}, \end{aligned} \tag{20}$$

where

$$\tilde{y}_{k+\tau+1} = \begin{cases} y_{k+\tau+1}^* + \alpha_k \tilde{u}_{k+1}^* & \text{if} \quad k = t \\ y_{k+\tau+1|k} + \alpha_k \tilde{u}_{k+1}^* & \text{if} \quad k = t-1, \ t-2, \dots \end{cases} \tag{21}$$

$\alpha_t = q_t/\hat{b}_{0,t}$; $\hat{b}_{0,t}$ is the current estimate of b_0 in $B(z^{-1})$. The current parameter estimates $\hat{c}_t$ are obtained in the process of identification of the closed loop by applying a recursive algorithm [Kam82, Kam88].

In the case where there are no restrictions for the control values, the control algorithm is defined only by (20), taking into account that $\tilde{y}_{k+\tau+1} = y_{k+\tau+1}^*$ for all k.

In case of (7) with an unstable control signal part, it is necessary to consider a plant operation model with transfer functions (9) and (11) which have equal denominators, i.e. with $R(z^{-1}) \equiv A(z^{-1})$. Then, instead of (18) and (20) we have the equations

$$y_{t+1|t}(c) = P^{-1}(z^{-1})\left\{z[P(z^{-1}) - A(z^{-1})]y_t + B(z^{-1})u_{t+1-\tau}\right\} \tag{22}$$

and controller equations (19) with

$$\begin{aligned} \tilde{u}_{t+1} &= [\hat{B}_t(z^{-1}) + \alpha_t \hat{A}_t(z^{-1})]^{-1} \\ &\times \{\hat{A}_t(z^{-1})\tilde{y}_{t+\tau+1} - \hat{L}_t(z^{-1})\varepsilon_{t|t-1}(\hat{c}_{t-1})\}. \end{aligned} \tag{23}$$

Let's discuss two possible ways of constructing predictor-based self-tuning control systems for non minimum-phase plants.

The generalized minimum variance control algorithm is capable of coping with this problem by means of adequate choice of the coefficient q_t [Cla84, Cla94]. In this case the coefficient q_t might be considered as a root-locus parameter, and it can force the potentially unstable roots due to $B(z^{-1})$ to migrate towards the roots of $A(z^{-1})$. If there are several possible q_t values (or a set of them), then an additional criterion can be applied in order to choose an appropriate q value, e.g.

$$I(q_t) = \sum_{i=1}^{n_s} |z_i(q_i)|^2 \to \min_{q_t}, \tag{24}$$

where $z_i(q_t)$ are the roots of the equation

$$S^*(z) = 0, \quad S^*(z) = z^{n_s} S(z^{-1}), \tag{25}$$

$$S(z^{-1}) = B(z^{-1}) + \alpha_t A(z^{-1}) = \sum_{i=1}^{n_s} s_i z^{-i}, \quad n_s = \max\{n_a, n_b\}. \tag{26}$$

Sometimes a certain polynomial $S(z^{-1})$ might be used instead of the coefficient q_t. There might be cases when the minimum generalized variance controller is incapable of coping with the non minimum-phase plant, i.e. there are no appropriate q_t values, or the values result in far too large losses in the control quality. In such cases other methods might be applied, e.g. factorization methods [AW84]

The polynomial $B(z^{-1})$ must be decomposed into two factors

$$B(z^{-1}) = B_+(z^{-1})B_-(z^{-1}), \tag{27}$$

where

$$B_+(z^{-1}) = 1 + \sum_{i=0}^{n_b^+} b_i^+ z^{-i} \tag{28}$$

is a polynomial with all of its roots outside the unit circle;

$$B_-(z^{-1}) = \sum_{i=0}^{n} b_i^- z^{-i}, n = n_b - n_b^+ \tag{29}$$

is a polynomial with all of its roots in the unit circle or on its boundary. In this case the polynomial $B(z^{-1})$ in the controller equations must be substituted by a polynomial

$$\tilde{B}(z^{-1}) = B_+(z^{-1})\tilde{B}_-(z^{-1}), \tag{30}$$

$$\tilde{B}_-(z^{-1}) = \sum_{i=0}^{n} b_{n-i}^- z^{-i}. \tag{31}$$

In the case of a linear system with internal feedback, its operation may be defined by the following difference equations

$$y_t = W_0(z^{-1})v_{t-\tau} + H(z^{-1})\xi_t, \tag{32}$$

$$v_t = u_t + \mu_t, \mu_t = W_F(z^{-1})y_{t-\tau'}, \tag{33}$$

where

$$W_F(z^{-1}) = \frac{F(z^{-1})}{K(z^{-1})}, \tag{34}$$

$$K(z^{-1}) = 1 + \sum_{j=1}^{n_k} k_j z^{-j}, F(z^{-1}) = \sum_{j=0}^{n_f} f_j z^{-j}, \tag{35}$$

is the transfer function of the feedback; $W_0(z^{-1})$ and $H(z^{-1})$ are transfer functions, defined by (9), (11); τ' is pure delay in the feedback.

In this case controller equations (19) and (21) remain unchanged, and instead of (20) we have

$$\tilde{u}_{t+1} = \left[\hat{W}_{0,t}(z^{-1}) + \alpha_t\right]^{-1} \times \left\{\tilde{y}_{t+\tau+1} + z^{\tau+1}\left[\hat{E}_t(z^{-1}) - \hat{H}_t(z^{-1})\right]\right.$$
$$\left. \times\, \varepsilon_{t|t-1}(\hat{c}_{t-1}) - \hat{W}_{0,t}(z^{-1})\hat{W}_{F,t}(z^{-1})y_{t-\tau'}\right\}. \tag{36}$$

Plants with internal feedbacks are found in power systems, in medical care systems (simulation of human cardiovascular system), etc. It is always possible, by applying adequate multiplication operations, to arrive at a linear plant, but in this case we shall have a greater number of unknown parameters to be determined. It is inefficient, especially if either the parameters of the control channel or the feedback channell are known a priori, e.g. in nuclear power reactor [KJV90, KJV92].

4 Predictor-Based Self-tuning Control of Nonlinear Plants

Nonlinear dynamic systems are considered with an observed output signal y_t defined by

$$y_t = W_2(z^{-1})f(v_t;\theta) + H(z^{-1})\xi_t, \tag{37}$$

$$v_t = W_1(z^{-1})z^{-\tau}u_t, \tag{38}$$

where

$$f(v_t;\theta) = \sum_{i=1}^{n_\theta} \theta_i v_t^i, \tag{39}$$

is a nonlinear characteristic of the static element with the parameters

$$\theta^T = (\theta_1, \theta_2, ..., \theta_{n\theta}), n_\theta \geq 2, \theta_{n\theta} \neq 0, \tag{40}$$

$$W_1(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})}, W_2(z^{-1}) = \frac{D(z^{-1})}{G(z^{-1})}, \tag{41}$$

$$D(z^{-1}) = d_0 + \sum_{j=1}^{n_d} d_j z^{-j}, G(z^{-1}) = 1 + \sum_{i=1}^{n_g} g_i z^{-i}, \tag{42}$$

which are transfer functions in the form of (9), providing a unitary gain in the control signal channel.

Equations (37), (38) specify the so called Wiener–Hammerstein-type nonlinear stochastic plants with the nonlinear element standing between two linear dynamical parts. In particular cases, by removing the first or the second linear dynamical part, we can consider, respectively, Hammerstein-type or Wiener-type stochastic plants with polynomial nonlinearities.

The optimal $(\tau + 1)$-step ahead prediction of the output signal y_t at the discrete time t is

$$y_{t+\tau+1|t}(c) = z^{\tau+1}[1 - \tilde{H}^{-1}(z^{-1})]y_t + \tilde{H}^{-1}(z^{-1})W_2(z^{-1})f(v_t; \theta). \tag{43}$$

Applying the latter equation to control performance criterion equation (2) we find that the value $\tilde{u}_{t+1}$ is obtained as a real root of $(2n_\theta - 1)$-th order equation.

The minimum generalized variance controller equation is (19), where $\tilde{u}_{t+1}$ is any real root of

$$u_{t+1}^3 + \frac{3f'(s_t)}{2\hat{b}_{0,t}\hat{\theta}_{2,t}} u_{t+1}^2 + \frac{\hat{b}_{0,t}^2 \hat{d}_{0,t}\{2\hat{\theta}_{2,t}\beta_t + \hat{d}_{0,t}[f'(s_t)]^2\} + q_t}{2\hat{b}_{0,t}^4 \hat{d}_{0,t}^2 \hat{\theta}_{2,t}^2} u_{t+1}$$
$$+ \frac{\hat{b}_{0,t}\hat{d}_{0,t} f'(s_t)\beta_t - q_t \hat{y}_t^*}{2\hat{b}_{0,t}^4 \hat{d}_{0,t}^2 \hat{\theta}_{2,t}^2} = 0, \tag{44}$$

and

$$\beta_t = \hat{d}_{0,t} f(s_t) + \tau_t - \delta_t - \hat{y}_t^*, \tag{45}$$

$$\delta_t = z^{\tau+1}[\hat{E}_t(z^{-1}) - \hat{H}_t(z^{-1})]\varepsilon_{t|t-1}(\hat{c}_{t-1}), \tag{46}$$

$$s_t = z[\hat{W}_{1,t}(z^{-1}) - \hat{b}_{0,t}]u_{t-\tau}, \tag{47}$$

$$\tau_t = z[\hat{W}_{2,t}(z^{-1}) - \hat{d}_{0,t}]f(\hat{v}_t, \hat{\theta}_t), \tag{48}$$

$$f(s_t) = \hat{\theta}_{0,t} + \hat{\theta}_{1,t} s_t^2, f'(s_t) = \hat{\theta}_{1,t} + 2\hat{\theta}_{2,t}, \tag{49}$$

$$\hat{v}_t = \hat{W}_{1,t}(z^{-1}) z^{-\tau} u_t. \tag{50}$$

In the case of $n_\theta = 2$(the extremal characteristic) the reference value y_t^* may be the current maximum point of the characteristic. In this case

$$\hat{y}_t^* = \hat{\theta}_{0,t} - \frac{\hat{\theta}_{1,t}^2}{4\hat{\theta}_{2,t}} \tag{51}$$

and it is possible to obtain simpler controller equation, for example in the case where $q_t = 0$ we get the following controller equations [Kam88, KŠT91]

$$\tilde{u}_{t+1} = \hat{W}_{1,t}^{-1}(z^{-1}) \left(-\frac{\hat{\theta}_{1,t}}{2\hat{\theta}_{2,t}} \pm \sqrt{\max(0, v_t)} \right) \tag{52}$$

$$v_t = \frac{1}{\hat{\theta}_{2,t}} [\hat{W}_{2,t}(z^{-1})(\delta_t + \tilde{y}_t) - y^*], \tag{53}$$

$$\tilde{y}_k = \begin{cases} y^* & \text{if} \quad k = t \\ y_{k+\tau+1|k} & \text{if} \quad k = t-1,\ t-2, \dots \end{cases} \tag{54}$$

The alternate sign $\pm$ in the controller equation (46) precedes the square root operation, indicating that $\tilde{u}_{t+1}$ is obtained as the solution of a quadratic equation. The sign must be selected so that the signal magnitude and deviation constraints are violated as little as possible.

Such a model may be regarded as a good approximation of fuel combustion and steam condensation processes in power units of a thermal power plant [KTŠ88, KŠT91]. These processes are distinguished by their well-expressed nonlinearity (extremal characteristic) and inertness of input and output circuits.

5 Conclusions

Design problems of predictor-based self-tuning control with constraints were discussed. Practical issues of implementation of predictor-based self-tuning control systems for different types of control plants were considered.

Control algorithm synthesis is accomplished, taking into account amplitude and/or deviation constraints on the control signals. The unknown parameters of the one step predictor of the output signal are estimated in the identification process in the optimal closed loop system.

Simulation results are given in [KJV92] to illustrate the implementation of predictor-based self-tuning control systems for different control plants. Certain operating regimes of thermal neutron and fast breeder nuclear reactors, fuel combustion and steam condensation processes in the power units of thermal power plants were considered.

References

[AW84] Aström, K.J., Wittenmark, B.: Computer Controlled Systems – Theory and Design. Prentice Hall, Englewood Cliffs (1984)

[CB95] Camacho, E.F., Bordons, C.: Model Predictive Control in the Process Industry. Springer, London (1995)

[CB98] Camacho, E.F., Bordons, C.: Model Predictive Control. Springer, Berlin (1998)

[Cla84] Clarke, D.W.: Self-tuning control of non minimum-phase systems. Automatica, **20**(5), 501 517 (1984)

[Cla94] Clarke, D.W.: Advances in Model Predictive Control. Oxford Science Publications, Oxford, UK (1994)

[CMT87] Clarke, D.W., Mohtadi, C., Tuffs, P.S.: Generalized predictive control. Automatica, **23**(2), 137–160 (1987)

[EV98] Espinosa, J., Vandewalle, J.: Predictive control using fuzzy models applied to a steam generating unit. In: Ruan, D., Abderrahim, H.A., D'hondt, P., Kerre, E. (eds) Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry, Proceedings of the Third International FLINS Workshop, Vol. 3 of Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry. World Scientific, Singapore, 151–160 (1998)

[Ise81] Isermann, R.: Digital Control Systems. Springer, Berlin (1981)

[JK98] Juang, J.N., Kenneth, W. E.: Predictive Feedback and Feedforward Control for Systems with Unknown Disturbances. National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia (1998)

[Kam82] Kaminskas, V.: Dynamic System Identification via Discrete-time Observations. Pt. I. Statistical Method Foundations. Estimation in Linear Systems. Mokslas, Vilnius (1982)

[Kam88] Kaminskas, V.: Predictor-based self-tuning control systems. In: 33 Intern. Wiss. Koll., TH Ilmenau, Heft 1, 153–156 (1988)

[KJV90] Kaminskas, V., Janickienė, D., Vitkutė, D.: Self-tuning control of the nuclear reactor power. In: Preprints of 11th World Congress "Automatic Control in the Service of Mankind", Tallin, Vol. 11, 91–96 (1990)

[KJV92] Kaminskas, V., Janickienė, D., Vitkutė, D.: Self-turning control of a power plant. In: preprints of IFAC Symp. on Control of Power Plants and Power Systems, Munich, Vol. 1, 229–234 (1992)

[KŠT91] Kaminskas, V., Šidlauskas, K., Tallat-Kelpsa, C.: Constrained self-tuning control of stochastic extremal systems. Informatica, **2**(1), 33–52 (1991)

[KTŠ88] Kaminskas, V., Tallat-Kelpsa, C, Šidlauskas, K.: Self-tuning minimum-variance control of nonlinear Wiener-Hammerstein-type systems. In: Preprints 8th IFAC/IFORS Symp. on Ident. and Syst. Par. Est., Beijing, Vol. 1, 384–389 (1988)

[ML99] Morari, M., Lee, J.H.: Model predictive control: past, present and future. Computers and Chemical Engineering, **23**, 667–682 (1999)

[MS05] Murray-Smith, R., Sbarbaro, D.: Self-tuning control of non-linear systems using Gaussian process prior models. Switching and Learning. LNCS 3355, Springer, Berlin, 140–157 (2005)

[MSRG03] Murray-Smith, R., Sbarbaro, D., Rasmussen, C.E., Girard, A.: Adaptive, Cautious, Predictive Control with Gaussian Process Priors, 13th IFAC Symposium on System Identification, IFAC, Rotterdam (2003)

[Pet84] Peterka, V.: Predictor-based self-tuning control. Automatica, **19**(5), 471 486 (1984)

[Ric93] Richalet, J.: Industrial applications of model based predictive control. Automatica, **23**(5), (1993)

A Template-Based Mixed-Integer Linear Programming Sequence Alignment Model

Scott R. McAllister, Rohit Rajgaria, and Christodoulos A. Floudas

Department of Chemical Engineering, Princeton University, Princeton, NJ 08544-5263 USA floudas@titan.princeton.edu

1 Introduction

The similarity of DNA or protein sequences between species is an approximate measure of the evolutionary distance between them. By identifying and aligning pairs of sequences, knowledge of the genetic importance of a gene or protein of one species can be extended to a second, relatively unknown species with a similar sequence.

The pairwise sequence alignment problem can be split into two categories: (a) global alignment and (b) local alignment. The goal of the global alignment algorithm is to identify the best sequence alignment that spans the entire length of both sequences. Since all base pairs from both sequences are aligned in this method, this approach is well-suited to sequence pairs of similar length and significant similarity. The pioneering work in this area was done by Needleman and Wunsch [NW70]. They used a dynamic programming approach to solve the global alignment program, relying on the principle that the score of the alignment at each intermediate step is the sum of the score of the "best" alignment up that point and the score of aligning the intermediate residue. Another dynamic programming-based algorithm tries to address the alignment problems where two sequences have an intermittent similarity regions or an order list of similar regions separated by different gaps [HC03].

Local alignment algorithms, unlike global alignment, attempt to identify only the best aligning subset of two sequences. This approach is more appropriate for sequences that may have a region of high similarity, but also some regions that are extremely dissimilar. Smith and Waterman extended the dynamic programming sequence alignment approach to address the local alignment problem [SW81].

Multiple sequence alignment is an extension of the pairwise alignment problem, where two sequences are aligned to identify the highly conserved residues. These conserved residues can then be used to derive information about common structure or function of a protein from a common evolutionary sequence. Rigorous dynamic programming can not be used for high throughput multiple sequence alignment as the complexity and space requirement is exponential in the number of aligned sequences. As a result, a number of heuristics-based algorithms are used for multiple sequence alignment. A comprehensive review can be found in Thompson et al., 1999 [TPP99].

In contrast to the recent focus on heuristic methods and large-scale or database searches, this chapter will instead address the global pairwise sequence alignment problem in a mathematically-detailed, rigorous and generic manner.

2 Methods

Two novel mixed-integer linear optimization (MILP) models were developed to address the global sequence alignment problem in a completely rigorous and generic fashion. As presented here, the sequence alignment scoring function includes the weights from a pairwise alignment scoring matrix and gap penalties based on both an affine gap penalty form and a concave gap penalty form. In addition, gaps that begin or end a sequence are not subject to the above gap penalty terms. Minor modifications to the models can address different forms of the scoring function, including cases of no gap penalties and all gaps weighted equally.

These optimization models are not only guaranteed to find the global maximum alignment for a scoring function, but can also use a series of integer cut constraints to identify a rank-ordered list of the sequence alignments with the highest scores. As Sect. 4 will show, the rank-ordered list is an invaluable tool for a detailed analysis of a pairwise sequence alignment.

3 Model

Consider two sequences $S1, S2$ of length M and N respectively, where $M \geq N$. Let the index i represent each position in sequence S1 and the index j represent each position in S2, as shown in (1-2).

$$i \in 1, 2 \ldots M \tag{1}$$

$$j \in 1, 2 \ldots N \tag{2}$$

The optimization model assigns each amino acid of both sequences to a template to generate the optimal alignment. Equation (3) defines the template length K as the sum of the length of the larger sequence and the parameter N_GAPS_m, representing the maximum number of allowed gaps. This also leads to the introduction of an index k to represent the position in the template, as defined by (4).

$$K = M + N\_GAPS_m \tag{3}$$

$$k \in 1, 2 \ldots K \tag{4}$$

The binary variables of (5-6), y_{ik} and z_{jk}, denote the assignment of an amino acid to a template position.

$$y_{ik} = \begin{cases} 1 \text{ if amino acid } i \text{ of S1 is assigned to template position } k \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

$$z_{jk} = \begin{cases} 1 \text{ if amino acid } j \text{ of S2 is assigned to template position } k \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

In most sequence alignment problems, not every position in the template is required to have an amino acid from both sequences in each position of the template. Therefore, additional binary variables must be introduced to represent the gaps. These variables, yg_k and zg_k, are defined in (7-8).

$$yg_k = \begin{cases} 1 \text{ if template position } k \text{ is a gap for sequence S1} \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

$$zg_k = \begin{cases} 1 \text{ if template position } k \text{ is a gap for sequence S2} \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

The objective function of this optimization model maximizes the alignment score, which is the sum of a scoring matrix value for each match minus any associated penalties for gaps inserted in the sequence. The scoring matrix will assign a weight, w_{ij}, to any template position k that contains the amino acid in position i of sequence S1 and also the amino acid in position j of S2.

The choice of the gap penalty form is a compromise between the simplicity of the mathematical formulation and the biological meaning. There are three main forms of gap penalties. The simplest form is the constant gap penalty, where all gaps are weighted equally. The second form is the affine gap penalty, which combines a gap opening penalty with a linear extension penalty term. The third and most complex of gap penalty terms is the concave gap penalty. Despite its complexity, it is sometimes used due to the more realistic representation of biology that it incorporates. The model presented in Sect. 3.1 implements the affine gap penalty form, although it can be readily transformed to a constant gap penalty term with only minor modifications. The concave gap penalty formulation will be presented separately in Sect. 3.2. If unweighted end gaps are not appropriate for a given application, they can easily be replaced by weighted values by removing only a few terms from the model.

3.1 Affine Gap Penalty Model

The basic model presented here will assume an affine gap penalty, wg, of the form of (9), where n_g is the length of the gap, wo is the gap opening penalty and wl is the gap extension penalty.

$$wg = wo + (n_g - 1) \cdot wl \tag{9}$$

To properly address this form of a weighted gap penalty, the following variables will be needed.
go_k^{S1} = 1 for the case of A - in $S1$ at position $k-1, k$
go_k^{S2} = 1 for the case of A - in $S2$ at position $k-1, k$
gl_k^{S1} = 1 for the case of - - in $S1$ at position $k-1, k$
gl_k^{S2} = 1 for the case of - - in $S2$ at position $k-1, k$
gb_k^{S1} = takes the value of 1 if $S1$ begins with a gap up to position k
gb_k^{S2} = takes the value of 1 if $S2$ begins with a gap up to position k
ge_k^{S1} = takes the value of 1 if $S1$ ends with a gap that extends back to position k
ge_k^{S2} = takes the value of 1 if $S2$ ends with a gap that extends back to position k

A concrete example of these variable values is presented in Fig. 1. This hypothetical sequence alignment contains a gap that begins a sequence, a gap that ends a sequence, and a gap in the middle of a sequences. First, let us address the gap opening values. As defined, gaps are opened in sequence 1 at position 1 and in sequence 2 at positions 7 and 12. This results in the variables $go_1^{S1}, go_7^{S2}, go_{12}^{S2}$ being activated. Likewise, gaps are extended in sequence 1 at position 2 and sequence 2 at position 8 and 9, leading to the activation of the variables $gl_2^{S1}, gl_8^{S2}, gl_9^{S2}$. Since both positions 1 and 2 of sequence 1 are gaps that begin a sequence, the variables gb_1^{S1}, gb_2^{S1} take a value of 1. Sequence 2 ends with a single gap residue at position 12, which can be represented by activating the variable ge_{12}^{S2}.

```
Position     123456789012
Sequence 1:  --YWGSACERLT
Sequence 2:  LAYYGT---RI-
```

Fig. 1. A hypothetical sequence alignment to illustrate variable definitions

Given the appropriate selection of a gap penalty representation, the objective function can then be posed as shown in (10). Here the value of the scoring matrix at positions i, j, w_{ij}, is counted only if position i of sequence 1 is assigned to position k of the template, y_{ik}, and if position j of sequence 2 is assigned to position k of the template, z_{jk}. The gap opening penalty value of wo is multiplied by the gap opening existence terms of sequence 1, go_k^{S1}, and sequence 2, go_k^{S2}, to assess the penalty for the first residue of any gap in a sequence. Likewise, the existence of a gap extension variable of either sequence, gl_k^{S1} and gl_k^{S2}, produces a penalty of wl for each occurrence. The active gl_k^{S1} and gl_k^{S2} variables that are contained within a beginning or an ending gap are counteracted by the product of gb_k^{S1},gb_k^{S2},ge_k^{S1}, or ge_k^{S2} with wl. Gap opening penalties at the beginning or end of a sequence are explicitly omitted through only summing over the reduced index, such that $2 \leq k \leq K-1$.

$$\max \sum_i \sum_j \sum_k w_{ij} \cdot y_{ik} \cdot z_{jk} - \sum_{k=2}^{K-1} (go_k^{S1} + go_k^{S2}) \cdot wo$$
$$- \sum_{k=2}^{K-1} \left[(gl_k^{S1} - gb_k^{S1} - ge_k^{S1}) + (gl_k^{S2} - gb_k^{S2} - ge_k^{S2}) \right] \cdot wl \qquad (10)$$

However, the product of y_{ik} and z_{jk} results in a nonlinear objective function. This issue can be addressed by the definition of an additional variable, ϕ_{ijk} as shown in (11).

$$\phi_{ijk} = y_{ik} \cdot z_{jk} \qquad (11)$$

Alternatively, this product of binary variables can be transformed into an equivalent set of linear constraints as shown in (12-13) [Flo95].

$$y_{ik} + z_{jk} - 1 \leq \phi_{ijk} \leq y_{ik} \qquad (12)$$
$$0 \leq \phi_{ijk} \leq z_{jk} \qquad (13)$$

Therefore, the linearized objective function for this model formulation is illustrated by (14).

$$\max \sum_i \sum_j \sum_k w_{ij} \cdot \phi_{ijk} - \sum_{k=2}^{K-1} (go_k^{S1} + go_k^{S2}) \cdot wo$$
$$- \sum_{k=2}^{K-1} \left[(gl_k^{S1} - gb_k^{S1} - ge_k^{S1}) + (gl_k^{S2} - gb_k^{S2} - ge_k^{S2}) \right] \cdot wl \qquad (14)$$

A number of constraints must be included in the model to properly address the sequence alignment problem. By definition, an amino acid in position i must be assigned to only a single template position k. Likewise, a position j must correspond to only one template position k. These restrictions are imposed by (15-16).

$$\sum_k y_{ik} = 1 \quad \forall i \qquad (15)$$

$$\sum_k z_{jk} = 1 \quad \forall j \qquad (16)$$

In a similar fashion, only one amino acid position can occupy a position in the template, k. Equations (17-18) enforce this constraint.

$$\sum_k y_{ik} \leq 1 \quad \forall k \qquad (17)$$

$$\sum_k z_{jk} \leq 1 \quad \forall k \qquad (18)$$

Equations (19-20) relate the variables that specify the existence of gaps at a position in the template, yg_k and zg_k, to the activity of the binary variables representing the assignment of an amino acid sequence position to a template position.

$$yg_k = 1 - \sum_i y_{ik} \quad \forall k \tag{19}$$

$$zg_k = 1 - \sum_j z_{jk} \quad \forall k \tag{20}$$

The constraints presented so far yield an alignment, but they do not preserve the ordering necessary to prevent the accidental scrambling of either sequence when the amino acids are assigned to a template position. This issue can be addressed by (21-22).

$$y_{ik} + y_{i+1,k'} \leq 1 \quad \forall i, k, k' \leq k \tag{21}$$

$$z_{jk} + z_{j+1,k'} \leq 1 \quad \forall j, k, k' \leq k \tag{22}$$

Several variables that make up part of the gap penalty term in the objective function need to be related to the assigned positions in the template. Equations (23-24) force the term go_k^{S1} to the correct value, as previously defined. Due to the exclusive relationship between gap positions that open a gap and those that extend a gap, (25) relates the gap opening variable, go_k^{S1}, the gap extension variable, gl_k^{S1}, and the binary variable representing the existence of a gap, yg_k.

$$yg_{k+1} - yg_k \leq go_{k+1}^{S1} \leq 1 - yg_k \quad \forall k < K \tag{23}$$

$$go_k^{S1} \leq yg_k \quad \forall k \tag{24}$$

$$gl_k^{S1} = yg_k - go_k^{S1} \quad \forall k \tag{25}$$

Consider the subsequence illustrated in Fig. 2. At position 2 of the subsequence, (26) defines the gap existence binary variables and (27-29) hold true based upon the constraints of (23-25). Here it is important to note that (23-25) only need to account for the current position and a single previous position to determine whether a residue opens a gap, extends a gap, or aligns to a residue in another sequence.

$$yg_1, yg_2 = 0 \tag{26}$$

$$0 \leq go_2^{S1} \leq 1 \tag{27}$$

$$go_2^{S1} \leq 0 \tag{28}$$

$$gl_2^{S1} = 0 \tag{29}$$

```
AA--A
```

Fig. 2. An example subsequence illustrating the validity of the gap variable existence constraints

At position 3 of the sequence in Fig. 2, the gap existence variables are in (30), and (23-25) become (31-33).

$$yg_2 = 0, yg_3 = 1 \tag{30}$$

$$\begin{align} 1 \leq go_3^{S1} &\leq 1 \tag{31}\\ go_3^{S1} &\leq 1 \tag{32}\\ gl_3^{S1} &= 0 \tag{33}\end{align}$$

At position 4 of the sequence in Fig. 2, the gap existence variables are in (34), and (23-25) turn into (35-37).

$$yg_3 = 1, yg_4 = 1 \tag{34}$$

$$\begin{align} 0 \leq go_4^{S1} &\leq 0 \tag{35}\\ go_4^{S1} &\leq 1 \tag{36}\\ gl_4^{S1} &= 1 \tag{37}\end{align}$$

Finally, at position 5 of the sequence in Fig. 2, the gap existence variables are in (38), and (23-25) become (39-41).

$$yg_4 = 1, yg_5 = 0 \tag{38}$$

$$\begin{align} -1 \leq go_5^{S1} &\leq 0 \tag{39}\\ go_5^{S1} &\leq 0 \tag{40}\\ gl_5^{S1} &= 0 \tag{41}\end{align}$$

A similar set of constraints, illustrated in (42-44), enforce the gap penalty values for the variables representing sequence 2.

$$zg_{k+1} - zg_k \leq go_{k+1}^{S2} \leq 1 - zg_k \quad \forall k < K \tag{42}$$

$$go_k^{S2} \leq zg_k \quad \forall k \tag{43}$$

$$gl_k^{S2} = zg_k - go_k^{S2} \quad \forall k \tag{44}$$

In this model formulation, a position can be a part of a gap that begins the aligned sequence when both (i) the current position is a gap and (ii) the previous position was a gap. This can be properly defined as shown in (45-48).

$$\begin{align} gb_k^{S1} &\leq \ yg_k && \forall k \tag{45}\\ gb_k^{S1} &\leq gb_{k-1}^{S1} && \forall k > 1 \tag{46}\\ gb_k^{S2} &\leq \ zg_k && \forall k \tag{47}\\ gb_k^{S2} &\leq gb_{k-1}^{S2} && \forall k > 1 \tag{48}\end{align}$$

Figure 3 is another example subsequence that will be used to illustrate the constraints necessary to assign the proper values to the variables representing gaps at the beginning and end of a sequence. At position 1 of the subsequence, (49) defines the gap existence binary variable and (50) holds true based upon the constraint in (45). Although only an upper bound is imposed on the variable gb_1^{S1}, the form of the objective function of (14), will cause the gap beginning variable to strictly assume its upper bounding value.

$$yg_1 = 1 \tag{49}$$

$$gb_1^{S1} \leq 1 \tag{50}$$

```
--A-A
```

Fig. 3. An example subsequence illustrating the validity of the gap variable existence constraints

At position 2 of the subsequence in Fig. 3, (51) defines the existing variables, and (52-53) are enforced based on the constraints in (45-46).

$$yg_2 = 1, gb_1^{S1} = 1 \tag{51}$$

$$gb_2^{S1} \leq 1 \tag{52}$$
$$gb_2^{S1} \leq 1 \tag{53}$$

At position 3 of the subsequence in Fig. 3, (54) defines the existing variables, and (55-56) are enforced based on the constraints in (45-46).

$$yg_3 = 1, gb_2^{S1} = 1 \tag{54}$$

$$gb_3^{S1} \leq 0 \tag{55}$$
$$gb_3^{S1} \leq 1 \tag{56}$$

Due to the cascading nature of the constraint in (46), once gb_k^{S1} is forced to 0, all $gb_{k'}^{S1}$ $(k' > k)$ are set to zero as well. Thus, since position 3 is not part of the gap that begins the sequence of Fig. 3, no subsequent positions are allowed to be part of this gap and avoid the typical gap penalty term.

Equations (57-60) are a similar set of constraints that identify the positions of gaps that end the sequence.

$$ge_k^{S1} \leq yg_k \quad \forall k \tag{57}$$
$$ge_k^{S1} \leq ge_{k+1}^{S1} \quad \forall k < K \tag{58}$$
$$ge_k^{S2} \leq zg_k \quad \forall k \tag{59}$$
$$ge_k^{S2} \leq ge_{k+1}^{S2} \quad \forall k < K \tag{60}$$

Equations (12-25), (42-48), (57-60) represent the sequence alignment model, a mixed-integer linear optimization (MILP) problem, which can be solved to optimality using commercial solvers (e.g. CPLEX [Ilo03], Xpress-MP [Das03]).

3.2 Concave Gap Penalty Model

The alternative model presented here requires the introduction of a new set, P, in (61) that represents the positions in the concave penalty function. For this second model formulation, we will assume a concave gap penalty which assumes the form of (62) where n_g is the length of the gap, wo is the gap opening penalty and wl is a weight factor for the concave gap penalty. Alternatively, the concave penalty function can be represented as a sum of predefined penalties for each position in the gap. This representation is illustrated by wc_p in (63) and will be used to simplify the constraints needed for the model.

$$p \in 1, 2 \ldots P \tag{61}$$

$$wg = wo + \sum_{p=2}^{n_g} wl \cdot \log(p) \cdot \tag{62}$$

$$wg = \sum_{p}^{n_g} wc_p \tag{63}$$

Given this new index, p, and the stated definition of the gap penalty form, the following variables will be needed.
gc_k^{S1}, p = 1 for the case of a gap of score wc_p in $S1$ at position k and gap penalty position p
gc_k^{S2}, p = 1 for the case of a gap of score wc_p in $S2$ at position k and gap penalty position p
gb_k^{S1} = takes the value of 1 if $S1$ begins with a gap up to position k
gb_k^{S2} = takes the value of 1 if $S2$ begins with a gap up to position k
ge_k^{S1} = takes the value of 1 if $S1$ ends with a gap that extends back to position k
ge_k^{S2} = takes the value of 1 if $S2$ ends with a gap that extends back to position k

Given the appropriate selection of a gap penalty representation, the objective function can then be posed as shown in (64). The value of the scoring matrix at positions i, j, w_{ij}, is added only if position i of sequence 1 is assigned to position k of the template, y_{ik}, and if position j of sequence 2 is assigned to position k of the template, z_{jk}. The position dependent gap penalty variables, $gc_{c,p}^{S1}$ and $gc_{c,p}^{S2}$, are multiplied by the position dependent wc_p parameter to calculate the overall gap penalty. The active gc_k^{S1} and gc_k^{S2} variables that are contained within a beginning or an ending gap are then counteracted by the product of gb_k^{S1},gb_k^{S2},ge_k^{S1}, or ge_k^{S2} with the position dependent weight wc_p. The correct positions are established for the end gaps by counting an increasing positional dependence for gaps that begin a sequence and a decreasing positional dependence for gaps that end a sequence.

$$\begin{aligned}\max \textstyle\sum_i \sum_j \sum_k w_{ij} \cdot y_{ik} \cdot z_{jk} \\ - \textstyle\sum_p^P \sum_k^K \left[gc_{k,p}^{S1} + gc_{k,p}^{S2}\right] \cdot wc_p \\ + \textstyle\sum_k^K \left[(gb_k^{S1} + gb_k^{S2}) \cdot wc_{p=k} + (ge_k^{S1} + ge_k^{S2}) \cdot wc_{p=K+1-k})\right]\end{aligned} \tag{64}$$

The product of y_{ik} and z_{jk} results in a nonlinear objective function, which can be resolved by the definition of an additional variable, ϕ_{ijk} as shown in (65) and again transformed into the equivalent linear constraints of (66-67) [Flo95].

$$\phi_{ijk} = y_{ik} \cdot z_{jk} \tag{65}$$

$$y_{ik} + z_{jk} - 1 \leq \phi_{ijk} \leq y_{ik} \tag{66}$$

$$0 \leq \phi_{ijk} \leq z_{jk} \tag{67}$$

The linearized objective function for this model formulation is rewritten as (68).

$$\begin{aligned}\max \textstyle\sum_i \sum_j \sum_k w_{ij} \cdot \phi_{ijk} \\ - \textstyle\sum_p^P \sum_k^K \left[gc_{k,p}^{S1} + gc_{k,p}^{S2}\right] \cdot wc_p \\ + \textstyle\sum_k^K \left[(gb_k^{S1} + gb_k^{S2}) \cdot wc_{p=k} + (ge_k^{S1} + ge_k^{S2}) \cdot wc_{p=K+1-k})\right]\end{aligned} \tag{68}$$

Equations (69-76) must again be included to assign only one amino acid to a template position, to assign only one template position to an amino acid position, to relate the gap binary variables to the template assignment binary variables, and to preserve the original ordering of the amino acid sequences.

$$\sum_k y_{ik} = 1 \quad \forall i \tag{69}$$

$$\sum_k z_{jk} = 1 \quad \forall j \tag{70}$$

$$\sum_k y_{ik} \leq 1 \quad \forall k \tag{71}$$

$$\sum_k z_{jk} \leq 1 \quad \forall k \tag{72}$$

$$yg_k = 1 - \sum_i y_{ik} \quad \forall k \tag{73}$$

$$zg_k = 1 - \sum_j z_{jk} \quad \forall k \tag{74}$$

$$y_{ik} + y_{i+1,k'} \leq 1 \quad \forall i, k, k' \leq k \tag{75}$$

$$z_{jk} + z_{j+1,k'} \leq 1 \quad \forall j, k, k' \leq k \tag{76}$$

Once these other constraints have been introduced, the variables that make up part of the gap penalty term in the objective function need to be related to the assigned positions in the template. Equations (77-78) force the first gap penalty term $gc_{k,1}^{S1}$ to the correct value, in a similar fashion to the model of Sect. 3.1. After the first term has the correct value, the subsequent concave gap penalty terms can be defined in relation to the term of the previous template position, as shown in (79-80). Equations (78) and (80) can be combined into a single equation, as the separation presented here is only for pedagogical purposes.

$$yg_{k+1} - yg_k \le gc_{k+1,1}^{S1} \le 1 - yg_k \quad \forall k < K \tag{77}$$
$$gc_{k,1}^{S1} \le yg_k \quad \forall k \tag{78}$$
$$yg_{k+1} + gc_{k,p}^{S1} \le gc_{k+1,p+1}^{S1} \le gc_{k,p}^{S1} \quad \forall k < K, 1 < p < P \tag{79}$$
$$gc_{k,p}^{S1} \le yg_k \quad \forall k, p > 1 \tag{80}$$

A similar set of constraints, illustrated in (81-83), enforce the gap penalty values for the variables representing sequence 2.

$$zg_{k+1} - zg_k \le gc_{k+1,1}^{S2} \le 1 - zg_k \quad \forall k < K \tag{81}$$
$$zg_{k+1} + gc_{k,p}^{S2} \le gc_{k+1,p+1}^{S2} \le gc_{k,p}^{S2} \quad \forall k < K, 1 < p < P \tag{82}$$
$$gc_{k,p}^{S2} \le zg_k \quad \forall k, p \tag{83}$$

The representation of the beginning and end gap variables, used to identify and properly weight gaps that start or end a sequence, are constrained in the same fashion as the model of Sect. 3.1. Equations (84-91) are then needed to complete this model.

$$gb_k^{S1} \le yg_k \quad \forall k \tag{84}$$
$$gb_k^{S1} \le gb_{k-1}^{S1} \quad \forall k > 1 \tag{85}$$
$$gb_k^{S2} \le zg_k \quad \forall k \tag{86}$$
$$gb_k^{S2} \le gb_{k-1}^{S2} \quad \forall k > 1 \tag{87}$$
$$ge_k^{S1} \le yg_k \quad \forall k \tag{88}$$
$$ge_k^{S1} \le ge_{k+1}^{S1} \quad \forall k < K \tag{89}$$
$$ge_k^{S2} \le zg_k \quad \forall k \tag{90}$$
$$ge_k^{S2} \le ge_{k+1}^{S2} \quad \forall k < K \tag{91}$$

Equations (66-91) represent the sequence alignment model for a concave gap representation, a mixed-integer linear optimization (MILP) problem, which can also be solved to optimality using commercial solvers (e.g. CPLEX [Ilo03], Xpress-MP [Das03]).

4 Results

A number of example problems have been addressed to further clarify the variables and constraints that make up the model, as well as to demonstrate the utility of this approach. The first example addresses a pair of hypothetical protein sequences, designed to illustrate the variable representation of the alignment model. The goal of the second example is to align the region of the first helix of the human dopamine G-protein coupled receptor to the first helix of bovine rhodopsin to illustrate the utility of producing a rank-ordered set of alignments through integer cuts. Finally, the third example shows the ease of augmenting the model to account for functionally significant alignments by considering the structure of the bovine pancreatic trypsin inhibitor.

4.1 Hypothetical Sequence

Consider the alignment of the sequences presented in Fig. 4, which will be named sequence A and sequence B for convenience.

Sequence A: `NDYIGKAYTSRK`
Sequence B: `LNEWIGFTMRR`

Fig. 4. Hypothetical sequences input to the global pairwise sequence alignment algorithm

If we allow for a template length of 15, the optimal alignment of the two sequences using the model of Sect. 3.1 is shown in Table 1. For this alignment, the gap opening penalty is assigned a value of 5, the gap extension penalty is 0.5, and the BLOSUM62 amino acid substitution matrix [HH92] is used to score the amino acid pairs.

Table 1. Hypothetical sequence alignment representation

Template position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sequence A	-	N	D	Y	I	G	K	A	Y	T	S	R	K	-	-
Sequence B	L	N	E	W	I	G	-	-	F	T	M	R	R	-	-
Score	0	6	2	2	4	6	-5	-0.5	3	5	-1	5	2	0	0

In the models of Sect. 3, the binary variable y_{ik} represents the assignment of an amino acid at position i to the template position k. Similarly, z_{jk} is active when an amino acid at position j is placed in the template position k. The active values for this example can be seen in (92-93).

$$y_{1,2}, y_{2,3}, y_{3,4}, y_{4,5}, y_{5,6}, y_{6,7}, y_{7,8}, y_{8,9}, y_{9,10}, y_{10,11}, y_{11,12}, y_{12,13} = 1 \quad (92)$$
$$z_{1,1}, z_{2,2}, z_{3,3}, z_{4,4}, z_{5,5}, z_{6,6}, z_{7,9}, z_{8,10}, z_{9,11}, z_{10,12}, z_{11,13} = 1 \quad (93)$$

At template positions $k = 7, 8, 14, 15$, there are gaps in sequence B, represented by the binary variable zg_k. Likewise, the alignment of sequence A requires an active value for the variable yg_k at template positions $k = 1, 14, 15$ for the gaps that appear there. For clarity, these active values are enumerated in (94-95).

$$yg_1, yg_{14}, yg_{15} = 1 \quad (94)$$
$$zg_7, zg_8, zg_{14}, zg_{15} = 1 \quad (95)$$

For the affine gap penalty model, we introduce a gap opening penalty wo and a gap extension penalty wl, but do not penalize gaps that begin or end a sequence. Thus, because these gaps are not penalized, only one gap opening variable is activated, as shown in (96). Equation (97) shows the activated gap extension variables, although those that are present in beginning or ending gaps are compensated by the active variables in (98).

$$go_7^{S2} = 1 \quad (96)$$
$$gl_{14}^{S1}, gl_8^{S2}, gl_{14}^{S2} = 1 \quad (97)$$
$$ge_{14}^{S1}, ge_{14}^{S2} = 1 \quad (98)$$

If instead the concave gap penalty model was applied to this example, we must introduce the series of gap penalty terms wc_p, but not penalize gaps that begin or end a sequence. The active gap penalty binary variables are shown in (99-100). Several of these active values are compensated by the gap beginning or ending variables of (101-102).

$$gc_{1,1}^{S1}, gc_{14,1}^{S1}, gc_{15,2}^{S1} = 1 \quad (99)$$
$$gc_{7,1}^{S2}, gc_{8,2}^{S2}, gc_{14,1}^{S2}, gc_{15,2}^{S2} = 1 \quad (100)$$
$$gb_1^{S1}, ge_{14}^{S1}, ge_{15}^{S1} = 1 \quad (101)$$
$$ge_{14}^{S2}, ge_{15}^{S2} = 1 \quad (102)$$

4.2 Bovine Rhodopsin

One interesting problem is the prediction of the structure of membrane proteins, especially G-protein coupled receptors. Since only one high resolution G-protein coupled receptor (bovine rhodopsin) has been crystallized [PKH$^+$00], a first step for the prediction of other G-protein coupled receptors involves the alignment of helical residues onto the known structure [LSN92, DST$^+$97].

This problem is nontrivial due to the low sequence similarity shared between these protein sequences [PSS92]. This example will focus on the alignment of the first helix of the human dopamine G-protein coupled receptor (DRD2) to the first helix of bovine rhodopsin. A few residues beyond the helical regions are included, slightly extending both sequences.

The longer sequence is from the human dopamine receptor, a 41 amino acid peptide sequence, which is aligned to a shorter 37 amino acid peptide sequence from bovine rhodopsin. The top 5 alignments between these subsequences, along with their associated alignment scores, are presented in Fig. 5.

In this case, instead of analyzing a single alignment, it may be more insightful to look at a series of high scoring alignments. The only known key conserved residue is the asparagine (N) residues at position 25 of the selected bovine rhodopsin sequence region. The 10 exact alignment matches present in the first four solutions remain constant. However, the fifth solution contains one differing exact match, between proline residues instead of valines. Considering the important role of proline in G protein coupled receptor helices, this fifth solution may turn out to be a more meaningful alignment. The usefulness of the integer cuts is especially important considering the variability that can arise between different substitution matrices and a range of possibilities for modeling gap regions.

4.3 Bovine Pancreatic Trypsin Inhibitor

The structure of bovine pancreatic trypsin inhibitor (BPTI), which has been determined experimentally [DS75, WWHS84], is unique due to the three stabilizing disulfide bonds between cysteine residues. This 58 amino acid globular protein functions as an inhibitor of serine protease proteins such as trypsin and chymotrypsin. A characteristic of the pancreatic trypsin inhibitor family, which includes BPTI, is the highly conserved cysteine residues that form these disulfide bonds. Therefore, if the purpose of the sequence alignment is to match the sequences in a functionally significant fashion, instead of just blindly searching for the highest scoring alignment, this additional information should be introduced to the sequence alignment model.

Consider the alignment of the bombyx mori (domestic silkwork) kazal-type serine proteinase inhibitor 1, a 76 amino acid protein, to BPTI. The longer sequence, the inhibitor from bombyx mori, is assigned to sequence 1 and BPTI will be sequence 2. In a mathematically rigorous fashion, we can incorporate the functionally important requirement of conserved cysteine residues, as shown in (103).

$$\sum_{i \in \mathrm{Cys}} \sum_{k} \phi_{ijk} = 1 \quad \forall j \in \text{BPTI disulfide Cys} \tag{103}$$

The sequence alignment model with an affine gap penalty representation, presented in Sect. 3.1, produces the optimal alignment given the conservation

```
Sequence 1:
1234567890 1234567890 1234567890 1234567890 1
DGKADRPHYN YYATLLTLLI AVIVFGNVLV CMAVSREKAL Q
Sequence 2:
1234567890 1234567890 1234567890 1234567
LAEPWQFSML AAYMFLLIML GFPINFLTLY VTVQHKK

Gap opening penalty:           5
Gap extension penalty:       0.5
Template length:              43
Scoring Matrix:         BLOSUM62
------------------------------------------------------
ITERATION: 1          OBJECTIVE:    24 (10 matches)
     1234567890 1234567890 1234567890 1234567890 123
S1:  DGKADRPHYN YYATLLTLLI AVIVFG-NVL VC-MAVSREK ALQ
        |         |    |||     |  | |      |   |
S2:  --LAEPWQFS MLAAYMFLLI -MLGFPINFL TLYVTVQHKK ---
------------------------------------------------------
ITERATION: 2          OBJECTIVE:    24 (10 matches)
     1234567890 1234567890 1234567890 1234567890 123
S1:  DGKADRPHYN YYATLLTLLI AVIVFG-NVL -VCMAVSREK ALQ
        |         |    |||     |  | |      |   |
S2:  --LAEPWQFS MLAAYMFLLI -MLGFPINFL TLYVTVQHKK ---
------------------------------------------------------
ITERATION: 3          OBJECTIVE:    23 (10 matches)
     1234567890 1234567890 1234567890 1234567890 123
S1:  DGKADRPHYN YYATLLTLLI AVIVFG-NVL V-CMAVSREK ALQ
        |         |    |||     |  | |      |   |
S2:  --LAEPWQFS MLAAYMFLLI -MLGFPINFL TLYVTVQHKK ---
------------------------------------------------------
ITERATION: 4          OBJECTIVE:    22 (10 matches)
     1234567890 1234567890 1234567890 1234567890 123
S1:  DGKADRPHYN YYATLLTLLI AVIVFG-NVL -VCMAVSREK ALQ
        |         |    |||     |  | |      |   |
S2:  --LAEPWQFS MLAAYMFLLI M-LGFPINFL TLYVTVQHKK ---
------------------------------------------------------
ITERATION: 5          OBJECTIVE:    22 (10 matches)
     1234567890 1234567890 1234567890 1234567890 123
S1:  DGKADRP-HY NYYATLLTLL IAVIVFG-NV LVCMAVSREK ALQ
        |  |       |    || |    |  |  |        |
S2:  --LAE-PWQF SMLAAYMFLL I-MLGFPINF LTLYVTVQHK K--
```

Fig. 5. Rank-ordered list of the top five optimal alignments of the helix 1 region of the human dopamine receptor (Sequence 1) to the helix 1 region of the bovine rhodopsin (Sequence 2)

of the cysteine residues necessary for the disulfide bridge formation. This alignment is presented in Fig. 6.

```
Sequence 1:
1234567890 1234567890 1234567890 1234567890 1234567890
MLKYTSISFL LIILLFSFTN ANPDCLLPIK TGPCKGSFPR YAYDSSEDKC
VEFIYGGCQA NANNFETIEE CEAACL

Sequence 2:
1234567890 1234567890 1234567890 1234567890 1234567890
RPDFCLEPPY TGPCKARIIR YFYNAKAGLC QTFVYGGCRA KRNNFKSAED
CMRTCGGA

Gap opening penalty:        5
Gap extension penalty:    0.5
Template length:           80
Scoring Matrix:      BLOSUM62

OBJECTIVE:      151 (26 exact matches)
     1234567890 1234567890 1234567890 1234567890 1234567890
S1:  MLKYTSISFL LIILLFSFTN ANPD-CLLPI KTGPCKGSFP RYAYDSSEDK
                             || || |    |||||     || |
S2:  ---------- ---------- -RPDFCLEPP YTGPCKARII RYFYNAKAGL

S1:  CVEFIYGGCQ ANANNFETIE ECEAACL--
     |  | ||||  |  |||   |  |   |
S2:  CQTFVYGGCR AKRNNFKSAE DCMRTCGGA
```

Fig. 6. Optimal alignment of bombyx mori kazal-type serine proteinase inhibitor 1 (Sequence 1) to bovine pancreatic trypsin inhibitor (Sequence 2), given the requirement of cysteine conservation

5 Discussion

As Sect. 4 began to illustrate, the models presented in Sect. 3 have several useful characteristics that distinguish it from other methodologies. The first key feature of the models is the ability to perform any specified number of integer cuts, producing a rank-ordered list of the top N solutions to the alignment problem. A biologist approaching a global pairwise sequence alignment problem has a number of choices to make, including the algorithm, the appropriate scoring matrix, and form of the gap penalty. Although the first choice may remain fixed, a considerable amount of uncertainty can be introduced into the problem as even the best available scoring matrix and gap penalty form may not be able to distinguish between different sequence alignment possibilities.

In these cases, a biologist can use the proposed algorithm to look a number of the highest-scoring alignments and assess the biological importance of each of them individually and as a group. Most other methods are limited in this respect. One dynamic programming formulation has been proposed that can identify the paths of alternative alignments that do not include the same sequence matches from the original alignment [WE87].

Although in theory the pairwise sequence alignment problem only considers the amino acid sequences of the two proteins, often more information is available. In some cases, functionally significant residues must be conserved to retain a structural feature that may stabilize the protein or a necessary binding site property. A mathematical model that can rigorously include this type of information is the most straightforward way to address this extra information. While including these constraints, the ability to still guarantee the global optimum pairwise alignment distinguishes the proposed method of Sect. 3 from other algorithms.

6 Conclusions

Two mixed-integer linear programming (MILP) models have been developed to address the global pairwise sequence alignment problem in the most generic fashion. These models address the two most difficult forms of gap penalty terms, specifically the affine gap penalty model of Sect. 3.1 and the concave gap penalty model of Sect. 3.2. Additional protein-specific functional requirements can easily be incorporated into these models in a mathematically-rigorous fashion and the introduction of integer cuts yields the ability to produce a rank-ordered list of the highest scoring pairwise sequence alignments.

The models presented here in Sect. 3 are based on the premise of assigning each sequence to a template to carry out the sequence alignment. It is possible to formulate a more advanced mixed-integer linear programming model in a "template-free" manner, such that the sequences are aligned to each other without the use of a template. This second type of mathematical formulation will appear in a future publication.

Acknowledgement. CAF gratefully acknowledges financial support from both the National Science Foundation and the National Institutes of Health (R01 GM52032; R24 GM069736).

References

[Das03] Dash Optimization: Xpress-MP: Getting Started (2003)

[DS75] Deisenhofer, J., Steigemann, W.: Crystallographic refinement of structure of bovine pancreatic trypsin-inhibitor at 1.5 Å resolution. Acta. Crystallogr. Sect. B, **31**, 238–250 (1975)

[DST$^+$97] Du, P., Salon, J.A., Tamm, J.A., Hou, C., Cui, W., Walker, M.W., Adham, N., Dhanoa, D.S., Islam, I., Vaysse, P.J., Dowling, B., Shifman, Y., Boyle, N., Rueger, H., Schmidlin, T., Yamaguchi, Y., Branchek, T.A., Weinshank, R.L, Gluchowski, C.: Modeling the G protein-coupled neuropeptide Y Y1 receptor agonist and agonist binding sites. Protein Eng., **10**, 109–117 (1997)

[Flo95] Floudas, C.A.: Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press (1995)

[HC03] Huang, X., Chao, K.-M.: A generalized global alignment algorithm. Bioinf., **19**, 228–233 (2003)

[HH92] Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci., **89**, 10915–10919 (1992)

[Ilo03] ILOG: CPLEX User's Manual 9.0. (2003)

[LSN92] Livingstone, C.D., Strange, P.G., Naylor, L.H.: Molecular modelling of D2-like dopamine receptors. Biochem. J., **287**, 277–282 (1992)

[NW70] Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol., **48**, 443–453 (1970)

[PKH$^+$00] Palczewski, K., Kumasaka, T., Hori, T., Behnke, C.A., Motoshima, H., Fox, B.A., Le Trong, I., Teller, D.C., Okada, T., Stenkamp, R.E., Yamamoto, M., Miyano, M.: Crystal structure of rhodopsin: A G protein-coupled receptor. Science, **289**, 739–745 (2000)

[PSS92] Probst, W.C., Snyder, L.A., Schuster, D.I.: Sequence alignment of the G protein-coupled receptor superfamily. DNA Cell Biol., **11**, 1–20 (1992)

[SW81] Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol., **147**, 195–197 (1981)

[TPP99] Thompson, J.D., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. Nuc. Acids Res., **27**, 2682–2690 (1999)

[WE87] Waterman, M.S., Eggert, M.: A new algorithm for best subsequence alignments with application to tRNA-tRNA comparisons. J. Mol. Biol., **197**, 723–728 (1987)

[WWHS84] Wlodawer, A., Walter, J., Huber, R., Sjolin, L.: Structure of bovine trypsin-inhibitor: Results of joint neutron and x-ray refinement of crystal form-ii. J. Mol. Biol., **180**, 301–329 (1984)

Index