Springer Optimization and Its Applications 107

Panos M. Pardalos
Anatoly Zhigljavsky
Julius Žilinskas *Editors*

Advances in Stochastic and Deterministic Global Optimization

Springer

Springer Optimization and Its Applications

VOLUME 107

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 *Springer Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs and state-of-the-art expository work 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.

More information about this series at http://www.springer.com/series/7393

Panos M. Pardalos • Anatoly Zhigljavsky
Julius Žilinskas
Editors

Advances in Stochastic and Deterministic Global Optimization

Springer

Editors
Panos M. Pardalos
Department of Industrial
and Systems Engineering
University of Florida
Gainesville, FL, USA

Anatoly Zhigljavsky
Cardiff School of Mathematics
Cardiff University
Cardiff, UK

Julius Žilinskas
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

ISSN 1931-6828 ISSN 1931-6836 (electronic)
Springer Optimization and Its Applications
ISBN 978-3-319-29973-0 ISBN 978-3-319-29975-4 (eBook)
DOI 10.1007/978-3-319-29975-4

Library of Congress Control Number: 2016939972

Mathematics Subject Classification (2010): 90C26, 90C29, 68W20

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

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 (aspirantura) 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 and 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 in 1986–1988, where he founded the Department of Informatics in 1988 and held a position of professor and head of this department in 1988–1993. He worked later as a professor of this department until 2000. He founded the Department of Applied Informatics at Vytautas Magnus University in 1994 and was holding 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; and 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 at Åbo Akademi, Technical University Aachen, Copenhagen University, University College London, and Cardiff University.

A. Žilinskas is a member of Lithuanian Academy of Sciences. He is a member of editorial boards of Journal of Global Optimization, Control and Cybernetics, Informatica, The Open Cybernetics and Systemics Journal, and International Journal of Grid and High Performance Computing. He is a member of IFIP working group Optimization-Based Computer-Aided Modeling and Design and of American Mathematical Society. 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 the 1970s and the 1980s; 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, Copenhagen University (Denmark), and Maastricht University (the 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 200 papers mainly on statistical global optimization theory, algorithms, and applications, 6 monographs, and 6 textbooks; the titles of the monographs are:

- 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)
- Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer (2008)
- Pardalos, P.M., Žilinskas, A., Žilinskas, J.: Non-Convex Multi-Objective Optimization. Springer, New York (2016)

Current research interests of A. Žilinskas are statistical theory of global optimization, multi-objective optimization, optimization-based modeling and design, and analysis of multidimensional data by means of visualization. Research is oriented to development of statistical models for global optimization, implementation and investigation of the corresponding algorithms, and application of these algorithms to practical problems.

This book is dedicated to A. Žilinskas on the occasion of his 70th birthday. The chapters cover some of the research interests of A. Žilinskas. The book is divided into three parts: I. Theory and Algorithms for Global Optimization; II. Applications of Global Optimization; and III. Multi-Objective Global Optimization.

On behalf of all the contributors of this Festschrift, we would like to congratulate Antanas Žilinskas on the occasion of his 70th birthday and wish him well and continued success in his scientific career.

Acknowledgements Panos M. Pardalos was partially supported by the Paul and Heidi Brown Preeminent Professorship in ISE at the University of Florida. The work of Anatoly Zhigljavsky was supported by the Russian Science Foundation, project No. 15-11-30022 "Global optimization, supercomputing computations, and application".

Gainesville, FL, USA — Panos M. Pardalos
Cardiff, UK — Anatoly Zhigljavsky
Vilnius, Lithuania — Julius Žilinskas
January 2016

Contents

List of Contributors

Oluwamayowa Amusat Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK

Jolita Bernatavičienė Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

I. David L. Bogle Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK

James M. Calvin New Jersey Institute of Technology, Newark, NJ, USA

Gareth Davies Cardiff School of Mathematics, Cardiff University, Cardiff, UK

André Deutz Multiobjective Optimization and Decision Analysis (MODA) Research Group, LIACS, Faculty of Science, Leiden University, Leiden, The Netherlands

Gintautas Dzemyda Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Michael Emmerich Multiobjective Optimization and Decision Analysis (MODA) Research Group, LIACS, Faculty of Science, Leiden University, Leiden, The Netherlands

Pascual Fernández Department of Statistics and Operations Research, University of Murcia, Murcia, Spain

Carlos M. Fonseca CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal

Dominique Fortin INRIA, Rocquencourt, France

Eric S. Fraga Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK

Jonathan Gillard Cardiff School of Mathematics, Cardiff University, Cardiff, UK

Mario R. Guarracino High Performance Computing and Networking Institute, National Research Council, Naples, Italy

Rob J. Hyndman Department of Econometrics and Business Statistics, Monash University, Clayton, VIC, Australia,

Audrone Jakaitiene System Analysis Department, Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Olga Kosheleva University of Texas at El Paso, El Paso, TX, USA

Vladik Kreinovich University of Texas at El Paso, El Paso, TX, USA

Olga Kurasova Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Dmitri E. Kvasov Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, Rende (CS), Italy

Department of Software and Supercomputing Technologies, Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia

Algirdas Lančinskas Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Virginijus Marcinkevičius Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Ken McKinnon School of Mathematics, University of Edinburgh, Edinburgh, UK

Viktor Medvedev Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Jonas Mockus Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Marat S. Mukhametzhanov Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, Rende (CS), Italy

Department of Software and Supercomputing Technologies, Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia

Panos M. Pardalos Center for Applied Optimization, University of Florida, Gainesville, FL, USA

Blas Pelegrín Department of Statistics and Operations Research, University of Murcia, Murcia, Spain

Carlos Pérez-Galván Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK

Andrzej Pownuk University of Texas at El Paso, El Paso, TX, USA

Mike Preuss European Research Center for Information Systems (ERCIS), WWU Münster, Münster, Germany

Lina Pupeikiene Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Günter Rudolph Department of Computer Science, Technische Universität Dortmund, Dortmund, Germany

Mara Sangiovanni High Performance Computing and Networking Institute, National Research Council, Naples, Italy

Yaroslav D. Sergeyev Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, Rende (CS), Italy

Department of Software and Supercomputing Technologies, Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia

Mojdeh Shirazi-Manesh School of Mathematical Sciences, Monash University, Clayton, VIC, Australia

Kate Smith-Miles School of Mathematical Sciences, Monash University, Clayton, VIC, Australia

Ingrida Steponavičė School of Mathematical Sciences, Monash University, Clayton, VIC, Australia

Povilas Treigys Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Ider Tseveendorj University of Versailles, Université Paris-Saclay, Versailles, France

Laura Villanova School of Mathematical Sciences, Monash University, Clayton, VIC, Australia

Hao Wang Multiobjective Optimization and Decision Analysis (MODA) Research Group, LIACS, Faculty of Science, Leiden University, Leiden, The Netherlands

Simon Wessing Department of Computer Science, Technische Universität Dortmund, Dortmund, Germany

Kaifeng Yang Multiobjective Optimization and Decision Analysis (MODA) Research Group, LIACS, Faculty of Science, Leiden University, Leiden, The Netherlands

Yu Yu School of Mathematics, University of Edinburgh, Edinburgh, UK

Anatoly Zhigljavsky Cardiff School of Mathematics, Cardiff University, Cardiff, UK

Julius Žilinskas Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Part I
Theory and Algorithms for Global Optimization

On the Asymptotic Tractability of Global Optimization

James M. Calvin

Abstract We consider the intrinsic difficulty of global optimization in high dimensional Euclidean space. We adopt an asymptotic analysis, and give a lower bound on the number of function evaluations required to obtain a given error tolerance. This lower bound complements upper bounds provided by recently proposed algorithms.

Keywords Lower complexity bounds • Tractability • Adaptive algorithms

Introduction

In this chapter we consider the following optimization problem. Given a twice-continuously differentiable function $f : [0,1]^d \to \mathbb{R}$, we can evaluate the function at points $t_1, t_2, \ldots, t_n \in [0,1]^d$, where $t_k = t_k(t_1, f(t_1), t_2, f(t_2), \ldots, t_{k-1}, f(t_{k-1}))$ for $2 \le k \le n$. The goal is to make the error

$$\Delta_n(f) \equiv \min_{1 \le i \le n} f(t_i) - \min_{t \in [0,1]^d} f(t)$$

small.

Results on the worst-case intractability of global optimization are well known. The following is a specialization of a result from [4]; see [8]. Let $F(k,p)$ denote the class of k-times continuously differentiable functions on the d-dimensional unit cube $[0,1]^d$ with the property that

$$\left| \frac{d^k}{dt^k} f(\mathbf{x} + t\mathbf{u}) \right| \le p$$

for all $x \in [0,1]^d$ for any unit vector $\mathbf{u}$. Let A be any optimization algorithm that uses information obtained by evaluating f and its partial derivatives, and assume that for

J.M. Calvin (✉)
New Jersey Institute of Technology, Newark, NJ 07102-1982, USA
e-mail: calvin@njit.edu

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_1

any $f \in F(k,p)$, A is guaranteed to identify an $\mathbf{x}$ such that $f(\mathbf{x}) - f^* \leq \epsilon$. Then there exists a function $f \in F(k,p)$ such that algorithm A requires at least

$$C_{d,k} \cdot \left(\frac{p}{\epsilon}\right)^{d/k}$$

evaluations for input f [8]. This shows that even with uniform bounds on the derivatives, the worst-case number of required function evaluations grows exponentially in the dimension d of the domain.

Global optimizers need not be deterred by this intractability; it is merely a reflection of the fact that a worst-case analysis is inappropriate for global optimization of large classes of continuous functions. If the algorithm has to work with n values of an arbitrary continuous function, no matter how that information is processed the error of any approximation can be arbitrarily large.

If we know that f is convex, then there exist algorithms which obtain an ϵ approximation with a number of evaluations that is polynomial in d and in $\log(p/\epsilon)$ [8]. For non-convex functions, the number of evaluations is exponential in d and in $\log(p/\epsilon)$.

Thus the curse of dimension can be broken by restricting to convex functions. We are interested in alternative ways that do not restrict the functions so much. There are at least three ways to proceed. Each starts with the assumption that the function to be minimized is a member of some given class F.

1. One approach is to fix the objective function and randomize the optimization algorithm. Then when applied to a suitable objective function, the random time until an ϵ approximation is obtained can be analyzed. This is the approach taken in [3, 7]. Examples of randomized algorithms include simulated annealing and genetic algorithms, but we are unaware of error bounds of the type considered here for those methods.
2. Another approach is to randomize the objective function; that is, view the objective function as a sample path of a stochastic process or random field. Much of the work in this area has been initiated by A. Žilinskas, including [10–12]. The questions center on the average number of function evaluations required to obtain an ϵ-approximation. Studies have shown that global optimization is average-case tractable in the univariate setting, and that adaptive methods are much more powerful than nonadaptive methods in this case.

   Adaptive algorithms choose the next function evaluation point as a function of all previous information, while nonadaptive algorithms choose points independently of past information. Examples of the latter class of algorithms are algorithms that choose points independently according to some fixed probability distribution, or according to a fixed grid. Nonadaptive global optimization algorithms are seldom used in practice, and there are good theoretical reasons for why that is the case if we adopt a worst-case perspective. Suppose that our objective function f is in some class of functions F, and assume that F is convex and symmetric. That is, for $0 \leq \lambda \leq 1$ and $f, g \in F$, $-f \in F$ and

$\lambda f + (1-\lambda)g \in F$. This type of class of objective functions provides a natural setting for global optimization problems (for example, k-times continuously differentiable functions). For such a class, adaptive algorithms are essentially no more powerful than nonadaptive algorithms. For any adaptive algorithm that obtains an error of at most ϵ with n function evaluations, there exists a nonadaptive algorithm using no more than $n+1$ function evaluations that also obtains an error of at most ϵ [6].

3. A third approach is to fix the objective function and analyze the asymptotic error as the number of function evaluations tends to infinity. The motivation is that algorithms that are efficient from an asymptotic point of view are likely to be efficient for a moderate number of function evaluations as well. Numerical experiments with algorithms developed with this approach (reviewed below) indicate that they are efficient compared to a sample of alternative algorithms that have appeared in the literature.

In this chapter we adopt the third viewpoint, considering asymptotic error for a certain class of functions. In addition to recalling an algorithm that gives an upper error bound, we describe a lower bound for adaptive algorithms.

Background

The Euclidean norm is denoted by $\|\cdot\|$. Let $F = C^2\left([0,1]^d, \mathbb{R}\right)$ denote the set of twice-continuously differentiable real-valued functions defined on the d-dimensional unit cube. For $f \in F$, let $f^* = \min_{t\in[0,1]^d} f(t)$ denote the global minimum of f. Let $F_1 \subset F$ denote the subset of functions that have a unique global minimizer x^* in the interior of $[0,1]^d$. Let $n(\epsilon, d, f)$ be the minimal number of function evaluations needed to obtain an approximation to the minimum with error at most ϵ for f defined on $[0,1]^d$.

A two-dimensional algorithm based on Delaunay triangulations was presented in [1]. That algorithm has the property that for large enough n (depending on f), the error after n function evaluations is at most

$$c_1 \exp(-c_2\sqrt{n})$$

for c_1, c_2 depending on f. The algorithm presented in [1] was based on a Delaunay triangulation of the domain and the error bound could only be proved for dimension $d = 2$, due to the fact that the algorithm required a certain quality guarantee for the triangulations which could not be proved for higher dimensions.

In [2] a rectangular decomposition was used instead of the Delaunay decomposition. The main result on the convergence rate of the error for that algorithm follows.

Suppose that $f \in F_1$. There is a number $n_0(f)$ such that for $n \geq n_0(f)$,

$$\Delta_n \leq \frac{1}{8}\left\|D^2 f\right\|_{\infty,[0,1]^d} (q\cdot d) \exp\left(-\sqrt{n}\beta(f,d)\right),$$

where $\left\|D^2 f\right\|_{\infty,[0,1]^d}$ is a seminorm measuring the size of the second derivative of f (defined at (3) below), $q \approx 1.27$ is a numerical parameter, and

$$\beta(f,d) = \left(\frac{\exp\left(-\frac{3}{4} - \frac{16}{q}\right) \Gamma(1+d/2) \left(\det(D^2 f(x^*))\right)^{1/2}}{d^2 \left(2\pi q d^2\right)^{d/2}} \right)^{1/2}.$$

Note that the limiting error is smaller, the larger the determinant of the second derivative at the minimizer; a larger second derivative allows the search effort to concentrate more around the minimizer. The convergence rate in terms of the number of function evaluations n is quite fast, but the term $\beta(f,d)$ decreases exponentially fast as d increases: as $d \to \infty$, $\beta(f,d) = O\left(d^{-(d+3)/2}\right)$.

Bounds for Nonadaptive Algorithms

The *dispersion* of a set of points $P_n = \{x_1, x_2, \ldots, x_n\} \subset [0,1]^d$ is given by

$$d_{P_n} = \sup_{x \in [0,1]^d} \min_{1 \le i \le n} \|x - x_i\|;$$

this is the radius of the largest ball in $[0,1]^d$ that contains no point of P_n. (See [5]; here we consider only the Euclidean metric.) For any $n \ge 1$ the dispersion is bounded below by $d_{P_n} = \Omega\left(n^{-1/d}\right)$ ([5], p. 150). In particular, for any sequence,

$$d_{P_n} \ge \left(\frac{\Gamma(1+d/2)}{\pi^{d/2}} \right)^{1/d} n^{-1/d}. \tag{1}$$

Let $f \in F_1$; recall that this means that $f \in C^2\left([0,1]^d\right)$ and that f has a unique minimizer x^* in the interior of the cube. Denote the matrix of second-order partial derivatives by $D^2 f(x^*)$, which is positive definite since $f \in F_1$. Denote the eigenvalues of $D^2(x^*)$ by

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_d > 0.$$

For any nonadaptive algorithm, there exists a function $f \in F$ for which the following lower bound on the error holds:

$$\Delta_n \ge \lambda_d \left(\frac{\Gamma(1+d/2)}{\pi^{d/2}} \right)^{2/d} n^{-2/d}.$$

The bound follows from Taylor's theorem and the lower bound for dispersion given at (1). This means that for any nonadaptive method, for small enough $\epsilon > 0$, about

$$\left(\frac{1}{\epsilon}\right)^{d/2} \left(\frac{\lambda_d}{\pi}\right)^{d/2} \Gamma(1+d/2)$$

function values are needed to obtain an error of at most ϵ.

Using Stirling's approximation for the gamma function,

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

we can express the last bound as

$$\left(\frac{1}{\epsilon}\right)^{d/2} \left(\frac{\lambda_d}{\pi}\right)^{d/2} \sqrt{2\pi/(1+d/2)} \left(\frac{1+d/2}{e} \right)^{d/2}.$$

We therefore see that for nonadaptive methods, global optimization suffers from a "curse of dimension": The number of function evaluations increases exponentially in the dimension d.

Bounds for Adaptive Algorithms

For a compact set K and $f \in C^2(K)$, define the seminorm

$$\left\|D^2 f\right\|_{\infty,K} \equiv \sup_{x \in K} \sup_{\substack{u_1, u_2 \in \mathbb{R}^2 \\ \|u_i\|=1}} \left| D_{u_1} D_{u_2} f(x) \right|, \tag{3}$$

where $D_y f$ is the derivative of f in the direction y. An equivalent definition is

$$\left\|D^2 f\right\|_{\infty,K} \equiv \sup_{x \in K} \sup_{\substack{u \in \mathbb{R}^d \\ \|u\|=1}} \left| u' D^2 f(x) u \right|.$$

This is a measure of the maximum size of the second derivative of f over K.

The following result is a corollary of a theorem proved in [9].

Lemma 1. *Consider a regular simplex T and function $f \in C^2(T)$. Let R denote the radius of the smallest sphere that circumscribes T. Let L denote the linear function that interpolates the values of f at the vertices of the simplex. Then we have the sharp bound*

$$\max_{x \in T} |f(x) - L(x)| \leq \frac{1}{2} R^2 \left\|D^2 f\right\|_{\infty,T}. \tag{4}$$

Lower Error Bound

In this section we derive an asymptotic lower bound for $n(\epsilon, f, d)$ for $f \in F_1$.

Let $B(x,y)$ denote the ball of radius y centered at x:

$$B(x,y) = \{z \in \mathbb{R}^d : \|x - z\| \leq y\}.$$

For $f \in F_1$, there exist positive numbers η and c such that

$$f(x) \geq \underline{f}(x) \equiv f^* + c\,\|x - x^*\|^2$$

for $x \in B(x^*, \eta)$. Extend the definition of $\underline{f}$ to the remainder of $[0,1]^d$ in such a way that $\underline{f} \in C^2$ and $\underline{f}(x) \geq f^* + c\eta^2$ for $x \in [0,1]^d \setminus B(x^*, \eta)$. We obtain a lower bound for $n(\epsilon, f, d)$ by obtaining a lower bound for the number of function evaluations required for an ϵ approximation of the minimum $\underline{f}^* = f^*$ of $\underline{f}$ on $B(x^*, \eta)$. For any set of evaluations, the error on the ball for f will be at least the error for $\underline{f}$, and so it suffices to prove a lower bound for $\underline{f}$ on $B(x^*, \eta)$. For $\underline{f}$, we have

$$\left\| D^2 \underline{f} \right\|_{\infty, K} = 2c, \quad \det(D^2 \underline{f}(x^*)) = (2c)^d$$

for any compact K.

For a set of n points $\{x_1, x_2, \ldots, x_n\} \subset [0,1]^d$, let $\{T_i : 1 \leq i \leq s(n)\}$ denote the Delaunay triangulation of the points. The number of simplexes $s(n)$ has the bound $s(n) = O(n^{\lceil d/2 \rceil})$. Furthermore, define

$$\underline{f}^i = \max_{x \in T_i} \underline{f}(x), \quad v_i = |T_i|, \quad M_n = \min_{1 \leq i \leq s(n)} \underline{f}(x_i), \quad \Delta_n = M_n - f^*,$$

where $|T_i|$ denotes the volume of T_i. For $\Delta_n \leq \epsilon$, we require that for each i,

$$\underline{f}^i - \max_{x \in T_i} \left| L(x) - \underline{f}(x) \right| \geq M_n - \epsilon,$$

or equivalently

$$\underline{f}^i - M_n + \epsilon \geq \max_{x \in T_i} \left| L(x) - \underline{f}(x) \right|.$$

The most favorable geometry of the simplexes, from the point of view of interpolation, is that of the regular simplex. This is not attainable for all the $\{T_i\}$, but since we seek a lower bound on the number of function evaluations let us assume that all the simplexes are regular.

Consider a regular simplex in $\mathbb{R}^d$ with edge length h. The outer radius is

$$R_d(h) \equiv \left(\frac{d}{2(d+1)} \right)^{1/2} h \tag{5}$$

(Jung's theorem) and the volume is

$$V_d(h) \equiv \frac{\sqrt{d+1}}{d! 2^{d/2}} h^d. \tag{6}$$

From Lemma 1 we have the tight bound

$$\max_{x\in T_i} \left|L(x) - \underline{f}(x)\right| \leq \frac{1}{2} R_{T_i}^2 \left\|D^2 \underline{f}\right\|_{\infty,T_i},$$

where R_{T_i} is the radius of the smallest sphere containing the regular simplex T_i. Using (5) and (6) we can relate the radius R_{T_i} to the volume v_i of the simplex by

$$R_{T_i}^2 = v_i^{2/d} \left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right).$$

Therefore, $\Delta_n \leq \epsilon$ entails that

$$\frac{2}{\left\|D^2 \underline{f}\right\|_{\infty,T_i}} \geq \left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right) \frac{v_i^{2/d}}{\underline{f}^i - M_n + \epsilon},$$

or

$$\begin{aligned}\frac{v_i}{(\underline{f}^i - M_n + \epsilon)^{d/2}} &\leq \frac{2^{d/2}}{\left\|D^2 \underline{f}\right\|_{\infty,T_i}^{d/2}} \left(\left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right)\right)^{-d/2} \\ &= c^{-d/2} \left(\left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right)\right)^{-d/2}\end{aligned}$$

for each $i \leq s(n)$. Summing over all simplexes gives

$$\sum_{i=1}^{s(n)} \frac{v_i}{(\underline{f}^i - M_n + \epsilon)^{d/2}} \leq s(n) c^{-d/2} \left(\left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right)\right)^{-d/2}.$$

Letting

$$I_n(\epsilon) \equiv \sum_{i=1}^{s(n)} \frac{v_i}{(\underline{f}^i - M_n + \epsilon)^{d/2}},$$

this implies the bound

$$s(n) \geq I_n(\epsilon) c^{d/2} \left(\left(\frac{d!}{\sqrt{d+1}}\right)^{2/d} \left(\frac{d}{d+1}\right)\right)^{d/2}.$$

Since $n = \Omega(s(n)^{2/(d+1)})$, we have (ignoring a constant term and using Stirling's approximation),

$$n \geq I_n(\epsilon)^{2/(d+1)} c^{d/(d+1)} \left(\left(\frac{d!}{\sqrt{d+1}} \right)^{2/d} \left(\frac{d}{d+1} \right) \right)^{d/(d+1)}$$
$$\geq I_n(\epsilon)^{2/(d+1)} c^{d/(d+1)} \left(\frac{2\pi d}{d+1} \right)^{1/(d+1)} \left(\frac{d}{e} \right)^{2d/(d+1)} \left(\frac{d}{d+1} \right)^{d/(d+1)}.$$

The lower bound will involve the function

$$\mathscr{I}_f(\epsilon) \equiv \int_{[0,1]^d} \frac{dx}{(f(x) - f^* + \epsilon)^{d/2}}$$

for $\epsilon > 0$.

The following lemma is proved in [2].

Lemma 2. *For $f \in F_1$,*

$$\lim_{\epsilon \downarrow 0} \frac{\mathscr{I}_f(\epsilon)}{\log(1/\epsilon)} = \frac{d(2\pi)^{d/2}}{2\Gamma(1+d/2)} \cdot \left(\det(D^2 f(x^*)) \right)^{-1/2} \equiv \alpha(f,d).$$

For $\bar{f}$ we have

$$\alpha(\bar{f},d) = \frac{d(\pi/c)^{d/2}}{2\Gamma(1+d/2)}.$$

Since $I_n(\epsilon) \sim \mathscr{I}(\epsilon)$ as $\epsilon \to 0$ and $n \to \infty$, we have

$$n \geq I_n(\epsilon)^{2/(d+1)} c^{d/(d+1)} \left(\frac{2\pi d}{d+1} \right)^{1/(d+1)} \left(\frac{d}{e} \right)^{2d/(d+1)} \left(\frac{d}{d+1} \right)^{d/(d+1)}$$
$$= \left(\frac{I_n(\epsilon)}{\mathscr{I}_f(\epsilon)} \right)^{2/(d+1)} \left(\frac{\mathscr{I}_f(\epsilon)}{\log(1/\epsilon)} \right)^{2/(d+1)} \log(1/\epsilon)^{2/(d+1)}$$
$$\cdot c^{d/(d+1)} \left(\frac{2\pi d}{d+1} \right)^{1/(d+1)} \left(\frac{d}{e} \right)^{2d/(d+1)} \left(\frac{d}{d+1} \right)^{d/(d+1)}$$
$$\sim \alpha(\bar{f},d)^{2/(d+1)} \log(1/\epsilon)^{2/(d+1)} c^{d/(d+1)} \left(\frac{2\pi d}{d+1} \right)^{1/(d+1)} \left(\frac{d}{e} \right)^{2d/(d+1)} \left(\frac{d}{d+1} \right)^{d/(d+1)}$$

$$= \left(\frac{d(\pi/c)^{d/2}}{2\Gamma(1+d/2)}\right)^{2/(d+1)} \log(1/\epsilon)^{2/(d+1)} c^{d/(d+1)} \left(\frac{2\pi d}{d+1}\right)^{1/(d+1)} \left(\frac{d}{e}\right)^{2d/(d+1)} \left(\frac{d}{d+1}\right)^{d/(d+1)}$$

$$= \pi 2^{-1/(d+1)} \frac{d^3}{d+1} \frac{\log(1/\epsilon)^{2/(d+1)}}{\Gamma(1+d/2)^{2/(d+1)}}.$$

It follows from (2) that

$$\Gamma(1+d/2)^{2/(d+1)} = \Theta(d)$$

as $d \to \infty$, and so we obtain the lower bound

$$n = \Omega\left(d \cdot \log(1/\epsilon)^{2/(d+1)}\right).$$

Note that our assumptions on f are favorable. In the case of constant f ($f(x) \equiv C$ for $x \in [0,1]^d$), the value of $\mathscr{I}(f) = \epsilon^{-d/2}$ and we obtain a lower bound that grows exponentially with d.

The fact that the lower bound given above does not grow exponentially in d of course does not imply that we should expect algorithms that have similar complexity. The assumption of regular simplexes in the Delaunay triangulation is unrealistic and only used to derive a lower bound.

Conclusions

Global optimization is intractable if we restrict ourselves to nonadaptive algorithms. In that case, $n(\epsilon,d,f)$ increases exponentially in the dimension of the domain.

In this paper we have considered asymptotic bounds for a class of twice-continuously differentiable functions with a unique global minimizer. Adaptive algorithms exist for which $n(\epsilon,d,f)$ grows only logarithmically in ϵ^{-1} but with a constant factor growing exponentially in dimension d. We presented a lower bound that does not grow exponentially in dimension, though it is not clear that there can be an algorithm that does not have a bound growing exponentially in the dimension.

Acknowledgements The motivation for this investigation grew out of discussions with A. Žilinskas.

References

1. Calvin, J.M., Žilinskas, A.: On a global optimization algorithm for bivariate smooth functions. J. Optim. Theory Appl. **163**, 528–547 (2014)
2. Calvin, J., Gimbutiene, G., Phillips, W., Žilinskas, A.: On a global optimization algorithm for multivariate smooth functions (2015, submitted)
3. Nekrutkin, V.V., Tikhomirov, A.S.: Speed of convergence as a function of given accuracy for random search methods. Acta Appl. **33**, 89–108 (1993)
4. Nemirovsky, A.S., Yudin, D.B.: Problem Complexity and Method Efficiency in Optimization. Wiley, Chichester (1983)
5. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics, Philadelphia (1992)
6. Ritter, K.: Average-Case Analysis of Numerical Problems. Springer, Berlin (2000)
7. Tikhomirov, A.S.: On the Markov homogeneous optimization method. Comput. Math. Math. Phys. **3**, 361–375 (2006)
8. Vavasis, S.: Complexity issues in global optimization: a survey. In: Horst, R., Pardalos, P. (eds.) Handbook of Global Optimization, vol. 1, pp. 27–41. Kluwer Academic Publishers, Dordrecht (1995)
9. Waldron, S.: Sharp error estimates for multivariate positive linear operators which reproduce the linear polynomials. In: C.K. Chui, L.L. Schumaker (eds.) Approximation Theory IX, vol. 1, pp. 339–346. Vanderbilt University Press, Vanderbilt (1998)
10. Žilinskas, A.: On statistical models for multimodal optimization. Mathematische Operationsforschung und Statistik, ser. Statistics **9**, 255–266 (1978)
11. Žilinskas, A.: Axiomatic characterization of global optimization algorithm and investigation of its search strategy. OR Lett. **4**, 35–39 (1985)
12. Žilinskas, A.: On similarities between two models of global optimization: statistical models and radial basis functions. J. Glob. Optim. **48**, 173–182 (2010)

Combining Interval and Probabilistic Uncertainty: What Is Computable?

Vladik Kreinovich, Andrzej Pownuk, and Olga Kosheleva

Abstract In many practical problems, we need to process measurement results. For example, we need such data processing to predict future values of physical quantities. In these computations, it is important to take into account that measurement results are never absolutely exact, that there is always measurement uncertainty, because of which the measurement results are, in general, somewhat different from the actual (unknown) values of the corresponding quantities. In some cases, all we know about measurement uncertainty is an upper bound; in this case, we have an *interval* uncertainty, meaning that all we know about the actual value is that is belongs to a certain interval. In other cases, we have some information—usually partial—about the corresponding probability distribution. New data processing challenges appear all the time; in many of these cases, it is important to come up with appropriate algorithms for taking uncertainty into account.

Before we concentrate our efforts on designing such algorithms, it is important to make sure that such an algorithm is possible in the first place, i.e., that the corresponding problem is algorithmically computable. In this paper, we analyze the computability of such uncertainty-related problems. It turns out that in a naive (straightforward) formulation, many such problems are not computable, but they become computable if we reformulate them in appropriate practice-related terms.

Keywords Probabilistic uncertainty • Interval uncertainty • Combining different types of uncertainty • Computability • Constructive mathematics • Computable analysis

V. Kreinovich (✉) • A. Pownuk • O. Kosheleva
University of Texas at El Paso, El Paso, TX, USA
e-mail: vladik@utep.edu; ampownuk@utep.edu; olgak@utep.edu

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_2

Formulation of the Problem

Need for Data Processing In practice, we are often interested in a quantity y which is difficult to measure directly. Examples of such quantities are distance to a star, amount of oil in the well, or tomorrow's weather. This is important, since one of the main objectives of science is to predict future values of different quantities.

To estimate such quantities, we find easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y by a known dependence $y = f(x_1, \ldots, x_n)$. For example, to predict the future values of important quantities, we can use the known relations between the current and future values of different quantities.

Once such a relation is known, we measure the auxiliary quantities x_i and use the measurement results $\widetilde{x}_i$ to compute an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y. For example, to predict the future values of physical quantities, we use the results $\widetilde{x}_i$ of measuring the current values x_i of these (and related) physical quantities and the known relations $y = f(x_1, \ldots, x_n)$ between the current (x_i) and future (y) values of different quantities.

The corresponding estimation is what constitutes *data processing*.

Need to Take Uncertainty into Account When Processing Data The resulting estimates are never 100 % accurate:

- measurements are never absolutely accurate,
- physical models used for predictions are usually only approximate, and
- sometimes (like in quantum physics) these models only predict the probabilities of different events.

It is desirable to take this uncertainty into account when processing data.

In some cases, we know all the related probabilities; in this case, we can potentially determine the values of all statistical characteristics of interest: mean, standard deviation, correlations, etc.

In most practical situations, however, we only have partial information about the corresponding probabilities. For example, for measurement uncertainties, often, the only information that we have about this uncertainty is the upper bound Δ on its absolute value; in this case, after we get a measurement result $\widetilde{X}$, the only information that we have about the actual (unknown) value of the corresponding quantity X is that it belongs to the interval $[\widetilde{X} - \Delta, \widetilde{X} + \Delta]$. We may know intervals containing the actual (unknown) cumulative distribution function, we may know bounds on moments, etc. In such situations of partial knowledge, for each statistical characteristic of interest, we can have several possible values. In such cases, we are interested in the interval of possible values of this characteristic, i.e., in the smallest and the largest possible values of this characteristic. In some cases, there are efficient algorithms for computing these intervals, in other cases, the corresponding general problem is known to be NP-hard or even not algorithmically computable; see, e.g., [3].

Studying computability—just like studying NP-hardness—is important, since it prevents us from vain attempts to solve the problem in too much generality, and helps us concentrate on doable cases. In view of this importance, in this paper, we describe the most general related problems which are still algorithmically solvable.

What Is Computable: A Brief Reminder

What Is Computable: General Idea We are interested in processing uncertainty, i.e., in dealing with a difference between the exact models of physical reality and our approximate representation of this reality. In other words, we are interested in models of physical reality.

Why do we need mathematical models in the first place? One of our main objectives is to predict the results of different actions (or the result of not performing any action). Models enable us to predict these results without the need to actually perform these actions, thus often drastically decreasing potential costs. For example, it is theoretically possible to determine the stability limits of an airplane by applying different stresses to several copies of this airplane until each copy breaks, but, if we have an adequate computer-based model, it is cheaper and faster to simulate different stresses on this model without having to destroy actual airplane frames.

From this viewpoint, a model is computable if it has algorithms that allow us to make the corresponding predictions. Let us recall how this general idea can be applied to different mathematical objects.

What Is Computable: Case of Real Numbers In modeling, real numbers usually represent values of physical quantities. This is what real numbers were originally invented for—to describe quantities like length, weight, etc., this is still one of the main practical applications of real numbers.

The simplest thing that we can do with a physical quantity is measure its value. In line with the above general idea, we can say that a real number is computable if we can predict the results of measuring the corresponding quantity.

A measurement is practically never absolutely accurate, it only produces an approximation $\widetilde{x}$ to the actual (unknown) value x; see, e.g., [6]. In modern computer-based measuring instruments, such an approximate value $\widetilde{x}$ is usually a binary fraction, i.e., a rational number.

For every measuring instrument, we usually know the upper bound Δ on the absolute value of the corresponding measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$: $|\Delta x| \le \Delta$. Indeed, without such a bound, the difference Δx could be arbitrary large, and so, we would not be able to make any conclusion about the actual value x; in other words, this would be a wild guess, not a measurement.

Once we know Δ, then, based on the measurement result $\widetilde{x}$, we can conclude that the actual value x is Δ-close to $\widetilde{x}$: $|x - \widetilde{x}| \le \Delta$. Thus, it is reasonable to say that a real number x is computable if for every given accuracy $\Delta > 0$, we can efficiently generate a rational number that approximates x with the given accuracy.

One can easily see that it is sufficient to be able to approximate x with the accuracy 2^{-k} corresponding to k binary digits. Thus, we arrive at the following definition of a computable real number (see, e.g., [8]):

Definition 1. A real number x is called *computable* if there is an algorithm that, given a natural number k, generates a rational number r_k for which

$$|x - r_k| \le 2^{-k}.$$

Comment. It is worth mentioning that not all real numbers are computable. The proof of this fact is straightforward.

Indeed, to every computable real number, there corresponds an algorithm, and different real numbers require different algorithms. An algorithm is a finite sequence of symbols. There are countably many finite sequences of symbols, so there are no more than countably many computable real numbers. And it is known that the set of all real numbers is *not* computable: this was one of the first results of set theory. Thus, there are real numbers which are not computable.

How to Store a Computable Number in the Computer The above definition provides a straightforward way of storing a computable real number in the actual computer: namely, once we fix the accuracy 2^{-k}, all we need to store in the corresponding rational number r_k.

What Is Computable: Case of Functions from Reals to Reals In the real world, there are many dependencies between the values of different quantities. Sometimes, the corresponding dependence is *functional*, in the sense that the values $x_1, \ldots, x_n$ of some quantities x_i uniquely determine the value of some other quantity y. For example, according to the Ohm's Law $V = I \cdot R$, the voltage V is uniquely determined by the values of the current I and the resistance R.

It is reasonable to say that the corresponding function $y = f(x_1, \ldots, x_n)$ is computable if, based on the results of measuring the quantities x_i, we can predict the results of measuring y. We may not know beforehand how accurately we need to measure the quantities x_i to predict y with a given accuracy k. If the original accuracy of measuring x_i is not enough, the prediction scheme can ask for more accurate measurement results. In other words, the algorithm can ask, for each pair of natural numbers $i \le n$ and k, for a rational number r_{ik} such that $|x_i - r_{ik}| \le 2^{-k}$. The algorithm can ask for these values r_{ik} as many times as it needs, all we require is that at the end, we always get the desired prediction. Thus, we arrive at the following definition: [8]:

Definition 2. We say that a function $y = f(x_1, \ldots, x_n)$ from real numbers to real numbers is *computable* if there is an algorithm that, for all possible values x_i, given a natural number ℓ, computes a rational number s_ℓ for which $|f(x_1, \ldots, x_n) - s_\ell| \le 2^{-\ell}$. This algorithm,

- in addition to the usual computational steps,
- can also generate *requests*, i.e., pairs of natural numbers (i, k) with $i \le n$.

As a reply to a request, the algorithm then gets a rational number r_{ik} for which $|x_i - r_{ik}| \le 2^{-k}$; this number can be used in further computations.

It is known that most usual mathematical functions are computable in this sense.

How to Store a Computable Function in a Computer In contrast to the case of a computable real number, here, even if we know the accuracy $2^{-\ell}$ with which we need to compute the results, it is not immediately clear how we can store the corresponding function without explicitly storing the while algorithm.

To make storage easier, it is possible to take into account that in practice, for each physical quantity X_i, there are natural bounds $\underline{X}_i$ and $\overline{X}_i$: velocities are bounded by the speed of light, distances on Earth are bounded by the Earth's size, etc. Thus, for all practical purposes, it is sufficient to only consider values $x_i \in [\underline{X}_i, \overline{X}_i]$. It turns out that for such functions, the definition of a computable function can be simplified:

Proposition 1. *For every computable function $f(x_1,\ldots,x_n)$ on a rational-valued box $[\underline{X}_1,\overline{X}_1]\times\ldots\times[\underline{X}_n,\overline{X}_n]$, there exists an algorithm that, given a natural number ℓ, computes a natural number k such that if $|x_i - x_i'| \le 2^{-k}$ for all i, then*

$$|f(x_1,\ldots,x_n) - f(x_1',\ldots,x_n')| \le 2^{-\ell}.$$

This "ℓ to k" algorithm can be effectively constructed based on the original one.

Comment. For reader's convenience, all the proofs are placed in the special Proofs section.

Because of this result, for each ℓ, to be able to compute all the values $f(x_1,\ldots,x_n)$ with the accuracy $2^{-\ell}$, it is no longer necessary to describe the whole algorithm, it is sufficient to store finitely many rational numbers. Namely:

- We use Proposition 1 to find select a value k corresponding to the accuracy $2^{-(\ell+1)}$.
- Then, for each i, we consider a finite list of rational values

$$r_i = \underline{X}_i,\ \ r_i = \underline{X}_i + 2^{-k},\ \ r_i = \underline{X}_i + 2\cdot 2^{-k},\ldots, r_i = \overline{X}_i.$$

- For each combination of such rational values, we use the original function's algorithm to compute the value $f(r_1,\ldots,r_n)$ with accuracy $2^{-(\ell+1)}$.

These are the values we store.

Based on these stored values, we can compute all the values of the function $f(x_1,\ldots,x_n)$ with the given accuracy $2^{-\ell}$. Specifically, for each combination of computable values $(x_1,\ldots,x_n)$, we can

- compute 2^{-k}-close rational value $r_1,\ldots,r_n$, and then
- find, in the stored list, the corresponding approximation $\widetilde{y}$ to $f(r_1,\ldots,r_n)$, i.e., the value $\widetilde{y}$ for which $|f(r_1,\ldots,r_n) - \widetilde{y}| \le 2^{-(\ell+1)}$.

Let us show that this value $\widetilde{y}$ is indeed the $2^{-\ell}$-approximation to $f(x_1,\ldots,x_n)$.

Indeed, because of our choice of ℓ, from the fact that $|x_i - r_i| \le 2^{-k}$, we conclude that $|f(x_1,\ldots,x_n) - f(r_1,\ldots,r_n)| \le 2^{-(\ell+1)}$. Thus,

$$|f(x_1,\ldots,x_n) - y| \le |f(x_1,\ldots,x_n) - f(r_1,\ldots,r_n)| + |f(r_1,\ldots,r_n) - \widetilde{y}| \le$$
$$2^{-(\ell+1)} + 2^{-(\ell+1)} = 2^{-\ell},$$

i.e., that the value $\widetilde{y}$ is indeed the desired $2^{-\ell}$-approximation to $f(x_1,\ldots,x_n)$.

A Useful Equivalent Definition of a Computable Function Proposition 1 allows us to use the following equivalent definition of a computable function:

Definition 2′. We say that a function $y = f(x_1,\ldots,x_n)$ defined on a rational-valued box $[\underline{X}_1,\overline{X}_1] \times \ldots \times [\underline{X}_n,\overline{X}_n]$ is *computable* if there exist two algorithms:

- the first algorithm, given a natural number ℓ and rational values $r_1,\ldots,r_n$, computes a $2^{-\ell}$-approximation to $f(r_1,\ldots,r_n)$;
- the second algorithm, given a natural number ℓ, computes a natural number k such that if $|x_i - x_i'| \le 2^{-k}$ for all i, then

$$|f(x_1,\ldots,x_n) - f(x_1',\ldots,x_n')| \le 2^{-\ell}.$$

Comment. As a corollary of Definition 2′, we conclude that every computable function is continuous. It should be mentioned, however, that not all continuous function is computable. For example, if a is a non-computable real number, then a linear function $f(x) = a \cdot x$ is clearly continuous but not computable. Indeed, if the function $f(x)$ was computable, we would be able to compute its value $f(1) = a$, and we know that the number a is not computable.

Not All Usual Mathematical Functions Are Computable According to Definition 2′, every computable function is continuous. Thus, discontinuous functions are not continuous, in particular, the following function:

Definition 3. By a *step function*, we mean a function $f(x_1)$ for which:

- $f(x_1) = 0$ for $x < 0$ and
- $f(x_1) = 1$ for $x_1 \ge 0$.

Corollary. *The step function $f(x_1)$ is not computable.*

Comment. This corollary can be proven directly, without referring to a (rather complex) proof of Proposition 2. This direct proof is also given in the Proofs section.

Consequences for Representing a Probability Distribution: We Need to Go Beyond Computable Functions We would like to represent a general probability distribution by its cdf $F(x)$. From the purely mathematical viewpoint, this is indeed the most general representation—as opposed, e.g., to a representation that uses a probability density function, which is not defined if we have a discrete variable.

Since the cdf $F(x)$ is a function, at first glance, it may make sense to say that the cdf is computable if the corresponding function $F(x)$ is computable. For many distributions, this definition makes perfect sense: the cdfs corresponding to uniform, Gaussian, and many other distributions are indeed computable functions.

However, for the degenerate random variable which is equal to $x = 0$ with probability 1, the cdf is exactly the step function, and we have just proven that the step function is not computable. Thus, we need to find an alternative way to represent cdfs, beyond computable functions.

What We Do in this Chapter In this chapter, we provide the corresponding general description:

- first for case when we know the exact probability distribution, and
- then for the general case, when we only have a partial information about the probability distribution.

What We Need to Compute: An Even Briefer Reminder

The ultimate goal of all data processing is to make decision. It is known that a rational decision maker maximizes the expected value of his/her utility $u(x)$; see, e.g., [2, 4, 5, 7]. Thus, we need to be able to compute the expected values of different functions $u(x)$.

There are known procedures for eliciting from the decision maker, with any given accuracy, the utility value $u(x)$ for each x [2, 4, 5, 7]. Thus, the utility function is *computable*. We therefore need to be able to compute expected values of computable functions.

Comment. Once we are able to compute the expected values $E[u(x)]$ of different computable functions, we will thus be able to compute other statistical characteristic such as variance. Indeed, variance V can be computed as $V = E[x^2] - (E[x])^2$.

Simplest Case: A Single Random Variable

Description of the Case Let us start with the simplest case of a single random variable X. We would like to understand in what sense its cdf $F(x)$ is computable.

According to our general application-based approach to computability, this means that we would like to find out what we can compute about this random variable based on the observations.

What Can We Compute About $F(x)$? By definition, each value $F(x)$ is the *probability* that $X \le x$. So, in order to decide what we can compute about the value $F(x)$, let us recall what we can compute about probabilities in general.

What Can We Compute About Probabilities: Case of an Easy-to-Check Event Let us first consider the simplest situation, when we consider a probability of an easy-to-check event, i.e., an event for which, from each observation, we can tell whether this event occurred or not. Such events—like observing head when tossing a coin or getting a total of seven points when throwing two dice—are what probability textbooks start with.

In general, we cannot empirically find the exact probabilities p of such an event. Empirically, we can only estimate *frequencies* f, by observing samples of different size N. It is known that for large N, the difference $d = p - f$ between the (ideal) probability and the observed frequency is asymptotically normal, with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{p \cdot (1-p)}{N}}$. We also know that for a normal distribution, situations when $|d - \mu| < 6\sigma$ are negligibly rare (with probability $< 10^{-8}$), so for all practical purposes, we can conclude that $|f - p| \le 6\sigma$.

If we believe that the probability of 10^{-8} is too high to ignore, we can take 7σ, 8σ, or $k_0 \cdot \sigma$ for an even larger value k_0. No matter what value k_0 we choose, for any given value $\delta > 0$, for sufficiently large N, we get $k_0 \cdot \sigma \le \delta$.

Thus, for each well-defined event and for each desired accuracy δ, we can find the frequency f for which $|f - p| \le \delta$. This is exactly the definition of a computable real number, so we can conclude that the probability of a well-defined event should be a computable real number.

What About the Probability that $X \le x$? The desired cdf is the probability that $X \le x$. The corresponding event $X \le x$ is *not* easy to check, since we do not observe the actual value X, we only observe the measurement result $\widetilde{X}$ which is close to X.

In other words, after repeating the experiment N times, instead of N actual values $X_1, \ldots, X_n$, we only know approximate values $\widetilde{X}_1, \ldots, \widetilde{X}_n$ for which

$$|\widetilde{X}_i - X_i| \le \varepsilon$$

for some accuracy ε. Thus, instead of the "ideal" frequency $f = \mathrm{Freq}(X_i \le x)$—which is close to the desired probability $F(x) = \mathrm{Prob}(X \le x)$—based on the observations, we get a slightly different frequency $f = \mathrm{Freq}(\widetilde{X}_i \le x)$.

What can we say about $F(x)$ based on this frequency? Since $|\widetilde{X}_i - X_i| \le \varepsilon$, the inequality $\widetilde{X}_i \le x$ implies that $X_i \le x + \varepsilon$. Similarly, if $X_i \le x - \varepsilon$, then we can conclude that $\widetilde{X}_i \le x$. Thus, we have

$$\mathrm{Freq}(X_i \le x - \varepsilon) \le f = \mathrm{Freq}(\widetilde{X}_i \le x) \le \mathrm{Freq}(X_i \le x + \varepsilon).$$

We have already discussed that for a sufficiently large sample, frequencies are δ-close to probabilities, so we conclude that

$$\mathrm{Prob}(X \le x - \varepsilon) - \delta \le f \le \mathrm{Prob}(\widetilde{X}_i \le x) \le \mathrm{Prob}(X_i \le x + \varepsilon) + \varepsilon.$$

So, we arrive at the following definition:

Definition 4. We say that a cdf $F(x)$ is *computable* if there is an algorithm that, given rational values x, $\varepsilon > 0$, and $\delta > 0$, returns a rational number f for which

$$F(x-\varepsilon) - \delta \le f \le F(x+\varepsilon) + \delta.$$

How to Describe a Computable cdf in a Computer How can we describe a computable cdf in a computer? The above definition prompts us to store the algorithm computing f, but algorithms may take a long time to compute. It is desirable to avoid such time-consuming computations and store only the pre-computed values—at least the pre-computed values corresponding to the given accuracy. We cannot do this by directly following the above definition, since this definition requires us to produce an appropriate f for all infinitely many possible rational values x. Let us show, however, that a simple and natural modification of this idea makes storing finitely many values possible.

Indeed, for two natural numbers k and ℓ, let us take $\varepsilon_0 = 2^{-k}$ and $\delta_0 = 2^{-\ell}$. On the interval $[\underline{T}, \overline{T}]$, we then select a grid $x_1 = \underline{T}$, $x_2 = \underline{T} + \varepsilon_0, \ldots$ Due to Definition 4, for every point x_i from this grid, we can then find the value f_i for which

$$F(x_i - \varepsilon_0) - \delta_0 \le f_i \le F(x_i + \varepsilon_0) + \delta_0.$$

Let us also set up a grid 0, δ_0, $2\delta_0$, etc., on the interval $[0,1]$ of possible values f_i, and instead of the original values f_i, let us store the closest values $\widetilde{f}_i$ from this grid.

Thus, for each pair (k,ℓ), we store a finite number of rational numbers $\widetilde{f}_i$ each of which take finite number of possible values (clearly not exceeding $1 + 1/\delta_0 = 2^\ell + 1$). Thus, for each k and ℓ, we have finitely many possible approximations of this type.

Let us show that this information is indeed sufficient to reconstruct the computable cdf, i.e., that if we have such finite-sets-of-values for all k and ℓ, then, for each rational x, $\varepsilon > 0$, and $\delta > 0$, we can algorithmically compute the value f needed in the Definition 4.

Indeed, for each ε_0 and δ_0, we can find the value x_i from the corresponding grid which is ε_0-close to x. For this x_i, we have a value $\widetilde{f}_i$ which is δ_0-close to the f_i for which

$$F(x_i - \varepsilon_0) - \delta_0 \le f_i \le F(x_i + \varepsilon_0) + \delta_0.$$

Thus, we have

$$F(x_i - \varepsilon_0) - 2\delta_0 \le \widetilde{f}_i \le F(x_i + \varepsilon_0) + 2\delta_0.$$

From $|x_i - x| \le \varepsilon_0$, we conclude that $x_i + \varepsilon_0 \le x + 2\varepsilon_0$ and $x - 2\varepsilon_0 \le x_i - \varepsilon_0$ and thus, that $F(x - 2\varepsilon_0) \le F(x_i - \varepsilon_0)$ and $F(x_i + \varepsilon_0) \le F(x + 2\varepsilon_0)$. Hence,

$$F(x - 2\varepsilon_0) - 2\delta_0 \le \widetilde{f}_i \le F(x + 2\varepsilon_0) + 2\delta_0.$$

So, if we take ε_0 and δ_0 for which $2\varepsilon_0 \le \varepsilon$ and $2\delta_0 \le \delta$, then we get

$$F(x-\varepsilon) \le F(x-2\varepsilon_0) - 2\delta_0 \le \widetilde{f}_i \le F(x+2\varepsilon_0) + 2\delta_0 \le F(x+\varepsilon) + \delta,$$

i.e., we have the desired double inequality

$$F(x-\varepsilon) - \delta \le \widetilde{f}_i \le F(x+\varepsilon) + \delta,$$

with $f = \widetilde{f}_i$.

Equivalent Definitions Anyone who seriously studied mathematical papers and books have probably noticed that, in addition to definitions of different notions and theorems describing properties of these notions, these papers and books often have, for many of these notions, several different but mathematically equivalent definitions. The motivation for having several definitions is easy to understand: if we have several equivalent definitions, then in each case, instead of trying to use the original definition, we can select the one which is the most convenient to use. In view of this, let us formulate several equivalent definitions of a computable cdf.

Definition 4′. We say that a cdf $F(x)$ is *computable* if there is an algorithm that, given rational values x, $\varepsilon > 0$, and $\delta > 0$, returns a rational number f which is δ-close to $F(x')$ for some x' for which $|x' - x| \le \varepsilon$.

Proposition 2. *Definitions 4 and 4′ are equivalent to each other.*

To get the second equivalent definition, we start with the pairs $(x_i, \widetilde{f}_i)$ that we decided to use to store the computable cdf. When $f_{i+1} - f_i > \delta$, we add intermediate pairs

$$(x_i, f_i + \delta), (x_i, f_i + 2\delta), \ldots, (x_i, f_{i+1}).$$

We can say that the resulting finite set of pairs is (ε, δ)-*close* to the graph

$$\{(x,y) : F(x-0) \le y \le F(x)\}$$

in the following sense.

Definition 5. Let $\varepsilon > 0$ and $\delta > 0$ be two rational numbers.

- We say that pairs (x,y) and (x',y') are (ε,δ)-*close* if $|x-x'| \le \varepsilon$ and $|y-y'| \le \delta$.
- We say that the sets S and S' are (ε,δ)-*close* if:
  - for every $s \in S$, there is a (ε,δ)-close point $s' \in S'$;
  - for every $s' \in S'$, there is a (ε,δ)-close point $s \in S$.

Comment. This definition is similar to the definition of ε-closeness in *Hausdorff metric*, where the two sets S and S' are ε-close if:

- for every $s \in S$, there is a ε-close point $s' \in S'$;
- for every $s' \in S'$, there is a ε-close point $s \in S$.

Definition 4″. We say that a cdf $F(x)$ is *computable* if there is an algorithm that, given rational values $\varepsilon > 0$ and $\delta > 0$, produces a finite list of pairs which is (ε, δ)-*close* to the graph $\{(x, y) : F(x-0) \le y \le F(x)\}$.

Proposition 3. *Definition 4″ is equivalent to Definitions 4 and 4′.*

Comment. Proof of Proposition 3 is similar to the above argument that our computer representation is sufficient for describing a computable cdf.

What Can Be Computed: A Positive Result for the 1-D Case We are interested in computing the expected value $E_{F(x)}[u(x)]$ for computable functions $u(x)$. For this problem, we have the following result:

Theorem 1. *There is an algorithm that:*

- *given a computable cdf $F(x)$,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes $E_{F(x)}[u(x)]$ with accuracy δ.

What If We Only Have Partial Information About the Probability Distribution?

Need to Consider Mixtures of Probability Distributions The above result deals with the case when we have a single probability distribution, and by observing larger and larger samples we can get a better and better understanding of the corresponding probabilities. This corresponds to the ideal situation when all sub-samples have the same statistical characteristics. In practice, this is rarely the case. What we often observe is, in effect, a mixture of several samples with slightly different probabilities. For example, if we observe measurement errors, we need to take into account that a minor change in manufacturing a measuring instrument can cause a slight difference in the resulting probability distribution of measurement errors.

In such situations, instead of a *single* probability distribution, we need to consider a *set* of possible probability distributions.

Another case when we need to consider a set of distributions is when we only have partial knowledge about the probabilities. In all such cases, we need to process sets of probability distributions. To come up with an idea of how to process such sets, let us first recall how sets are dealt with in computations. For that, we will start with the simplest case: sets of numbers (or tuples).

Computational Approach to Sets of Numbers: Reminder In the previous sections, we considered computable numbers and computable tuples (and computable functions). A number (or a tuple) corresponds to the case when we have a complete information about the value of the corresponding quantity (quantities). In practice,

we often only have *partial* information about the actual value. In this case, instead of *single* value, we have a *set* of possible values. How can we represent such sets in a computer?

At first glance, this problem is complex, since there are usually infinitely many possible numbers—e.g., all numbers from an interval, and it is not clear how to represent infinitely many numbers in a computer—which is only capable of storing finite number of bits.

However, a more detailed analysis shows that the situation is not that hopeless: infinite number of values only appears in the idealized case when we assume that all the measurements are absolutely accurate and thus, produce the exact value. In practice, as we have mentioned, measurements have uncertainty and thus, with each measuring instrument, we can only distinguish between finitely many possible outcomes.

So, for each set S of possible values, for each accuracy ε, we can represent this set by a finite list S_ε of possible ε-accurate measurement results. This finite list has the following two properties:

- each value $s_i \in S_\varepsilon$ is the result of an ε-accurate measurement and is, thus, ε-close to some value $s \in S$;
- vice versa, each possible value $s \in S$ is represented by one of the possible measurement results, i.e., for each $s \in S$, there exists an ε-close value $s_i \in S_\varepsilon$.

Comment. An attentive reader may recognize that these two conditions have already been mentioned earlier—they correspond to ε-closeness of the sets S and S_ε in terms of Hausdorff metric.

Thus, we naturally arrive at the following definition:

Definition 6. A set S is called *computable* if there is an algorithm that, given a rational number $\varepsilon > 0$, generates a finite list S_ε for which:

- each element $s \in S$ is ε-close to some element from this list, and
- each element from this list is ε-close to some element from the set S.

Comment. In mathematics, sets which can be approximated by finite sets are known as *compact sets*. Because of this, computable sets are also known as *computable compacts*; see, e.g., [1].

So How Do We Describe Partial Information About the Probability Distribution We have mentioned that for each accuracy (ε, δ), all possible probability distributions can be represented by the corresponding finite lists—e.g., if we use Definition 4″, as lists which are (ε, δ)-close to the corresponding cdf $F(x)$.

It is therefore reasonable to represent a set of probability distributions—corresponding to partial knowledge about probabilities—by finite lists of such distributions.

Definition 7. A set $\mathbf{S}$ of probability distributions is called *computable* if there is an algorithm that, given rational numbers $\varepsilon > 0$ and $\delta > 0$, generates a finite list $\mathbf{S}_{\varepsilon,\delta}$ of computable cdfs for which:

- each element $s \in \mathbf{S}$ is (ε, δ)-close to some element from this list, and
- each element from this list is (ε, δ)-close to some element from the set $\mathbf{S}$.

What Can Be Computed? For the same utility function $u(x)$, different possible probability distributions lead, in general, to different expected values. In such a situation, it is desirable to find the *range* $E_{\mathbf{S}}[u(x)] = [\underline{E}_{\mathbf{S}}[u(x)], \overline{E}_S[u(x)]]$ of possible values of $E_{F(x)}[u(x)]$ corresponding to all possible probability distributions $F(x) \in \mathbf{S}$:

$$\underline{E}_{\mathbf{S}}[u(x)] = \min_{F(x)\in\mathbf{S}} E_{F(x)}[u(x)]; \ \ \overline{E}_{\mathbf{S}}[u(x)] = \max_{F(x)\in\mathbf{S}} E_{F(x)}[u(x)].$$

It turns out that, in general, this range is also computable:

Theorem 2. *There is an algorithm that:*

- *given a computable set* $\mathbf{S}$ *of probability distributions,*
- *given a computable function* $u(x)$*, and*
- *given (rational) accuracy* $\delta > 0$*,*

computes the endpoints of the range $E_{\mathbf{S}}[u(x)]$ *with accuracy* δ.

Comment. This result follows from Theorem 1 and from the known fact that there is a general algorithm for computing maximum and minimum of a computable function on a computable compact; see, e.g., [1].

What to Do in a General Case (Not Necessarily 1-D)

Need to Consider a General Case What if we have a joint distribution of several variable? A random process—i.e., a distribution on the set of functions of one variable? A random field—a probability distribution on the set of functions of several variables? A random operator? A random set?

In all these cases, we have a natural notion of a distance (metric) which is computable, so we have probability distribution on a computable metric space M.

Situations When We know the Exact Probability Distribution: Main Idea In the general case, the underlying metric space M is not always ordered, so we cannot use cdf $F(x) = \text{Prob}(X \le x)$ to describe the corresponding probability distribution.

However, what we observe and measure are still numbers—namely, each measurement can be described by a computable function $g : M \to \mathbb{R}$ that maps each state $m \in M$ into a real number. By performing such measurements many times, we can get the frequencies of different values of $g(x)$. Thus, we arrive at the following definition:

Definition 8. We say that a probability distribution on a computable metric space is *computable* if there exists an algorithm, that, given:

- a computable real-valued function $g(x)$ on M, and
- rational numbers y, $\varepsilon > 0$, and $\delta > 0$,

returns a rational number f which is ε-close to the probability $\mathrm{Prob}(g(x) \le y')$ for some y' which is δ-close to y.

How Can We Represent this Information in a Computer? Since M is a computable set, for every ε, there exists an ε-*net* $x_1, \ldots, x_n$ for M, i.e., a finite list of points for which, for every $x \in M$, there exists an ε-close point x_i from this list, thus

$$X = \bigcup_i B_\varepsilon(x_i), \text{ where } B_\varepsilon(x) \stackrel{\text{def}}{=} \{x' : d(x,x') \le \varepsilon\}.$$

For each computable element x_0, by applying the algorithm from Definition 8 to a function $g(x) = d(x,x_0)$, we can compute, for each ε_0 and δ_0, a value f which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_0))$ for some ε' which is δ_0-close to ε_0.

In particular, by taking $\delta_0 = 2^{-k}$ and $\varepsilon_0 = \varepsilon + 2 \cdot 2^{-k}$, we can find a value f' which is 2^{-k}-close to $\mathrm{Prob}(B_{\varepsilon'}(x_0))$ for some $\varepsilon' \in [\varepsilon + 2^{-k}, \varepsilon + 3 \cdot 2^{-k}]$. Similarly, by taking $\varepsilon_0' = \varepsilon + 5 \cdot 2^{-k}$, we can find a value f'' which is 2^{-k}-close to $\mathrm{Prob}(B_{\varepsilon''}(x_0))$ for some $\varepsilon'' \in [\varepsilon + 4 \cdot 2^{-k}, \varepsilon + 6 \cdot 2^{-k}]$.

We know that when we have $\varepsilon < \varepsilon' < \varepsilon''$ and $\varepsilon'' \to \varepsilon$, then

$$\mathrm{Prob}(B_{\varepsilon''}(x_0) - B_{\varepsilon'}(x_0)) \to 0,$$

so the values f' and f'' will eventually become close. Thus, by taking $k = 1, 2, \ldots$, we will eventually compute the number f_1 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1))$ for all ε' from some interval $[\underline{\varepsilon}_1, \overline{\varepsilon}_1]$ which is close to ε (and for which $\underline{\varepsilon} > \varepsilon$).

We then:

- select f_2 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1) \cup B_{\varepsilon'}(x_2))$ for all ε' from some interval $[\underline{\varepsilon}_2, \overline{\varepsilon}_2] \subseteq [\underline{\varepsilon}_1, \overline{\varepsilon}_1]$,
- select f_3 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1) \cup B_{\varepsilon'}(x_2) \cup B_{\varepsilon'}(x_3))$ for all ε' from some interval $[\underline{\varepsilon}_3, \overline{\varepsilon}_3] \subseteq [\underline{\varepsilon}_2, \overline{\varepsilon}_2]$,
- etc.

At the end, we get approximations $f_i - f_{i-1}$ to probabilities of the sets

$$S_i \stackrel{\text{def}}{=} B_\varepsilon(x_i) - (B_\varepsilon(x_1) \cup \ldots \cup B_\varepsilon(x_{i-1}))$$

for all ε from the last interval $[\underline{\varepsilon}_n, \overline{\varepsilon}_n]$.

These approximations $f_i - f_{i-1}$ form the information that we store about the probability distribution—as well as the values x_i.

What Can We Compute? It turns out that we can compute the expected value $E[u(x)]$ of any computable function:

Theorem 3. *There is an algorithm that:*

- *given a computable probability distribution on a computable metric space,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes the expected value $E[u(x)]$ with accuracy δ.

What If We Have a Set of Possible Probability Distributions? In the case of partial information about the probabilities, we have a set $\mathbf{S}$ of possible probability distributions.

In the computer, for any given accuracies ε and δ, each computable probability distribution is represented by the values $f_1, \ldots, f_n$. A computable set of distributions can be then defined by assuming that, for every ε and δ, instead of a single tuple $(f_1, \ldots, f_n)$, we have a *computable set* of such tuples.

In this case, similar to the 1-D situation, it is desirable to find the *range* $E_{\mathbf{S}}[u(x)] = [\underline{E}_{\mathbf{S}}[u(x)], \overline{E}_S[u(x)]]$ of possible values of $E_P[u(x)]$ corresponding to all possible probability distributions $P \in \mathbf{S}$:

$$\underline{E}_{\mathbf{S}}[u(x)] = \min_{P \in \mathbf{S}} E_{F(x)}[u(x)]; \quad \overline{E}_{\mathbf{S}}[u(x)] = \max_{P \in \mathbf{S}} E_{F(x)}[u(x)].$$

In general, this range is also computable:

Theorem 4. *There is an algorithm that:*

- *given a computable set $\mathbf{S}$ of probability distributions,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes the endpoints of the range $E_{\mathbf{S}}[u(x)]$ with accuracy δ.

Comment. Similarly to Theorem 2, this result follows from Theorem 3 and from the known fact that there is a general algorithm for computing maximum and minimum of a computable function on a computable compact [1].

Proofs

Proof of Proposition 1

1°. Once we can approximate a real number x with an arbitrary accuracy, we can always find, for each k, a 2^{-k}-approximation r_k of the type $\frac{n_k}{2^k}$ for some integer n_k.

Indeed, we can first find a rational number r_{k+1} for which $|x - r_{k+1}| \le 2^{-(k+1)}$, and then take $r_k = \dfrac{n_k}{2^k}$ where n_k is the integer which is the closest to the rational number $2^k \cdot r_{k+1}$. Indeed, for this closest integer, we have $|2^k \cdot r_{k+1} - n_k| \le 0.5$. By dividing both sides of this inequality by 2^k, we get $|r_{k+1} - r_k| = \left|r_{k+1} - \dfrac{n_k}{2^k}\right| \le 2^{-(k+1)}$, and thus, indeed,

$$|x - r_k| \le |x - r_{k+1}| + |r_{k+1} - r_k| \le 2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}.$$

2°. Because of Part 1 of this proof, it is sufficient to consider situations in which, as a reply to all its requests (i,k), the algorithm receives the approximate value r_{ik} of the type $\dfrac{n_{ik}}{2^k}$.

3°. Let us prove, by contradiction, that for given ℓ, there exists a value $k_{\max}$ that bounds, from above, the indices k in the all the requests (i,k) that this algorithm makes when computing a $2^{-\ell}$-approximation to $f(x_1,\ldots,x_n)$ on all possible inputs.

If this statement is not true, this means that for every natural number x, there exists a tuple $x^{(k)} = (x_1^{(k)},\ldots,x_n^{(k)})$ for which this algorithm requests an approximation of accuracy at least 2^{-k} to at least one of the values $x_i^{(k)}$.

Overall, we have infinitely many tuples corresponding to infinitely many natural numbers. As a reply to each request (i,k), we get a rational number of the type $r_{ik} = \dfrac{n_{ik}}{2^k}$. For each natural number m, let us consider the value $\dfrac{p_i}{2^m}$ which is the closest to r_{ik}. There are finitely many possible tuples $(p_1,\ldots,p_n)$, so at least one of these tuples occurs infinitely many times.

Let us select such a tuple t_1 corresponding to $m = 1$. Out of infinitely many cases when we get an approximation to this tuple, we can select, on the level $m = 2$, a tuple t_2 for which we infinitely many times request the values which are 2^{-2}-close to this tuple, etc. As a result, we get a sequence of tuples t_m for which $|t_m - t_{m+1}| \le 2^{-m} + 2^{-(m+1)}$.

This sequence of tuples converges. Let us denote its limit by $t = (t_1,\ldots,t_n)$. For this limit, for each k, the algorithm follows the same computation as the k-th tuple and thus, will request some value with accuracy $\le 2^{-k}$. Since this is true for every k, this means that this algorithm will never stop—and we assumed that our algorithm always stops. This contradiction proves that there indeed exists an upper bound $k_{\max}$.

4°. How can we actually find this $k_{\max}$? For that, let us try values $m = 1,2,\ldots$ For each m, we apply the algorithm $f(r_1,\ldots,r_n)$ to all possible combinations of values of the type $r_i \dfrac{p_i}{2^m}$; in the original box, for each m, there are finitely many such tuples. For each request (i,k), we return the number of the type $\dfrac{n_{ik}}{2^k}$ which is the closest to t_i. When we reach the value $m = k_{\max}$, then, by definition of $k_{\max}$, this would mean that our algorithm never requires approximations which are more accurate than 2^{-m}-accurate ones.

In this case, we can then be sure that we have reached the desired value $k_{\max}$: indeed, for all possible tuples $(x_1,\ldots,x_n)$, this algorithm will never request values beyond this m-th approximation—and we have shown it for all possible combinations of such approximations. The proposition is proven.

Direct Proof of the Corollary to Proposition 1 The non-computability of the step function can be easily proven by contradiction. Indeed, suppose that there exists an algorithm that computes this function. Then, for $x_1 = 0$ and $\ell = 2$, this algorithm produces a rational number s_ℓ which is 2^{-2}-close to the value $f(0) = 1$ and for which, thus, $s_\ell \geq 0.75$. This algorithm should work no matter which approximate values r_{1k} it gets—as long as these values are 2^{-k}-close to x_1. For simplicity, let us consider the case when all these approximate values are 0s: $r_{1k} = 0$.

This algorithm finishes computations in finitely many steps, during which it can only ask for the values of finitely many such approximations; let us denote the corresponding accuracies by $k_1,\ldots,k_m$, and let $K = \max(k_1,\ldots,k_m)$ be the largest of these natural numbers. In this case, all the information that this algorithm uses about the actual value x is that this value satisfies all the corresponding inequalities $|x_1 - r_{1k_j}| \leq 2^{-k_j}$, i.e., $|x_1| \leq 2^{-k_j}$. Thus, for any other value x_1' that satisfies all these inequalities, this algorithm returns the exact same value $s_\ell \geq 0.75$. In particular, this will be true for the value $x_1' = -2^{-K}$. However, for this negative value x_1', we should get $f(x_1') = 0$, and thus, the desired inequality $|f(x_1') - y_\ell| \leq 2^{-2}$ is no longer satisfied. This contradiction proves that the step function is not computable.

Proof of Proposition 2 It is easy to show that Definition 4′ implies Definition 4. Indeed, if f is δ-close to $F(x')$ for some $x' \in [x-\varepsilon, x+\varepsilon]$, i.e., if $F(x') - \delta \leq f \leq F(x') + \delta$, then, due to $x - \varepsilon \leq x' \leq x + \varepsilon$, we get $F(x-\varepsilon) \leq F(x')$ and $F(x') \leq F(x+\varepsilon)$ and thus, that

$$F(x-\varepsilon) \leq F(x') - \delta \leq f \leq F(x') + \delta \leq F(x+\varepsilon) + \delta,$$

i.e., the desired inequality

$$F(x-\varepsilon) \leq f \leq F(x+\varepsilon) + \delta.$$

Vice versa, let us show that Definition 4 implies Definition 4′. Indeed, we know that $F(x+\varepsilon) - F(x+\varepsilon/3) \to 0$ as $\varepsilon \to 0$. Indeed, this difference is the probability of X being in the set $\{X : x+\varepsilon/3 \leq X \leq x+\varepsilon\}$, which is a subset of the set $S_\varepsilon \stackrel{\text{def}}{=} \{X : x < X \leq x+\varepsilon\}$. The sets S_ε form a nested family with an empty intersection, thus their probabilities tend to 0 and thus, the probabilities of their subsets also tend to 0.

Due to Proposition 4, for each $k = 1, 2, \ldots$, we can take $\varepsilon_k = \varepsilon \cdot 2^{-k}$ and find f_k and f_k' for which

$$F(x+\varepsilon_k/3) - \delta/4 \leq f_k \leq F(x+(2/3)\cdot\varepsilon_k) + \delta/4$$

and

$$F(x+(2/3)\cdot\varepsilon_k)-\delta/4\le f_k'\le F(x+\varepsilon_k)+\delta/4.$$

From these inequalities, we conclude that

$$-\delta/2\le f_k'-f_k\le F(x+\varepsilon_k)-F(x+\varepsilon_k/3)+\delta/2.$$

Since $F(x+\varepsilon_k)-F(x+\varepsilon_k/3)\to 0$ as $k\to\infty$, for sufficiently large k, we will have $F(x+\varepsilon_k)-F(x+\varepsilon_k/3)\le\delta/4$ and thus, $|f_k'-f_k|\le(3/4)\cdot\delta$. By computing the values f_k and f_k' for $k=1,2,\ldots$, we will eventually reach an index k for which this inequality is true. Let us show that this f_k is then δ-close to $F(x')$ for $x'=x+(2/3)\cdot\varepsilon_k$ (which is ε_k-close—and thus, ε-close—to x).

Indeed, we have

$$f_k\le F(x+(2/3)\cdot\varepsilon_k)+\delta/4\le F(x+(2/3)\cdot\varepsilon_k)+\delta.$$

On the other hand, we have

$$F(x+(2/3)\cdot\varepsilon_k)-\delta/4\le f_k'\le f_k+(3/4)\cdot\delta$$

and thus,

$$F(x+(2/3)\cdot\varepsilon_k)-\delta\le f_k\le F(x+(2/3)\cdot\varepsilon_k)+\delta.$$

The equivalence is proven.

Proof of Theorem 1 We have shown, in Proposition 1, that every computable function $u(x)$ is computably continuous, in the sense that for every $\delta_0>0$, we can compute $\varepsilon>0$ for which $|x-x'|\le\varepsilon$ implies $|u(x)-u(x')|\le\delta_0$.

In particular, if we take ε corresponding to $\delta_0=1$, and take the ε-grid $x_1,\ldots,x_i,\ldots$, then we conclude that each value $u(x)$ is 1-close to one of the values $u(x_i)$ on this grid. So, if we compute the 1-approximations $\widetilde{u}_i$ to the values $u(x_i)$, then each value $u(x)$ is 2-close to one of these values $\widetilde{u}_i$. Thus, $\max\limits_x|u(x)|\le U\stackrel{\text{def}}{=}\max\limits_i\widetilde{u}_i+2$. So, we have a computable bound $U\ge 2$ for the (absolute value) of the computable function $u(x)$.

Let us once again use computable continuity. This time, we select ε corresponding to $\delta_0=\delta/4$, and take an x-grid $x_1,\ldots,x_i,\ldots$ with step $\varepsilon/4$. Let G be the number of points in this grid.

According to the equivalent form (Definition 4′) of the definition of computable cdf, for each of these grid points x_i, we can compute the value f_i which is $(\delta/(4U\cdot G))$-close to $F(x_i')$ for some x_i' which is $(\varepsilon/4)$-close to x_i.

The function $u(x)$ is $(\delta/4)$-close to a piece-wise constant function $u'(x)$ which is equal to $u(x_i)$ for $x\in(x_i',x_{i+1}']$. Thus, their expected values are also $(\delta/4)$-close: $|E[u(x)]-E[u'(x)]|\le\delta/4$.

Here, $E[u'(x)] = \sum_i u(x_i) \cdot (F(x'_{i+1}) - F(x'_i))$. But $F(x'_i)$ is $(\delta/(4U \cdot G))$-close to f_i and $F(x'_{i+1})$ is $(\delta/(4U \cdot G))$-close to f_{i+1}. Thus, each difference $F(x'_{i+1}) - F(x'_i)$ is $(\delta/(2U \cdot G))$-close to the difference $f_{i+1} - f_i$.

Since $|u(x_i)| \le U$, we conclude that each term $u(x_i) \cdot (F(x'_{i+1}) - F(x'_i))$ is $(\delta/(2G))$-close to the computable term $u(x_i) \cdot (f_{i+1} - f_i)$. Thus, the sum of G such terms—which is equal to $E[u'(x)]$—is $(\delta/2)$-close to the computable sum

$$\sum_i u(x_i) \cdot (f_{i+1} - f_i).$$

Since $E[u'(x)]$ is, in its turn, $(\delta/4)$-close to desired expected value $E[u(x)]$, we thus conclude that the above computable sum

$$\sum_i u(x_i) \cdot (f_{i+1} - f_i)$$

is indeed a δ-approximation to the desired expected value.

The theorem is proven.

Proof of Theorem 3 The proof is similar to the proof of Theorem 1: we approximate the function $u(x)$ by a $(\delta/2)$-close function $u'(x)$ which is piece-wise constant, namely, which is equal to a constant $u_i = u(x_i)$ on each set

$$S_i = B_\varepsilon(x_i) - (B_\varepsilon(x_1) \cup \ldots \cup B_\varepsilon(x_{i-1})).$$

The expected value of the function $u'(x)$ is equal to $E[u'(x)] = \sum_i u_i \cdot \mathrm{Prob}(S_i)$.

The probabilities $\mathrm{Prob}(S_i)$ can be computed with any given accuracy, in particular, with accuracy $\delta/(2U \cdot n)$, thus enabling us to compute $E[u'(x)]$ with accuracy $\delta/2$.

Since the functions $u(x)$ and $u'(x)$ are $(\delta/2)$-close, their expected values are also $(\delta/2)$-close. So, a $(\delta/2)$-approximation to $E[u'(x)]$ is the desired δ-approximation to $E[u(x)]$.

The theorem is proven.

Conclusions

When processing data, it is important to take into account that data comes from measurements and is, therefore, imprecise. In some cases, we know the probabilities of different possible values of measurement error—in this case, we have a probabilistic uncertainty. In other cases, we only know the upper bounds on the measurement error; in this case, we have interval uncertainty.

In general, we have a partial information about the corresponding probabilities: e.g., instead of knowing the exact values of the cumulative distribution function (cdf) $F(x) = \text{Prob}(X \le x)$, we only know bounds on these values—i.e., in other words, an interval containing such bounds. In such situations, we have a combination of probabilistic and interval uncertainty.

The ultimate goal of data processing under uncertainty is to have efficient algorithms for processing the corresponding uncertainty, algorithms which are as general as possible. To come up with such algorithms, it is reasonable to analyze which of the related problems are algorithmically solvable in the first place: e.g., is it possible to always compute the expected value of a given computable function?

In this chapter, we show that a straightforward (naive) formulation, most corresponding problems are not algorithmically solvable: for example, no algorithm can always, given the value x, compute the corresponding value $F(x)$ of the cdf.

However, we also show that if we instead formulate these problems in practice-related terms, then these problems become algorithmically solvable. For example, if we take into account that the value x also comes from measurement and is, thus, only known with some accuracy, it no longer makes sense to look for an approximation to $F(x)$; instead, it is sufficient to look for an approximation to $F(x')$ for some x' which is close to x, and it turns out that such an approximation is always computable.

Acknowledgements This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

The authors are thankful to Walid Taha and to all the participants of the Second Hybrid Modeling Languages Meeting HyMC (Houston, Texas, May 7–8, 2015) for valuable discussions, and to the anonymous referees for useful suggestions.

References

1. Bishop, E., Bridges, D.: Constructive Analysis. Springer, Heidelberg (1985)
2. Fishburn, P.C.: Utility Theory for Decision Making. Wiley, New York (1969)
3. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht (1997)
4. Luce, R.D., Raiffa, R.: Games and Decisions: Introduction and Critical Survey. Dover, New York (1989)
5. Nguyen, H.T., Kosheleva, O., Kreinovich, V.: Decision making beyond Arrow's 'impossibility theorem', with the analysis of effects of collusion and mutual attraction. Int. J. Intell. Syst. **24**(1), 27–47 (2009)
6. Rabinovich, S.: Measurement Errors and Uncertainties: Theory and Practice. Springer, New York (2005)
7. Raiffa, H.: Decision Analysis. Addison-Wesley, Reading, MA (1970)
8. Weihrauch, K.: Computable Analysis. Springer, Berlin (2000)

Survey of Piecewise Convex Maximization and PCMP over Spherical Sets

Ider Tseveendorj and Dominique Fortin

Abstract The main investigation in this chapter is concerned with a piecewise convex function which can be defined by the pointwise minimum of convex functions, $F(x) = \min\{f_1(x), \ldots, f_m(x)\}$. Such piecewise convex functions closely approximate nonconvex functions, that seems to us as a natural extension of the piecewise affine approximation from convex analysis. Maximizing $F(\cdot)$ over a convex domain have been investigated during the last decade by carrying tools based mostly on linearization and affine separation. In this chapter, we present a brief overview of optimality conditions, methods, and some attempts to solve this difficult nonconvex optimization problem. We also review how the line search paradigm leads to a radius search paradigm, in the sense that sphere separation which seems to us more appropriate than the affine separation. Some simple, but illustrative, examples showing the issues in searching for a global solution are given.

Keywords Piecewise convex • Nonconvex optimization • Nonsmooth optimization

Introduction

Convexity is a central concept in optimization. Solving optimization problems somehow leads to separate the constraint set and the set of points no worse than a given candidate. In the convex optimization case, both sets are convex which makes the separation affordable by a hyperplane. However, when one deals with nonconvex optimization problems, one needs more appropriate tools because both sets or at least one of them can be nonconvex. A decade-long effort for finding such tools is

I. Tseveendorj (✉)
University of Versailles, Université Paris-Saclay, 45 avenue des États-Unis, 78035 Versailles Cedex, France
e-mail: Ider.Tseveendorj@uvsq.fr

D. Fortin
INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France
e-mail: Dominique.Fortin@inria.fr

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_3

presented briefly in the beginning of the chapter. The second part of the chapter is mainly devoted towards the ways of use spherical sets for the separation in nonconvex optimization instead of hyperplanes used in convex optimization.

We consider a nonconvex nonsmooth optimization problem:

$$\begin{cases} \text{maximize} & F(x) \\ \text{subject to} & x \in D, \end{cases} \qquad (PCMP)$$

where D is a nonempty compact in $\mathscr{R}^n$ and $F : \mathscr{R}^n \to \mathscr{R}$ is a piecewise convex function.

Definition 1. A function F is called a piecewise convex function iff it can be decomposed into the pointwise minimum of convex functions, namely:

$$F(x) = min\{f_j(x) \mid j \in M\},$$

where all functions $f_j : \mathscr{R}^n \to \mathscr{R}$ are convex $j \in M := \{1, 2..., m\}$.

Convex functions, concave functions are particular cases of piecewise convex functions, since clearly F is a convex when $m = 1$ and a concave whenever all functions f_j are affine.

For any real number $\alpha \in R$ the Lebesgue set of a function f is defined like

$$\mathscr{L}_f(\alpha) = \{x \mid f(x) \leq \alpha\}.$$

A quasiconvex (in particular convex f_j) function has the property that its Lebesgue set (in particular $\mathscr{L}_{f_j}(\alpha)$) is a convex set. Piecewise convex functions have a nice geometrical interpretation that the Lebesgue set of such function is the union of a finite number of convex sets

$$\mathscr{L}_F(\alpha) = \bigcup_{j=1}^{m} \mathscr{L}_{f_j}(\alpha).$$

For a given $y \in \mathscr{R}^n$ the Lebesgue set $\mathscr{L}_F(F(y))$ defines also, in the sense of maximization, the set of points no better than y. The Lebesgue sets of piecewise convex function $\mathscr{L}_F(F(y))$ are generally nonconvex and can be disconnected or even discrete sets. As a result, the nonconvexity of the objective function $F(\cdot)$ poses the major difficulty for solving piecewise convex maximization problems since they generally have a large number of local optima which are not global optima.

Before solving an optimization problem it is useful to investigate the information about where global optima are attained? At some extreme points, on the boundary, or in the interior of the feasible set, etc. With this respect, it is well known [10, 11, 17, 22] that the global maximum occurs at an extreme point for convex maximization over a convex set while a solution to DC (difference of convex) optimization lies on the boundary of the feasible set. As regards $(PCMP)$, in general,

its large number of local optima including the global maximum can lie anywhere in D. As such, it is computationally very difficult to solve, especially if one wishes to find the global optimum.

The space of piecewise convex functions has been studied in [8]; any continuous nonconvex function can be approximated by piecewise convex one. In addition virtually many optimization problems can, theoretically, be approximated by $(PCMP)$. Indeed, the latter justifies the importance of this class of optimization problems as a powerful tool in nonconvex optimization.

The reader is referred to [2, 11, 13, 15, 17, 18, 22, 26] for finding out the close relationship between $(PCMP)$ and the other nonconvex optimization problems like convex maximization, reverse convex minimization, DC, Lipschitz optimization, etc.

Despite the concerns mentioned above, $(PCMP)$ does not seem to have been extensively studied.

The purpose of this chapter is twofold:

- to present a brief survey of some useful results, optimality conditions, methods, ideas for $(PCMP)$;
- to propose a novel approach of solving $(PCMP)$ based on nonlinear separation and consider piecewise convex maximization problems over spherical sets (balls, spheres) which play the key role for this new research direction.

A Survey of Studies on PCMP

During the last decade we have been focusing on tools for solving $(PCMP)$. This section provides a brief survey of optimality conditions, methods, and some useful ideas for $(PCMP)$.

Global Optimality Conditions

Let us quote the article [20] for the global optimality conditions. First, we define an active index set at any z by

$$I(z) = \{i \in M \mid f_i(z) = F(z)\}$$

and at given $k \in M, z \in D$ a special subset of D by

$$D_k(z) = \{x \in D \mid f_j(x) > F(z) \text{ for all } j \in M \setminus \{k\}\}.$$

The following results summarize our findings on optimality conditions so far:

Proposition 1 ([20]). *If $z \in D$ is a global maximum of $(PCMP)$ then for all $k \in I(z)$*

$$\partial f_k(y) \bigcap N(D_k(z), y) \neq \emptyset \text{ for all } y \text{ such that } f_k(y) = F(z)$$

Theorem 1 ([20]). *Let $z \in D$ and assume there exist $k \in I(z)$ and $v \in \mathscr{R}^n$ such that $f_k(v) < f_k(z)$. Then a sufficient condition for z to be the global maximum for $(PCMP)$ is*

$$\partial f_k(y) \bigcap N(clco(D_k(z)), y) \neq \emptyset \forall y \text{ such that } f_k(y) = F(z).$$

Here *cl*, *co* stand for a closure and a convex hull of a set, respectively.

These necessary and sufficient conditions show that solving $(PCMP)$ leads to choose one function f_k and to maximize it over $D_k(\cdot)$ or over $clco(D_k(\cdot))$. This is the well known convex maximization problem

$$\begin{cases} \text{maximize} & f(x) \\ \text{subject to} & x \in D, \end{cases} \qquad (CM)$$

its optimality conditions have been obtained in [19]

$$\partial f(y) \bigcap N(D, y) \neq \emptyset \text{ such that } y \text{ such that } f(y) = f(z).$$

In [20], one can find geometric interpretation of the optimality conditions along with their illustrations in some examples.

Methods

Linearization Oriented Algorithm

To our knowledge, the first algorithm for solving $(PCMP)$ has been presented in [4]. The article provides an algorithm based on optimality conditions (Proposition 1 and Theorem 1.) presented in the previous subsection. For the sake of simplicity of presentation it is assumed that functions $f_j(\cdot), j \in M$ are strongly convex quadratic, the domain D a full dimensional polytope.

Quadratic convex maximization problems (the particular case of $(PCMP)$ when $m = 1$) are normally classified as NP-hard. Furthermore, just finding a local maximum of a nonconvex quadratic programming problems is NP-hard too. Thus, even the local solution search for $(PCMP)$ is not trivial. At the same time an efficient algorithm for finding local maxima may be the crucial factor in design of the global maximum search stage. Therefore, in [4] a local search algorithm for $(PCMP)$ has attracted considerable attention. For this important issue of the local search, an algorithm derived from linearization of convex functions is proposed, and its convergence is examined carefully.

Assume that we are given a local solution $y \in D$. In order to improve the best known local solution, according to the optimality conditions, one should look for a

point in $D_k(y)$. Then a practical global search algorithm is provided which combines the local search algorithm with successive inner and outer approximations to $D_k(\cdot)$.

Computational experiments on small examples in $\mathscr{R}^2, \mathscr{R}^3$ are reported, which show the efficiency of the approach. One can find also therein some details of implementation as well as new notions like an intersection graph on the Lebesgue sets $\mathscr{L}_{f_j}(F(\cdot))$, the relationship of the optima of $(PCMP)$ with Helly's theorem from discrete geometry.

Piece Adding TeCHnique (PATCH for Short)

For nonconvex optimization problems, many standard techniques rely on local search and the challenge still remains to escape from a local maximum area.

Among other things the question about how to escape from a local maximum area was investigated in article [8]. The authors first studied the space of piecewise convex functions and showed that this class is closed under operations like addition, positive scalar multiplication, operations "min", F^+, F^-. One can add missing operation "max" into the above list of operations, which has been observed recently.

The following one dimensional example given in [7] well illustrates the idea behind, so-called, the piece adding technique.

$$\begin{cases} \text{maximize } x^2 - 2x, \\ \text{subject to } 0 \leq x \leq 3. \end{cases}$$

Obviously, $x = 0$ is a local maximum, (but not the global !), an accumulation point of local search algorithms with a starting point $x^0 \in [0,1]$ and there is no clear way to escape from its region, once one is therein. The main idea for escaping is, it makes sense first to add into the objective function a convex piece $x^2 - 4$ issued from the local solution $x = 0$, then solve the following problem

$$\begin{cases} \text{maximize } \ min\{x^2 - 2x, x^2 - 4\}, \\ \text{subject to } \ 0 \leq x \leq 3. \end{cases}$$

Notice that the previous local solution $x = 0$ is a minimum of the new convex piece $x^2 - 4$. Now it is clear (Fig. 1) that any local search will not get back to $x = 0$ and easily finds the global solution $x = 3$ from any starting point.

Similarly for escaping from a local solution y of $(PCMP)$, it is proposed to solve the following problem

$$\text{maximize } \ min\{F(x) - F(y), p(x)\}, \text{ subject to } x \in D \qquad (PP)$$

where $p(x)$ is a convex function, that will be specified hereafter.

Remark 1. Here we underline that by adding a convex piece into objective function

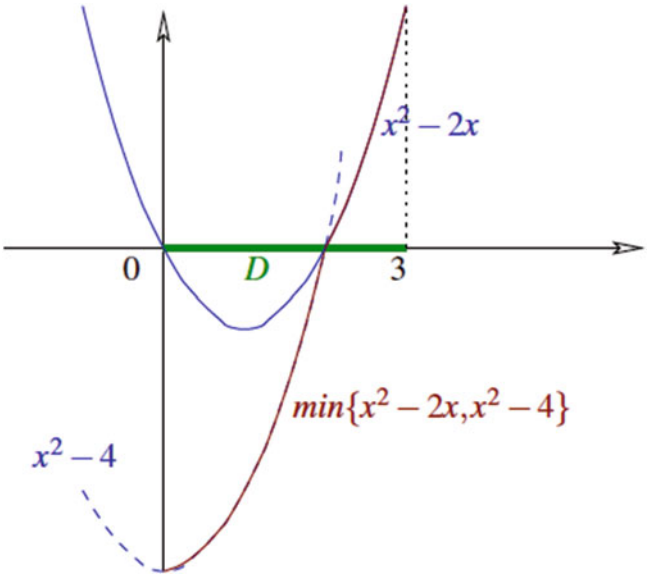

Fig. 1 Piecewise Adding Technique (PATCH) idea

- one cuts off *virtually* from D a subset of points no better than y; *("virtual cut" in the sense that we add a convex piece in objectives rather than a reverse convex constraint $p(x) \geq 0$)*
- the objective function remains piecewise convex;
- the space of the problem is unchanged.

The function $p(\cdot)$ may be defined in different ways among which we select two.

Definition 2. Let z be the global solution to $(PCMP)$. A strictly convex function $p_y : \mathscr{R}^n \to \mathscr{R}$ is called **a patch** around a local solution y of the problem $(PCMP)$ iff

- $p_y(y) < 0$;
- $p_y(x) \geq 0$ for all $x \in D$ such that $F(x) > F(y)$;
- moreover $p_y(z) > 0$

We notice that the conditions for patches are not easy to verify with respect to unknown global solution z, therefore we weaken those conditions and introduce a relaxed function with almost the same features.

Definition 3. A strictly convex function $p_v : \mathscr{R}^n \to \mathscr{R}$ is called **a pseudopatch** at $v \in D$ iff

- $p_v(v) < 0$ and
- there is $u \in D$ such that $F(u) > F(v)$ and $p_v(u) \geq 0$.

Unlike *the patch*, the *virtual* cutting $p_v(x) \geq 0$ defined by *the pseudopatch* could cut a feasible point v together with some better points (even the global solution!)

but it should leave at least one better point. For practical purposes, pseudopatches are added in the objective function temporarily then, after they are dropped at each improvement since the global solution z can be incidentally cut off by a pseudopatch.

Outline of algorithm

1. Compute a local solution y^k $(k = 1,2,\dots)$ obtained from a feasible point $u \in D$;
2. Construct a strictly convex function $p_k(\cdot)$ as a pseudopatch or a patch around y^k;
3. Solve

$$\text{maximize } \Phi_k(x), \text{ subject to } x \in D, \qquad (P_k)$$

where

$$\Phi_k(x) := \min\{F(x) - F(y^k), p_1(x), p_2(x), \dots, p_k(x)\}.$$

Let u be an optimal solution to (P_k);

4. If $p_k(\cdot)$ is a pseudopatch then drop it from $\Phi_k(x)$;
5. Repeat the sequence with $k = k+1$

In a like manner, after one iteration we either obtain a better point due to pseudopatch or reduce *virtually* the domain by new patch $p_k(\cdot)$ around y^k.

For ease of presenting the main result, let consider two problems for a given local solution y:

$$\text{maximize } F(x) - F(y), \text{ subject to } x \in D, \qquad (PCMP)$$

and

$$\text{maximize } \Phi(x) \text{ subject to } x \in D, \qquad (PP)$$

where $\Phi(x) := \min\{F(x) - F(y), p_y(x)\}$, $p_y(\cdot)$ is a patch around y.

Assumption 1. *Let us assume that $p_y(z) \geq F(z) - F(y)$ at the global solution z.*

Proposition 2. *If z is a global solution to (PP) and $p_y(\cdot)$ is a patch satisfying Assumption 1 then z solves globally $(PCMP)$ too.*

This technique aims also at carrying piecewise affine approximation from convex optimization to piecewise convex approximation for the nonconvex case; since among others DC and Lipschitz functions have locally tight piecewise convex majorants, it shows the potential strength of this approach.

The key tool lies behind the *virtual* cutting function; we call it either a patch to avoid cycling through the same local solutions or a pseudo patch to early detect a better point.

Attractive Force Search Algorithm

Newton's law of universal gravitation states that *any two bodies in the universe attract each other with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them.*

$$\mathscr{F} = \mathscr{G}\frac{m_1 \times m_2}{r^2}$$

defines *the attractive force between two bodies that possess masses* m_1, m_2, respectively.

Inspired by the well known law, the authors provided an algorithm in article [9] that calculates an improvement from a current local solution y of $(PCMP)$ by using some analogy of Newton's attractive force.

How does one search for an improvement from a feasible solution x? Up until now, there have been used a local solution search algorithm to find a local solution y with subsequent checking the inclusion $D \subset \mathscr{L}_F(F(y))$ for a possible further improvement. Of course, one can check also the inclusion $D \subset \mathscr{L}_F(F(x))$ directly at $x \in D$. On the other hand, since

$$\mathscr{L}_F(F(\cdot)) = \bigcup_{i=1}^{m} \mathscr{L}_{f_i}(F(\cdot))$$

an improvement can occur when

$$D \not\subset \bigcup_{i=1}^{m} \mathscr{L}_{f_i}(F(\cdot)).$$

Anyway, for an improvement we seek a point in D, but outside of all the Lebesgue sets $\mathscr{L}_{f_i}(F(\cdot))$.

If we imagine each of these nonempty Lebesgue sets $\mathscr{L}_{f_i}(F(y))$ as a convex body that possesses mass, then a direction of improvement at a current point y could be calculated by analogy to Newton's attractive force.

In [9] a feasible set is assumed to be a full dimensional polytope defined by linear constraints

$$D = \{x \in \mathscr{R}^n \mid Ax \leq b\} = \{x \in \mathscr{R}^n \mid \langle a^j, x\rangle \leq b_j, j = 1,2,\ldots,k\}.$$

To deal with the polyhedral domain we consider implicitly convex functions that approximate linear functions: $f_j(x) \approx b_j - \langle a^j, x\rangle$. The linear constraints $b_j - \langle a^j, x\rangle \geq 0$ in the domain, may be seen as the linearization of reverse convex constraints $f_j(x) \geq 0$ for some strictly convex quadratic functions $f_j(\cdot)$; (a huge flat ellipsoid). This viewpoint looks strange at first sight, however, it turns all the Lebesgue sets of constraints, objective functions into convex bodies. The interaction between these convex bodies seems to be related to Newton's law, which gives the main idea of our

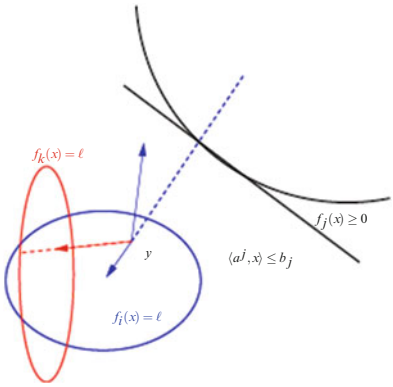

Fig. 2 Newton's gravitational force

algorithm. At a given point y, two kind of forces can be involved from each body (the Lebesgue set): attracting and repelling forces.

An example, given in Fig. 2, shows for y

- a repelling force from $\mathscr{L}_{f_i}(\ell)$ body,
- an attracting force from $\mathscr{L}_{f_k}(\ell)$ body

since $f_i(y) < \ell$ and $f_k(y) > \ell$, and in addition, a repelling force from the constraint. More precisely, we select respectively

- the gradient of the function at y multiplied by a positive (resp. negative) scalar in repelling (resp. attracting) case as an analogy for the direction of Newton's force;
- the distance from y to $\mathscr{L}_{f_i}(\ell)$ as an analogy for the distance between the masses.

Now at y, we are able to compute a gravitational force as the weighted sum of all attracting/repelling forces from each body similarly with Newton's attractive force model:

$$G(y) = \sum_{i \in M} \frac{F(\mathrm{Pr}_{f_i}(y)) - F(y)}{\| \mathrm{Pr}_{f_i}(y) - y \|^2} \frac{\nabla f_i(y)}{\| \nabla f_i(y) \|},$$

where $\mathrm{Pr}_{f_i}(y)$ stands for projection of y on the Lebesgue set $\mathrm{Pr}_{\mathscr{L}_{f_i}(\ell)}(y)$ with ℓ, the best known level set value.

Remark 2. When i is active $i \in I(y)$, in other words $f_i(y) = \ell$ the projection is replaced by c_i the center of $\mathscr{L}_{f_i}(\ell)$ which is a minimum of $f_i(\cdot)$ and the gradient by the direction $x - c_i$ (the *mass* of the convex body is supposed at center). For active linear constraints, the center is assumed at infinity so that no *attracting/repelling force* contributes to $G(x)$.

Before describing the algorithm, let us introduce a set which appears very convenient in data structure.

Definition 4. The resolving border of $F(\cdot)$ at level set value ℓ is defined like the set of points:

$$\mathrm{rb}(F,\ell) = \left\{x \in \mathscr{R}^n \mid \exists i, f_i(x) = \ell, F(x) \leq \ell\right\}.$$

For each i, we may distinguish three parts:

1. $f_i(x) = \ell, f_j(x) > \ell$ for all $j \neq i$
2. $f_i(x) = \ell, f_j(x) \geq \ell$ for all $j \neq i$
3. $f_i(x) = \ell, f_j(x) > \ell$ for some $j \neq i$.

We notice that cases (1.) and (2.) easily lead to a better point.

A resolving border data structure stores points according to active functions ordered by decreasing values of $F(\cdot)$.

For sake of consistency with previous versions of global search algorithm [4, 8] we still refer to **Newton's Attractive force Search Algorithm** as a local search, but it clearly outperforms a strict local searching since it cruises in the surroundings of the resolving border.

PCMP local Search (Newton's Attractive force Search Algorithm)

- $(D, M, w^j \in D)$
- Initialize RB=(key,sortedset) associating map
- y^1=setAndBetter(w^j,RB)
- **if** y^1 not null **then return** y^1
- dir $= G(w^j)$; Newton attraction at w^j
- $u = w^j + \alpha$ dir ; $\alpha > 0$ gives the nearest intersection with the resolving border
- y^2=setAndBetter(u,RB)
- **if** y^2 not null **then return** y^2
- **else return** findBetter(RB)

Relationship with Other Optimization Problems

- **Combinatorial Optimization** Many practical problems give rise to combinatorial optimization problems can be formulated by the binary constraint $x \in \{0,1\}^n$, by the permutation constraint like the assignment problems. The following two articles [5, 6] are devoted to continuous approaches for combinatorial optimization problems. The hardness of these problems consists in nonconvex domains. The current subsection highlights a couple of ideas about how to solve some combinatorial optimization problems using $(PCMP)$.

  We consider the multiknapsack problem [5]

$$\begin{cases} \max \ \langle c, x\rangle \\ \text{s.t.} \quad Ax \leq b \\ \qquad x \in \{0,1\}^n \end{cases} \qquad (MKP)$$

Since the binary constraint $x_i \in \{0,1\}$ can be written like

$$x_i(x_i-1) \geq 0, 0 \leq x_i \leq 1,$$

(MKP) has an equivalent continuous formulation

$$\begin{cases} \max \ \langle c,x \rangle \\ \text{s.t.} \quad Ax \leq b \\ \qquad x \in [0,e] \\ \qquad x_i(x_i-1) \geq 0, \forall i = 1,\ldots,n, \end{cases}$$

where $e = (1,\ldots,1)^\top \in \mathscr{R}^n$. Introducing a function

$$\varphi(x) = min\{x_i(x_i-1) \mid i = 1,2,\ldots,n\},$$

we replace n constraints $x_i(x_i-1) \geq 0,\ i = 1,\ldots,n$ with a constraint $\varphi(x) \geq 0$ and obtain an equivalent to (MKP) problem

$$\begin{cases} \max \ \varphi(x) \\ \text{s.t.} \quad Ax \leq b \\ \qquad x \in [0,e] \\ \qquad \langle c,x-y \rangle \geq 0. \end{cases}$$

for an admissible point y of (MKP). The latter is the piecewise convex maximization problem with n pieces.

Let assume that there is suitable index set's division $J_1,\ldots,J_m$ such that $\bigcup_{i=1}^m J_i = \{1\ldots n\}$ and $J_i \cap J_j = \emptyset,\ \forall i \neq j$. Then it holds also

$$x \in \{0,1\}^n \Leftrightarrow f_1(x) \geq 0,\ldots,f_m(x) \geq 0, 0 \leq x \leq e,$$

where $f_i(x)$ denotes $f_{J_i}(x)$ defined like

$$f_J(x) = \Sigma_{i \in J} \left(x_i - \frac{1}{2}\right)^2 - \frac{|J|}{4}$$

for any $J \subseteq \{1\ldots n\}$ of $|J|$ elements. In this way we obtain $(PCMP)$ where number of pieces m much less than the dimension of the problem $(m < n)$

$$\begin{cases} \max \ min\{f_i(x) \mid i = 1\ldots m\} \\ \text{s.t.} \quad Ax \leq b \\ \qquad x \in [0,e] \\ \qquad \langle c,x-y \rangle \geq 0. \end{cases}$$

But, in practice, the nature of pieces as well as a number of pieces are crucial for solving $(PCMP)$. One can find in [5] a practical algorithm with $m = 2$ of this approach along with computational results in contrast with the best known solutions found by heuristics from combinatorial optimization.

What concerns the permutation constrained problems investigated in [6], in order to retain our focus on common features of continuous optimization, first, we study shortly relationship between $(PCMP)$ and DC optimization, then as a result $(PCMP)$ formulation for the well known quadratic assignment problems (QAP) is given. We consider a problem of maximization of a difference of two convex functions f, g over a convex compact $D \subset \mathscr{R}^n$

$$\begin{aligned} &\max f(x) - g(x) \\ &\text{s.t.} \quad x \in D. \end{aligned}$$

It is straightforward to turn it into $(PCMP)$ of dimension $n+1$ by introducing another variable $t = g(x)$ and splitting the equality constraint into a convex constraint $g(x) - t \leq 0$ while the converse inequality is *dualized* to add a new piece as $F(x,t) = \min\{f(x) - t, g(x) - t\}$. Then, it is equivalent to solving the following problem

$$\begin{aligned} \max \quad & F(x,t) \\ \text{s.t.} \quad & x \in D, t \in \mathscr{R} \\ & g(x) - t \leq 0. \end{aligned}$$

- **Multicriteria Optimization**
  We consider multicriteria optimization problem

$$\begin{cases} \text{minimize} \ \ \Omega(x), \\ \text{subject to } x \in D \end{cases} \qquad (MOP)$$

where $\Omega(\cdot)$ is a vector valued function from $\mathscr{R}^n$ to $\mathscr{R}^m$ whose components are the convex functions $f_i(\cdot), i \in M = \{1, 2, \ldots, m\}$ namely :

$$\Omega(x) = (f_1(x), \ldots, f_m(x))^\top \in \mathscr{R}^m.$$

Let us formally recall the definition of Pareto optimal solutions.

Definition 5. A solution $y \in D$ is called Pareto optimal, if there is no $x \in D$ such that $f_i(x) \leq f_i(y), i = 1, \ldots, m$ and $f_j(x) < f_j(y)$ for some $j \in M$.

An interesting relationship between $(PCMP)$ and multicriteria optimization is presented [21] and afterwards in [9].

We recall also some basic definitions and results from multicriteria optimization.

Definition 6.

– $y \in D$ is called weakly Pareto optimal if there is no $x \in D$ such that

$$f_i(x) < f_i(y),\ \forall i = 1,\ldots,m$$

– $y \in D$ is called strictly Pareto optimal if there is no $x \in D, x \neq y$ such that

$$f_i(x) \leq f_i(y),\ \forall i = 1,\ldots,m \text{ and } f_j(x) < f_j(y) \text{ for some } j \in M.$$

We define also so called *the level curve, the strict level set* of $f(\cdot)$ at α respectively

$$\mathscr{L}_f^{=}(\alpha) = \{x \,|\, f(x) = \alpha\},\ \ \mathscr{L}_f^{<}(\alpha) = \{x \,|\, f(x) < \alpha\}$$

Theorem 2 ([1], Chap. 2). *Let $y \in D$ then*

1. y is strictly Pareto optimal if and only if

$$\bigcap_{i=1}^{m} \mathscr{L}_{f_i}(f_i(y)) = \{y\}.$$

2. y is Pareto optimal if and only if

$$\bigcap_{i=1}^{m} \mathscr{L}_{f_i}(f_i(y)) = \bigcap_{i=1}^{m} \mathscr{L}_{f_i}^{=}(f_i(y)).$$

3. y is weakly Pareto optimal if and only if

$$\bigcap_{i=1}^{m} \mathscr{L}_{f_i}^{<}(f_i(y)) = \emptyset.$$

The following results summarize our findings so far:

Proposition 3. *If y is local maximum to $(PCMP)$ such that $y \in int(D)$ then for some $\epsilon > 0$*

$$F(y) = \min_{x \in B(y,\varepsilon)} max\{f_i(x) \mid i \in I(y)\},$$

where $B(y,\varepsilon)$ stands for the ball of radius ε, centered at y.

Let us denote $\Omega_N(\cdot) \in \mathscr{R}^{|N|}$ with corresponding components $f_i(\cdot)$ for $i \in N \subset M$. Proposition 3 together with Theorem 2 provide the following relationship.

Theorem 3. *If y is local maximum to $(PCMP)$ such that $y \in int(D)$ then y is strictly Pareto optimal to the following multicriteria optimization problem:*

$$\begin{cases} \textit{minimize } \ \Omega_{I(y)}(x), \\ \textit{subject to } x \in D \end{cases}$$

Remark 3. Weakly Pareto optimality of y is implied by Proposition 6.5 from [1].

Abstract Nonlinear Covering Method

The remainder of the chapter is devoted to new approach of solving $(PCMP)$.

For all maximization problems, in particular for $(PCMP)$, clearly, z is the global maximum iff all points of the domain are no better than z, in other words:

$$D \subset \mathscr{L}_F(F(z)).$$

Since as we observed early $\mathscr{L}_F(F(y)) = \bigcup_{i=1}^m \mathscr{L}_{f_i}(F(y))$. the above inclusion means that

- for all x no better than z (i.e. $F(x) \leq F(z)$) there exists $j \in M$ such that

$$x \in \mathscr{L}_{f_j}(F(z)),$$

- if there is $u \in D$ better than z (i.e. $F(z) < F(u)$) then u does not belong to any Lebesgue set:

$$u \notin \mathscr{L}_{f_i}(F(z)) \text{ for all } i \in M.$$

In order to present the main idea of a new approach for solving $(PCMP)$, we give a definition along with an abstract result on an equivalence of problems.

Definition 7. An open subset C satisfying conditions

$$C \subset \mathscr{L}_F(F(y)) \text{ and } C \neq int(\mathscr{L}_F(F(y)))$$

is called **a covering set at level** $F(y)$.

Proposition 4. *Let y be a feasible point for $(PCMP)$ such that*

$$F(y) = max\{F(x) \mid x \in D\} - \delta$$

for some $\delta > 0$. Let also C be a covering set at level $F(y)$. Then the following problem is equivalent to $(PCMP)$:

$$\begin{cases} \textit{maximize } \ F(x) \\ \textit{subject to } \ x \in D \setminus C. \end{cases} \qquad (CC)$$

The main algorithmic feature now looks like

- to cover the feasible set (the domain) by a union of covering sets.
- if the domain is covered by C totally, then stop and the global optimum is found.
- otherwise, solve problem (CC) for an improvement.

In other words, one have to construct an "(union of covering sets)" such that

$$D \subset \text{ (union of covering sets)} \subset \mathscr{L}_F(F(z)).$$

Starting with an initial guess of covering sets, a method bootstraps its way up to ever more accurate "sandwich" approximations to answer "the global optimum" or "improvement". What concerns the covering set, the first that comes to mind, is use balls (spherical set) as a simpler nonlinear shape.

(*PCMP*) over Spherical Sets

Let $S(c;\rho)$ and $B(c;\rho)$ be respectively a sphere and a ball of center c and radius ρ. This section is devoted to the following two (*PCMP*), solutions of them are going to be key tools in nonlinear separation as stated in previous section.

$$\begin{cases} \text{maximize } F(x), \\ \text{subject to } x \in S(c;\rho), \end{cases} \quad \begin{cases} \text{maximize } F(x), \\ \text{subject to } x \in B(c;\rho). \end{cases}$$

We turn our attention to the problem over a sphere, that is a problem of piecewise maximization over a nonconvex feasible set with an empty interior. Since all feasible points are degenerated even the local search for this problem is worth-while to study. We consider a convex function f and notice that the KKT optimality condition for an optimizer of f (either min or max) over $S(c,\rho)$ implies collinearity of gradient $\nabla f(u)$ and $u-c$. It is well known that the projected gradient method, from a reasonably good starting point, quickly converges to the optimal solution. If no meaningful guess is known, the gradient $\nabla f(c)$ is a good ray direction to find a starting point for maximizing f over the sphere; for minimizing we start from the antigradient $-\nabla f(c)$.

Since $F(x) = \min\{f_1(x),\ldots,f_m(x)\}$ a piecewise convex function; using $2\times m$ times the above optimization paradigm (for both $l^j = \arg\min_S f_j(x)$ and $u^j = \arg\max_S f_j(x)$) yields a *sparse* set of points which the actual values $F(l^j)$ and $F(u^j)$ may be computed for all $j = 1,\ldots,m$. We call for the best value of $\max\{F(l^i),F(u^j) \mid i,j = 1,\ldots,m\}$ as sparse piecewise convex maximization value over a sphere and denote by z the point of this value.

Now we borrow from [9] the resolving border heuristic that focuses on points on the level set in the vicinity of $D_k(\cdot)$ for some k that are likely to improve the easy sparse optimizers.

We use the following two examples as the main lead to illustrate the different steps in Piecewise Convex Maximizing:

Algorithm 1 PCMP-over-sphere: Resolving border for $argloc \max_{x\in S} F(x)$

Input: $c, \rho, F = \{f_j \mid j = 1,m\}$
Output: $z = argloc \max_{x\in S} F(x)$
  pq=PriorityQueue(sparseoptimize(S,F)) // decreasing values of sparse $argloc$ max
  while pq $\neq \emptyset$ **do**
    u =pop(pq)
    if $F(u)$ better **then**
      $z = u$
    end if
    for $m = 1, M$ **do**
      $v = \Pr_{f_j(x)=F(z)}(u)$
      $u^j = c + \rho \frac{v-c}{\|v-c\|}$ push(pq,$F(u^j), u^j$) //enqueue value for unprocessed points only
    end for
  end while
  return z

$$
\begin{aligned}
F_1^{2D} = \min\{\ & x_1^2 + (x_2+4)^2 - 36, \\
& (x_1+8)^2 + (x_2-3)^2 - 36, \\
& x_1^2 + (x_2-8)^2 - 16, \\
& (x_1-8)^2 + (x_2-3)^2 - 53, \\
& (x_1-10)^2 + (x_2+10)^2 - 4\}, \\
F_2^{2D} = \min\{\ & x_1^2 + (x_2+2)^2 - 9, \\
& 9(x_1+3)^2 + 4x_2^2 - 36, \\
& (x_1+1)^2 + (x_2-4)^2 - 4, \\
& \tfrac{1}{9}(x_1-3)^2 + \tfrac{1}{36}(x_2-4)^2 - 1, \\
& (x_1-5)^2 + (x_2+5)^2 - 1\}
\end{aligned}
$$

respectively in spherical domain (spheres/balls) with $(c,\rho) = ([0,0],4)$.

Despite there is no proof of global optimality, we experimented an effective behavior to find points on the sphere well ordered by the priority queue and likely to lay close to $\cap D_j(z)$ (the best z found is drawn as a blue dot on Fig. 3).

In the remainder we concentrate on elementary coding steps, starting on optimization over a sphere towards more involved recursive calls of ($PCMP$) over a ball.

There is a subtlety when $v = c$ for defining w; we choose $w = \frac{z+c}{2}$ betting on a point in $\cap D_k(z)$ (early detection of a better point). In both cases Figs. 3 and 4 blue points were found quickly which localize the global solution areas.

Concluding Remarks

In this chapter we have presented an overview of studies on piecewise convex maximization problems: necessary and sufficient global optimality conditions, methods including linearization oriented algorithm, piece adding technique (patch for short), and Newton's attractive force search algorithm (nasa for short). A novel

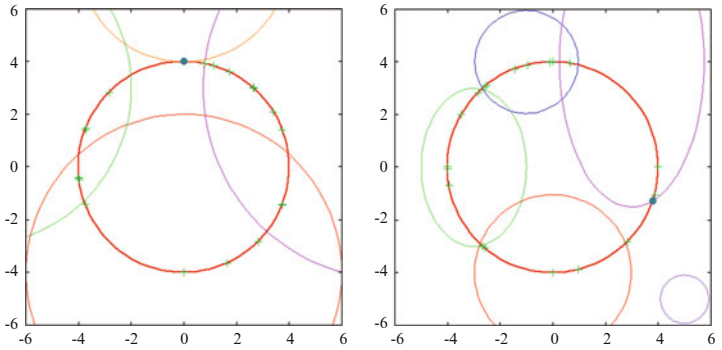

Fig. 3 Maximize F_i^{2D} over a sphere, $i = 1, 2$ (*left*, *right*)

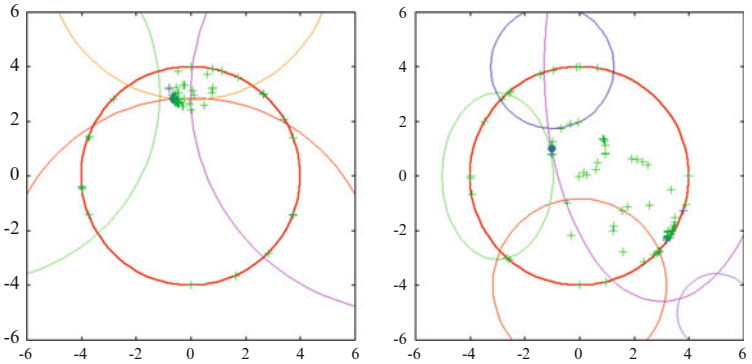

Fig. 4 Maximize F_i^{2D} over a ball, $i = 1, 2$ (*left*, *right*)

approach of solving $(PCMP)$ based on a nonlinear separation (instead of the traditional affine separation) has been proposed. We call it a nonlinear covering method. As a rule the approach opens up a field which should be investigated more thoroughly: a choice of the covering sets, algorithms of solving subproblems over them, etc.

Our aim is now to construct an "(union of covering sets)" such that

$$D \subset \text{ (union of covering sets)} \subset \mathscr{L}_F(F(z))$$

to check the global optimality of z for $(PCMP)$.

As a preliminary we have considered piecewise convex maximization problems over spherical sets (balls, spheres) which play the key role for this new research direction. Algorithms along with some computational results with simple, but illustrative, examples have been reported.

Algorithm 2 PCMP-over-ball: Recursive radius search for $argloc \max_{x\in B} F(x)$

Input: $c, \rho, F = \{f_j \mid j = 1,\dots,m\}$
Output: $z = argloc \max_{x\in B} F(x)$
  $r = \rho$
  $z = c$
  for $l = 1$,maxiterations **do**
    if $r \le \varepsilon$ **then**
      return z
    end if
    $z =$ **PCMP-over-sphere**$(S(c;r), F)$ // $z = argloc \max_S F$ State locz $= z$
    for $j = 1, m$ **do**
      $v = \Pr_{f_j(x)=F(z)}(u)$ // projection on non active function
      if $d(v,z) \le \varepsilon$ or $d(v,z) > 2r$ **then**
        continue // skip projection too close or too far
      end if
      if $d(v,c) > r$ **then**
        $w = c + \frac{r}{d(v,c)}(v-c)$ // outside $B(c;r) \Rightarrow w \in S(c;r)$
      else
        $w = z + \frac{d(v,z)}{r}(c-z)$ // inside $B(c;r) \Rightarrow w \in B(c;r)$
      end if
      if w better locz **then**
        locz $= w$
      end if
    end for
    if locz better z **then**
      $\rho = \min\{\| z - \text{locz} \|, \| c - \text{locz} \|\}$
      z=**PCMP-over-ball**$(B(\text{locz}, \rho), F)$
      $r = \| c - z \|$
    else
      $r = \frac{r}{2}$ // halve radius
    end if
  end for
  return z

Acknowledgements This research benefited from the support of the FMJH "Program Gaspard Monge for optimization and operations research", and from the support from EDF.

The authors acknowledge use of the IBM ILOG CPLEX under the academic initiative license.

The authors would like to thank the anonymous referee for his/her useful comments on this chapter.

References

1. Ehrgott, M.: Multicriteria Optimization. Lecture Notes in Economics and Mathematical Systems, vol. 491. Springer, Berlin (2000)
2. Enkhbat, R.: Quasiconvex Programming. National University of Mongolia, Ulaanbaatar (2004)
3. Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gümüş, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: Handbook of Test Problems in Local and Global Optimization. Nonconvex Optimization and its Applications, vol. 33. Kluwer Academic Publishers, Dordrecht (1999)

4. Fortin, D., Tsevendorj, I.: Piecewise-convex maximization problems: algorithm and computational experiments. J. Global Optim. **24**(1), 61–77 (2002). doi:10.1023/A:1016221931922. http://dx.doi.org/10.1023/A:1016221931922
5. Fortin, D., Tseveendorj, I.: Piecewise convex maximization approach to multiknapsack. Optimization **58**(7), 883–895 (2009). doi:10.1080/02331930902945033. http://dx.doi.org/10.1080/02331930902945033
6. Fortin, D., Tseveendorj, I.: A trust branching path heuristic for permutation problems. Int. J. Pure Appl. Math. **56**(3), 329–343 (2009)
7. Fortin, D., Tseveendorj, I.: Piece adding technique for convex maximization problems. J. Global Optim. **48**(4), 583–593 (2010). doi:10.1007/s10898-009-9506-z. http://dx.doi.org/10.1007/s10898-009-9506-z
8. Fortin, D., Tseveendorj, I.: Piecewise convex maximization problems: piece adding technique. J. Optim. Theory Appl. **148**(3), 471–487 (2011). doi:10.1007/s10957-010-9763-5. http://dx.doi.org/10.1007/s10957-010-9763-5
9. Fortin, D., Tseveendorj, I.: Attractive force search algorithm for piecewise convex maximization problems. Optim. Lett. **6**(7), 1317–1333 (2012). doi:10.1007/s11590-011-0395-y. http://dx.doi.org/10.1007/s11590-011-0395-y
10. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 2nd edn. Springer, Berlin (1993). doi:10.1007/978-3-662-02947-3. http://dx.doi.org/10.1007/978-3-662-02947-3
11. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Nonconvex Optimization and its Applications, vol. 48, 2nd edn. Kluwer Academic Publishers, Dordrecht (2000). doi:10.1007/978-1-4615-0015-5. http://dx.doi.org/10.1007/978-1-4615-0015-5
12. Mordukhovich, B.S.: Variational Analysis and Generalized Differentiation. I. Basic Theory Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 330. Springer, Berlin (2006)
13. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. Springer Briefs in Optimization. Springer, New York (2014). doi:10.1007/978-1-4614-9093-7. http://dx.doi.org/10.1007/978-1-4614-9093-7
14. Penot, J.P.: Duality for anticonvex programs. J. Global Optim. **19**(2), 163–182 (2001)
15. Sergeyev, Y.D., Kvasov, D.: Diagonal global optimization methods (in Russian). Fizmatlit, Moscow (2008)
16. Strekalovskiĭ, A.S.: On the problem of the global extremum. Dokl. Akad. Nauk SSSR **292**(5), 1062–1066 (1987)
17. Strekalovskii, A.S.: Elements of Nonconvex Optimization (in Russian). NAUKA, Novosibirsk (2003)
18. Törn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989). doi:10.1007/3-540-50871-6. http://dx.doi.org/10.1007/3-540-50871-6
19. Tseveendorj, I.: On the conditions for global optimality. J. Mong. Math. Soc. (2), 58–61 (1998)
20. Tsevendorj, I.: Piecewise-convex maximization problems: global optimality conditions. J. Global Optim. **21**(1), 1–14 (2001). doi:10.1023/A:1017979506314. http://dx.doi.org/10.1023/A:1017979506314
21. Tseveendorj, I., Fortin, D.: Pareto optima and local optimality of piecewise convex maximization problems. In: Proceedings of the 15-th Baikal International Conference on Optimization Methods and Their Applications, "Mathematical Programming", vol. 2, pp. 25–29. ISDCT SB of the Russian Academy of Sciences' Publisher, Irkutsk (2011)
22. Tuy, H.: Convex Analysis and Global Optimization. Nonconvex Optimization and Its Applications, vol. 22. Kluwer Academic Publishers, Dordrecht (1998)
23. Zălinescu, C.: On the maximization of (not necessarily) convex functions on convex sets. J. Global Optim. **36**(3), 379–389 (2006)

24. Zhiglyavskiĭ, A.A., Zhilinskas, A.G.: Metody poiska globalnogo ekstremuma. “Nauka”, Moscow (1991). With an English summary
25. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer Optimization and Its Applications, vol. 9. Springer, New York (2008)
26. Zhilinskas, A.: Globalnaya optimizatsiya. “Mokslas”, Vilnius (1986). Aksiomatika statisticheskikh modelei, algoritmy, primeneniya. [Axiomatics of statistical models, algorithms, and applications.] With Lithuanian and English summaries

Assessing Basin Identification Methods for Locating Multiple Optima

Simon Wessing, Günter Rudolph, and Mike Preuss

Abstract Basin identification is an important ingredient in global optimization algorithms for the efficient use of local searches. An established approach for this task is obtaining topographical information about the objective function from a discrete sample of the search space and representing it in a graph structure. So far, different variants of this approach are usually assessed by evaluating the performance of a whole optimization algorithm using them as components. In this work, we compare two approaches on their own, namely topographical selection and nearest-better clustering, regarding their ability to identify the distinct attraction basins of multimodal functions. We show that both have different strengths and weaknesses, as their behavior is very dependent on the problem instance.

Keywords Basin identification • Multi-local optimization • Topographical selection • Nearest-better clustering • Sampling

Introduction

Two-stage algorithms for global optimization are meta-heuristics consisting of a global and a local stage, which are executed alternatingly [25, p. 14]. The global stage is responsible for exploration, the local one for exploitation, as, for example, in the algorithm in [13]. The local stage is usually understood as a local search algorithm started at a certain point and running until convergence is detected. As this is quite expensive in terms of function evaluations, the global stage ideally has the capability of carefully selecting promising starting points [25, p. 66]. This task shall be denoted as *basin identification* [16, Sect. 3.2]. Originally, it was accomplished

S. Wessing (✉) • G. Rudolph
Department of Computer Science, Technische Universität Dortmund, Dortmund 44227, Germany
e-mail: simon.wessing@tu-dortmund.de; guenter.rudolph@tu-dortmund.de

M. Preuss
European Research Center for Information Systems (ERCIS),
WWU Münster, Münster 48149, Germany
e-mail: mike.preuss@uni-muenster.de

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_4

by conventional clustering methods [25, pp. 95–116], but nowadays more refined methods have been developed, which are not necessarily clustering methods in a strict sense anymore [22].

In this work, we directly compare two basin identification methods regarding their ability to detect distinct attraction basins. Correspondingly, the task of approximating all or at least several local optima of a multimodal objective function in one optimization run has received increased attention of the optimization community in recent years. We collect appearances of this topic in the classical global optimization literature in section "Literature Report" and try to reconstruct why it has somehow become known under the term *multimodal optimization* in the evolutionary computation community [7]. We propose to use the term *multi-local optimization* instead.

Anyway, it is clear that the aim of finding, say, the best k optima is only a slight shift in perspective, as global optimization is a special case of it. This change in perspective may be partly due to the rise of a-posteriori approaches for multiobjective optimization, where it is considered a critical property of the algorithms that they are able to generate whole Pareto front approximations, and it is left to the user to finally choose one of the available solutions. In order to enable an informed decision, it was already argued in [17] that alternative solutions in the search space are valuable in multiobjective optimization, even if one point in objective space has been selected. The same argument may be utilized for multi-local optimization, such that one wants to obtain the set of best solutions not only for finding the global optimum therein but also to have alternatives at hand when the seemingly best solution cannot be implemented [2, 15].

The remainder of this chapter is structured as follows. In the next section, we give an overview of previous work dealing with the topic of identifying several optima. Then, the two investigated basin identification methods are presented in section "Basin Identification Methods". In section "Sampling Algorithms", we give attention to sampling algorithms, which are needed to have something to select from in the first place, and explain the recent development maximin reconstruction [28, pp. 70–76]. Then, two experiments follow in section "Experiments". The first has the goal of finding regression models to predict favorable parameter settings for the basin identification methods. In the second one, the enhanced methods are tested on different optimization problems. The chapter ends with our conclusions in section "Conclusion".

Literature Report

According to Zentralblatt Math Database the article [29] by Antanas Žilinskas in 1978 was the first publication whose title contained the term *multimodal optimization*. Although not explicitly mentioned, the final goal in this work was the development of a method to locate the global optimum of a multimodal function. Thus, the term *multimodal optimization* in [29] described the task to find the global

optimum of a multimodal function; the detection of several local optima only was a side effect of the particular optimization strategy.

In the field of evolutionary computation the term *multimodal optimization* apparently appeared about one decade later (1987) for the first time in the title of a paper by Goldberg and Richardson [9]. In this paper the term is not defined explicitly, but it is clear that the authors aimed at developing a method to find the best peak of a multimodal function. For that purpose they introduced a mechanism gleaned from nature, in which the population of solutions is separated into niches which in turn should represent the basins of local optima. If such a mechanism is successful the population concentrates on the best local optima. Again, the detection of several local optima only was a side effect of the particular optimization strategy.

In 1993 Beasley et al. [5] proposed a niching strategy that explicitly aimed at locating all local optima of a multimodal function. Although they did not state that this task is the meaning of *multimodal optimization*, this paper may have had some bearing on the misconception of the authors (and reviewers!) of subsequent papers in bio-inspired optimization that the term *multimodal optimization* stands for finding all local optima. As can be extracted from Zentralblatt Math Database the semantic shift of the term began around 2000 and the new meaning was apparently established in 2011 in a survey paper on evolutionary multimodal optimization [7].

This process of an unnecessary semantic shift of a scientific term is the more remarkable, as the task of finding all local optima was explicitly termed *multi-local optimization* [12] in 1998 already. Actually, this notion can be traced back in the context of semi-infinite optimization at least to a PhD thesis [18] from the year 1992. Therefore we strongly support the initiation of a process of reversing the semantic shift and encourage the usage of the term *multi-local optimization* in future.

Basin Identification Methods

In this work, we consider the two basin identification methods: topographical selection and nearest-better clustering (NBC). The fitness criteria employed by the two are quite complementary: the former uses the number of nearest neighbors that are not better, the latter calculates the distance to the next better solution. So, both effectively utilize the same amount of information, namely the distances between solutions and their objective values.

Topographical Selection

Topographical selection (TS), as proposed by Törn and Viitanen [22], is described in Algorithm 3. It is designed to identify distinct local optima from a uniform sample of the search space. This works by building a directed graph containing for each point edges to its k nearest neighbors. The direction of the edges is always

Algorithm 3 Topographical selection

Input: points $\mathscr{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, number k of nearest neighbors
Output: nodes of the topograph with no outgoing edges

create a directed graph $G = (V, E)$ with $V = \{v_1, \ldots, v_N\}$ and $E = \emptyset$
for all $i \in \{1, \ldots, N\}$ **do**
  $J \leftarrow$ indices of the k nearest neighbors of $\mathbf{x}_i$
  for all $j \in J$ **do**
    if $f(\mathbf{x}_j) < f(\mathbf{x}_i)$ **then**
      $E \leftarrow E \cup \{(v_i, v_j)\}$ // add edge to graph
    else if using all edges, $f(\mathbf{x}_i) < f(\mathbf{x}_j)$ **then**
      $E \leftarrow E \cup \{(v_j, v_i)\}$ // add edge to graph
    end if
  end for
end for
return $\{v \in V \mid \deg^+(v) = 0\}$ // select nodes with no outgoing edges

pointing towards the better solution. (Note that we inverted the meaning of the edge direction in comparison to the original paper, to have a consistent definition of edges throughout all methods presented in this work.) The algorithm selects all points, whose nodes v in the graph have an outdegree $\deg^+(v) = 0$.

Törn and Viitanen [22] suggest to store the graph in a $(N \times k)$ matrix, with the indices to the k nearest neighbors in each row and using positive/negative signs to indicate the direction of the edges. They argue that it is less effort to only consider the respective row than to search the whole matrix for deciding if a point shall be selected. However, we would like to stress that by using adjacency lists as data structure to store the edges, this argument ceases to apply. We simply have to determine for each node if it has outgoing edges, which can be done in $O(1)$ with conventional adjacency lists. There also should be no other reason to restrict the topograph to "positive" edges, as Törn and Viitanen [23] already noted that differences between the two methods can be eliminated by appropriate choices of k.

Nearest-Better Clustering

NBC relies on the concept of the nearest-better neighbor

$$\mathrm{nbn}(\mathbf{x}, \mathscr{P}) = \arg\min_{\mathbf{y} \in \mathscr{P}} \{d(\mathbf{x}, \mathbf{y}) \mid f(\mathbf{y}) < f(\mathbf{x})\}$$

of a point $\mathbf{x}$, where $\mathscr{P}$ is typically a finite set of points. The distance to this point shall be denoted $d_{\mathrm{nb}}(\mathbf{x}, \mathscr{P}) := d(\mathbf{x}, \mathrm{nbn}(\mathbf{x}, \mathscr{P}))$. The conventional nearest-neighbor distance will be named $d_{\mathrm{nn}}(\mathbf{x}, \mathscr{P})$ in the following.

Algorithm 4 contains detailed pseudocode for NBC. In a first step, it creates a spanning tree consisting of edges from points to their nearest-better neighbors. Afterwards, the tree is divided into several connected components by removing

Algorithm 4 Nearest-better clustering

Input: points $\mathscr{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, parameter ϕ
Output: clusters in form of connected components of a graph
1: create a weighted, directed graph $G = (V, E)$ with $V = \{v_1, \ldots, v_N\}$ and $E = \emptyset$
2: **for all** $i \in \{1, \ldots, N\}$ **do**
3: **if** $\mathrm{nbn}(\mathbf{x}_i, \mathscr{P})$ exists **then**
4: $\mathbf{x}_j \leftarrow \mathrm{nbn}(\mathbf{x}_i, \mathscr{P})$
5: $e_i \leftarrow (v_i, v_j)$ // create edge
6: $w_{e_i} \leftarrow d(\mathbf{x}_i, \mathbf{x}_j)$ // set weight equal to distance
7: $E \leftarrow E \cup \{e_i\}$ // add edge to graph
8: **end if**
9: **end for** // G is now a spanning tree
10: $w_{\max} \leftarrow \phi \cdot d_{\mathrm{ref}}$ // calculate weight threshold for rule 1
11: $E' \leftarrow E$
12: let $e = (v_i, v_j)$
13: **if** $\deg^-(v_i) \geq 3$ **then**
14: let $e_1^-, \ldots, e_k^-$ be the incoming edges of v_i
15: **if** $w_e / \mathrm{median}\{w_{e_1^-}, \ldots, w_{e_k^-}\} > b$ **then**
16: $E \leftarrow E \setminus \{e\}$ // remove edge (rule 2)
17: **end if**
18: **end if**
19: **if** $w_e > w_{\max}$ **then**
20: $E \leftarrow E \setminus \{e\}$ // remove edge (rule 1)
21: **end if**
22: **return** G

"long" edges. The run time is governed by the quadratic number of distance computations necessary for building the graph.

For characterizing edges as long, two heuristics exist, which are called rule 1 and rule 2 in the pseudocode. Rule 1 simply removes all edges whose length exceeds the threshold $w_{\max}$. Rule 2 was added later. It is only applied to edges e whose tail v has an indegree $\deg^-(v) \geq 3$. The rule states to cut such an edge e if its length w_e is more than b times longer than the median of the incoming edges of v. The parameter b has been derived by extensive experimentation and is actually dependent on the number of points and the dimension [16, Sect. 4.5]:

$$b(N,n) = (-4.69 \cdot 10^{-4} n^2 + 0.0263n + 3.66n^{-1} - 0.457) \cdot \log_{10}(N) + 7.51 \cdot 10^{-4} n^2 - 0.0421n - 2.26n^{-1} + 1.83.$$

The aim of this involved rule 2 is to produce a correction yielding more clusters for large random uniform samples on highly multimodal functions, while not detecting more than 1.1 clusters on average in the case of unimodal functions [16, Sect. 4.5].

Historically, the reference distance d_{ref} in line 10 of Algorithm 4 was always calculated as the mean weight of all edges, i. e., $d_{\mathrm{ref,old}} = 1/|E| \sum_{i=1}^{|E|} w_{e_i}$. As these weights are simply nearest-better distances, the value of $d_{\mathrm{ref,old}}$ is dependent on the problem's number of optima. Investigations in [28, pp. 100–104] indicate that

the effect is opposite to what would be advisable, namely selecting the more points, the more optima exist. Thus we introduce $d_{\text{ref,new}} = 1/N \sum_{i=1}^{N} d_{\text{nn}}(\mathbf{x}_i, \mathscr{P})$ as an alternative definition of reference distance here. By only considering the conventional nearest-neighbor distances, the influence of the problem's landscape is eliminated.

The pruning rules are another issue. NBC with rule 1 alone can be used more flexibly than NBC with rules 1 and 2 (and TS), because one does not have to explicitly construct the graph G for applying rule 1. It is sufficient to calculate the key figure d_{nb} for each point and sort the sample accordingly. The threshold $w_{\max}$ then only determines how many points of this ranking are selected. If the graph is explicitly constructed, NBC also yields a hierarchical clustering of the points. This is possible because the initial graph is a spanning tree, which can be divided into several connected components by removing edges. We regard this flexibility as a conceptual advantage over TS.

Sampling Algorithms

As sampling algorithms, we regard simple random uniform sampling (SRS), generalized Halton sequences [8], and the maximin reconstruction (MmR) algorithm recently proposed in [28, pp. 70–76]. The core property of the used point samples for this application should be their uniformity [4, 22, 24]. While SRS provides some kind of baseline, Halton sequences asymptotically obtain a lower deviation from uniformity [8]. MmR explicitly optimizes the uniformity for the requested number of points, so it should offer the highest uniformity of the three.

Maximin Reconstruction

This algorithm is basically a variation of the "reconstruction algorithm" as described in [10, pp. 407–417]. The general idea of the reconstruction approach is to imitate a measured point process by minimizing the deviation of summary characteristics between the measured and the simulated point process. A common approach is to use a local search that exchanges one point per iteration and accepts the modification if the objective is improved. The references [19, 20] contain some more pointers to such exchange algorithms for experimental designs.

The pseudocode is outlined in Algorithm 5. The number of points is fixed in advance and the algorithm iteratively tries to replace one of the current points with a randomly chosen one. Instead of imitating an existing point set, we want to simply maximize uniformity. Thus, potentially existing fixed points are not taken as a reference set, but assumed to belong to the whole set, whose uniformity is to be maximized. Every improvement of the minimal distance between points in the set is accepted. In case no improvement is found, there is another extension of

Algorithm 5 Maximin reconstruction algorithm

Input: initial points $\mathscr{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, distance criterion $d(\cdot)$
Output: uniformly distributed points

$A \leftarrow \{1, \ldots, N\}$ // indices of candidates for replacement
$i \leftarrow$ random element of A // choose arbitrary candidate
$A \leftarrow A \setminus \{i\}$ // remove used index
repeat
  $\mathbf{y} \leftarrow$ random point in $\mathscr{X}$ // sample potential substitute
  if [**then**if improvement found]$d(\mathbf{y}) \geq d(\mathbf{x}_i)$
    $\mathbf{x}_i \leftarrow \mathbf{y}$ // replace the point in $\mathscr{P}$
    $A \leftarrow \{1, \ldots, N\} \setminus \{i\}$ // distances have changed, reset the available indices
  else if [**then**try to find point that is easier to replace]$A \neq \emptyset$
    $i' \leftarrow$ random element of A
    $A \leftarrow A \setminus \{i'\}$
    if [**then**if $\mathbf{x}_{i'}$ is easier to replace]$d(\mathbf{x}_{i'}) \leq d(\mathbf{x}_i)$
      $i \leftarrow i'$ // use it as new candidate for replacement
    end if
  end if
until termination
return $\mathscr{P}$

the basic algorithm. Instead of choosing the candidate for replacement randomly, we compare the current candidate with another one of the remaining, untested points in $\mathscr{P}$. The one with the smaller nearest-neighbor distance is chosen as the candidate for replacement in the next iteration. If the sequence of failed attempts is long enough, we will eventually find the point in $\mathscr{P}$ with the (currently) minimal nearest-neighbor distance, and replace it in one of the next iterations. Thus, we have a true maximin approach [11], and therefore the algorithm shall be called MmR. The attractiveness of the algorithm lies in its relatively economical use of distance computations without the need for a sophisticated data structure. Note, however, that we do not intend to produce the exact optimum, because we want to retain a rest of irregularity of the point set. If we disregarded this aspect, we would approach the topic of optimal sphere packing, for which certainly better local optimization algorithms could be devised for the last stage of optimization [1].

We still have to discuss the concrete definition of the distance criterion $d(\cdot)$ used in Algorithm 5. In its most basic form, $d(\mathbf{x}) = d_{\mathrm{nn}}(\mathbf{x}, \mathscr{Q})$, where $\mathscr{Q} = \mathscr{P}$, or, if additionally a set of fixed points $\mathscr{A}$ has to be considered, $\mathscr{Q} = \mathscr{P} \cup \mathscr{A}$. Unfortunately, maximin approaches are known for a drift towards the boundary [11], which means that the point density at the boundary is higher than in the interior [10, p. 145]. One possible remedy is to use periodic edge-correction (PEC) [10, p. 184]. For this purpose, an L_p torus distance

$$d_{\mathrm{torus}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} \min\{|x_i - y_i|, u_i - \ell_i - |x_i - y_i|\}^p \right)^{1/p}$$

is assumed as the internal distance in d_{nn}. In this case, the resulting sample is expected to be very uniform in the cuboid defined by the u_i and ℓ_i values, because edge effects are eliminated.

MmR's asymptotic run time is $O(t(N+|\mathscr{A}|)n)$ if t is the number of iterations and we want to generate N new points, while there are already $|\mathscr{A}|$ existing points, because the algorithm needs $N+|\mathscr{A}|$ distance computations in successful iterations and at most $2(N+|\mathscr{A}|)$ in unsuccessful iterations.

Experiments

Determining Regression Models

Research Question Which heuristics should be used to set the values of the methods' numerical parameters appropriately?

Pre-experimental Planning The existing literature [16, 22] and our preliminary experiments suggest that both basin identification methods are very dependent on their numerical parameters. Thus it seems ill-advised to compare the two methods with fixed parameter values. Instead we try to first find some regression models to predict an appropriate parameter value for a given problem. What complicates this experiment is that both methods have more than one nested factor, i. e., parameters that only apply to one of the two approaches. Thus we split the data analysis into two phases. First, we verify if our preferred categorial factor levels are competitive, then we try to determine the optimal numerical parameters for our favorite configurations.

Task As the identification of local optima is essentially a classification problem, we use the two simple measures precision and recall [14, p. 155] from the domain of information retrieval for evaluation. Recall has also become known as the *peak ratio* in the optimization community [21]. It uses the number of found optima

$$o = |\{\mathbf{x}^* \in \mathscr{O} \mid d_{\text{nn}}(\mathbf{x}^*, \mathscr{P}) \leq r\}|,$$

where $\mathscr{O}$ is the set of desired optima, $\mathscr{P}$ is the approximation set obtained by some algorithm, and r is a user-defined threshold parameter. The peak ratio is then $o/|\mathscr{O}|$, while precision is calculated as $o/|\mathscr{P}|$. We presume that precision is more important than peak ratio in our application, so in the first part of the analysis, the performance is assessed by a lexicographic ordering with these priorities. If there are several optimal configurations, we average the ϕ and k values over them. Linear regression models shall be fitted to the data of these optimal configurations by using ordinary least squares. The actual regression formula is chosen by manual experimentation considering the coefficient of determination R^2 and the model complexity as criteria.

Table 1 Factors for the experiment in section "Determining Regression Models"

Top-level factors	Type	Symbol	Levels
Problem topology	Non-observable		{random, funnel}
Number of local optima	Non-observable	ν	$\{5, 10, 20, 50\}$
Number of variables	Observable	n	$\{2, 3, 5, 10, 15, 20\}$
Number of points	Observable	N	$\{10n, 20n, 50n, 100n\}$
Sampling algorithm	Control		{SRS, GHalton, MmR}
Basin identification	Control		{TS, NBC}

Nested factors	Type	Symbol	Levels
Threshold factor (NBC)	Control	ϕ	$\{1.2, 1.3, 1.4, 1.5, 1.6, 1.8, 2.0, 2.25, 2.5, 2.75, 3.0\}$
Used rules (NBC)	Control		$\{\{1\}, \{1, 2\}\}$
Reference distance (NBC)	Control		$\{d_{\text{ref,old}}, d_{\text{ref,new}}\}$
Considered edges (TS)	Control		{all, positive}
Number of neighbors (TS)	Control	k	$\{1, 2, \ldots, 14, 16, 18, 20, 25, 30, 35, 40, 50\}$

Setup Table 1 shows the factors of our experiment. We randomly create artificial test instances with the generator from [27] to be able to control many problem features. Four environmental factors that are important for the problem difficulty [26] are varied. Only two of them are observable in the real world (n and N) so the regression models can use them as inputs. The other two are the number of local optima $\nu = |\mathcal{O}|$ and the global structure (topology) of the objective function. The number of points N is not added to the control factors, because we assume that we always employ the maximally possible budget, which would be limited through external circumstances. We further assume that we can sample the function ourselves, so the sampling algorithm is a controlled factor. This way, it is justified to build individual regression models for each sampling algorithm. To have some ground truth, we explicitly add all optima positions to the point samples. The remaining $N - \nu$ points are chosen by the respective sampling algorithm. For MmR we can explicitly consider the optima positions as fixed points in its distance calculations, so the overall distribution will be still very uniform. The generalized Halton sequence uses the implementation from https://github.com/fmder/ghalton, version 0.6. It is randomly initialized for every run. Note that the setup allows a few configurations with $N \leq \nu$; these are completely dismissed.

If a practitioner was confronted with a pre-built point set, it might be helpful to calculate the standard deviation of nearest-neighbor distances to get an idea which sampling the point set is similar to. The measure indicates the irregularity of the point set, which is somewhat antithetic to uniformity [28, pp. 47–48].

For each basin identification method, we carry out a full-factorial experiment over the top-level factors and its respective nested factors. There are 192 different configurations of environmental (observable and non-observable) factors. For each of them, we make 50 stochastic replications using common random numbers.

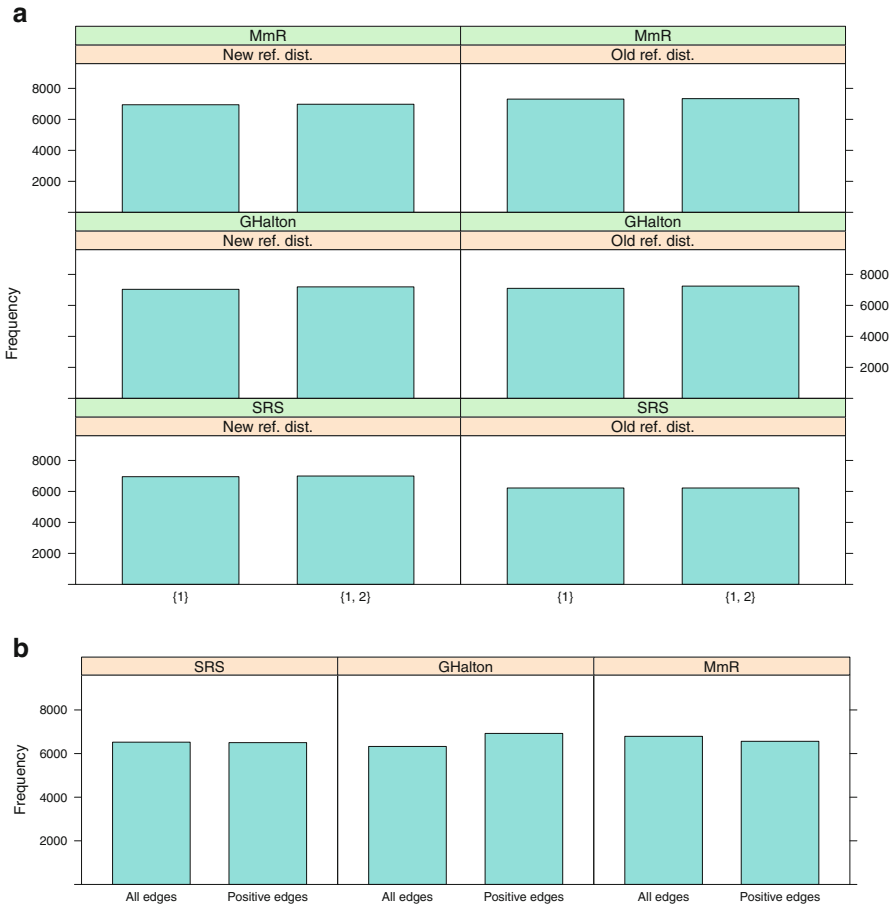

Fig. 1 Distribution of the best configurations on the categorial factors. (**a**) Nearest-better clustering. (**b**) Topographical selection

Results Figure 1 contains bar charts illustrating how often there is at least one optimal configuration for each categorial factor, depending on the sampling algorithm. Figure 2 shows the influence of n, N, and the irregularity of the sample on the optimal ϕ and k. The black regression lines in this figure are obtained by locally weighted scatterplot smoothing (LOESS). The same data was used to build the parametric regression models in Fig. 3, whose formulas and coefficient of determination R^2 are reproduced in Table 2.

Observations Figure 1 shows that for all configurations of categorial nested factors, there is at least one optimal setting of ϕ or k in the vast majority of the $192 \cdot 50 = 9600$ cases. So, the raw data does not privilege any configuration over the other. However, a preliminary analysis (not shown) indicates that the range of

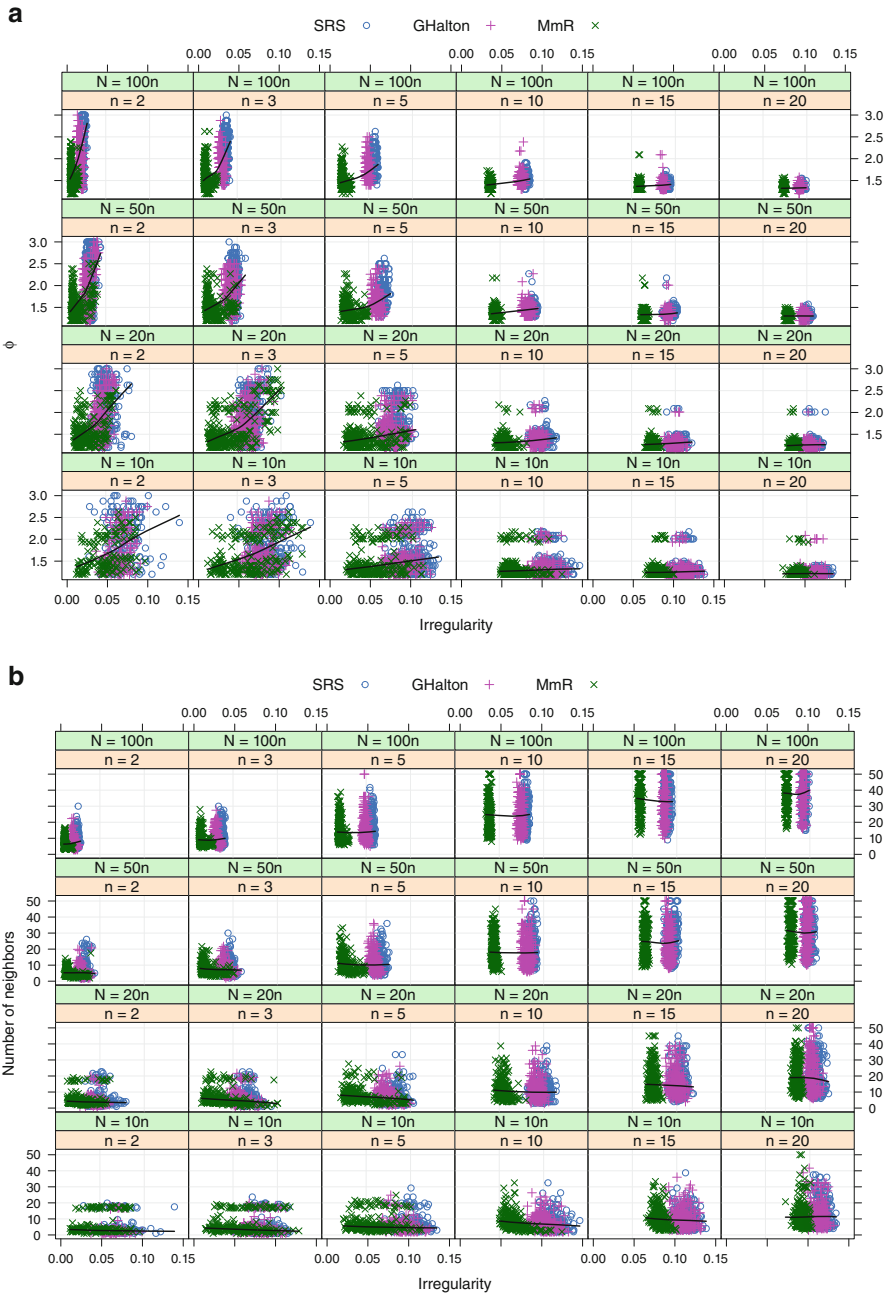

Fig. 2 Optimal values for ϕ and k depending on the dimension, the number of points, and the irregularity of the points sets. (**a**) Optimal edge-length factors for NBC. (**b**) Optimal numbers of neighbors for TS

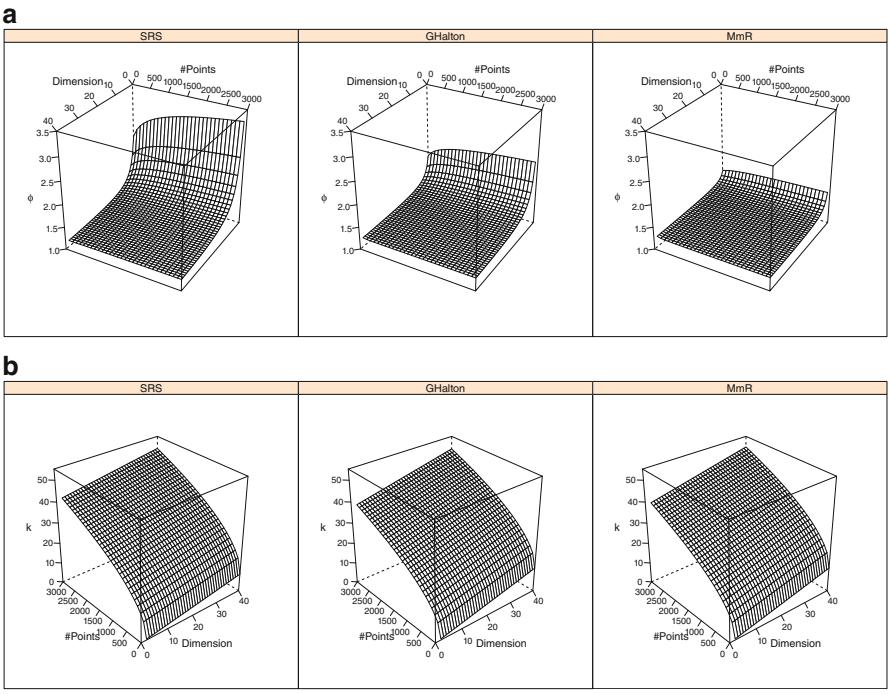

Fig. 3 Linear regression models on the data of Fig. 2. (**a**) Models for NBC. (**b**) Models for TS

Table 2 The formulas of the regression models in Fig. 3 and for the combined data sets

	NBC		TS	
Sampling	Formula	R^2	Formula	R^2
SRS	$1.144+2.288N^{-1}+0.541n^{-1}\ln N$	0.61	$0.215n+0.740N^{1/2}$	0.86
GHalton	$1.224+3.150N^{-1}+0.308n^{-1}\ln N$	0.42	$0.292n+0.680N^{1/2}$	0.86
MmR	$1.269+4.127N^{-1}+0.127n^{-1}\ln N$	0.18	$0.293n+0.697N^{1/2}$	0.86
All	$1.345+5.863N^{-1}+0.285n^{-1}\ln N$	0.20	$0.333n+0.777N^{1/2}$	0.77

admissible ϕ values is larger for NBC variants including rule 2, compared to those without rule 2. This is not surprising as the term $b(N,n)$ was fitted to experimental data in a very similar way as we do now for ϕ and k. But for the sake of brevity and in an attempt to compare the pure approaches first, we restrict the remainder of the analysis to the configurations we rated as more straightforward in section "Basin Identification Methods". This was NBC without rule 2, but with the new reference distance, and TS considering the topograph with all edges.

Under these circumstances, Fig. 2 shows that the optimal k is much less dependent on the sampling algorithm than the optimal ϕ, as indicated by the regression lines. This is confirmed by the fitted models, which are almost identical for TS,

while those for NBC differ considerably between the sampling algorithms, because the optimal ϕ grows with irregularity of the point sample (see Fig. 3). Also the R^2 of the NBC models becomes worse with increasing uniformity, which is not the case for TS models (see Table 2).

Another interesting observation are the two horizontal clusters appearing in the scatter plots of Fig. 2 when the number of points N is $20n$ or lower. For ϕ, they are more pronounced in higher dimensions, but for k they only appear in low dimensions. A closer inspection of the data (not shown) reveals that the clusters of higher ϕ and k values almost exclusively pertain to runs on funnel problems, while the lower clusters contain both topologies.

MmR obtains the least irregularity of all sampling algorithms on average, followed by the generalized Halton sequence and SRS. However, the variance is quite high, so the distributions of the three sampling algorithms are overlapping, especially in low dimensions.

Discussion The choice of k for TS seems to be relatively independent of the sampling algorithm. This might be a good sign, as it is desirable to have a robust basin identification method. For NBC, the number of points seems to have only little influence, especially if the point set is very regular. This may be a hint to prefer NBC for very large point sets. However, there remain the problem topology and the number of optima as problem properties which severely affect the prediction accuracy. So far, we have no means to treat their influence.

Validation

To validate the regression models from Table 2, we carry out another experiment using different random numbers and different values for ν and n than before. Table 3 shows the used configurations. The numerical parameters ϕ and k are now determined by the regression models. The integer k is obtained by rounding the model's response up. The optima are not included in the point sets in this experiment. Therefore, we now calculate the two performance measures from section "Determining Regression Models" with the number of basins defined as

$$o = \sum_{i=1}^{|\mathscr{O}|} \min \left\{ 1, \sum_{j=1}^{|\mathscr{P}|} b(\mathbf{x}_j, \mathbf{x}_i^*) \right\},$$

where

$$b(\mathbf{x}, \mathbf{x}^*) = \begin{cases} 1 & \text{if } \mathbf{x} \in \text{basin}(\mathbf{x}^*), \\ 0 & \text{else} \end{cases}$$

is an indicator for membership in the attraction basin of optimum $\mathbf{x}^*$, realized as described in [27].

Table 3 Factors for the validation experiment in section "Validation"

Top-level factors	Type	Symbol	Levels
Problem topology	Non-observable		{random, funnel}
Number of local optima	Non-observable	ν	$\{3, 12, 25, 75\}$
Number of variables	Observable	n	$\{2, 4, 8, 16, 24, 32\}$
Number of points	Observable	N	$\{10n, 20n, 50n, 100n\}$
Sampling algorithm	Control		{SRS, GHalton, MmR}
Basin identification	Control		{TS, NBC}

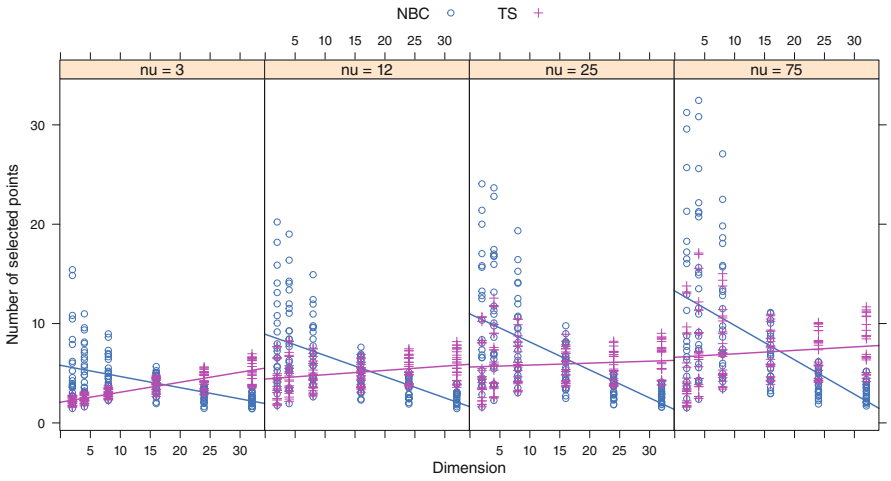

Fig. 4 The number of selected points for the tuned basin identification methods. Linear regression lines illustrate the general trend. NBC (*circles*) and TS (*plus signs*) show opposite behavior

With this setup we let NBC and TS again try to identify the basins. Figure 4 shows that the two methods react differently to a varying dimension, even though they are both using parameters optimized for the same criterion. NBC selects many points in low dimensions and reduces the number for growing n. TS shows exactly the opposite behavior. This difference between the two methods is also transferred over to the quality indicators precision and recall/basin ratio, as illustrated by Fig. 5. NBC obtains a lower precision and higher basin ratio in low dimensions, for TS the same can be said in high dimensions. We can also see that while the precision of TS is independent of the sampling algorithm, NBC scores the better, the lower the irregularity is.

The contrasting behavior of the two methods may be caused by the curse of dimensionality, which probably affects both methods differently. It is well known that the variation of Euclidean distances between pairs of uniformly distributed points vanishes with increasing dimension [6]. This probably has the effect that

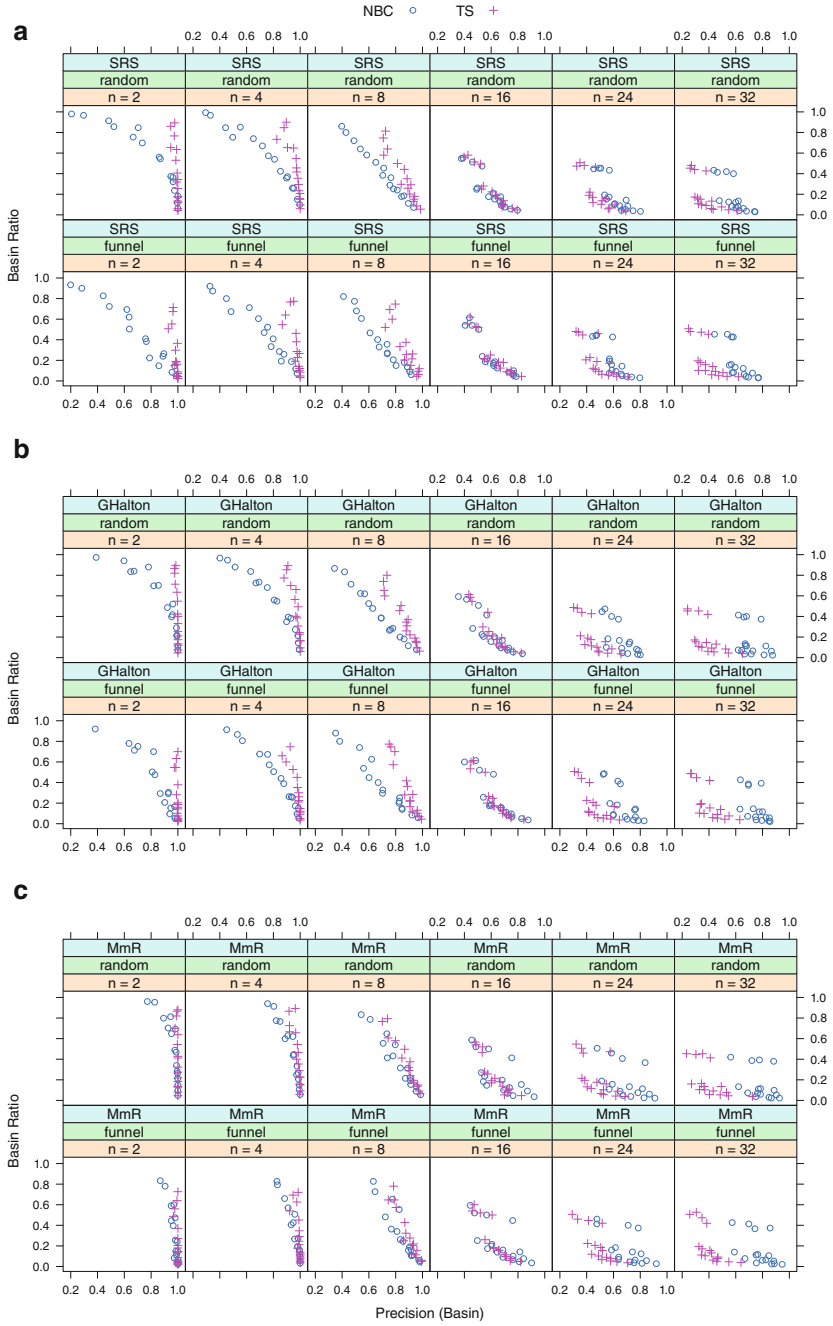

Fig. 5 Precision and basin ratio on the validation set. (**a**) Simple random sampling. (**b**) Generalized Halton. (**c**) Maximin reconstruction

it becomes increasingly unlikely for a nearest-better distance to exceed the given threshold. On the other hand, the k nearest neighbors of a point are probably a rather arbitrary choice in high dimensions, because the space is only sampled very sparsely anyway. So, it will often happen by chance that a point is selected by TS because none of its neighbors is better.

Conclusion

The basin identification methods' strong dependence on parameters is a nuisance. Some knowledge or at least a hypothesis about the problem's number of optima would be very helpful in determining the right parameter values. In absence of this knowledge, we can only try to employ the dimension of the space and the sample size as input to our prediction models, as done in this work. We admit that the resulting prediction models are still very rough and somewhat depend on the number of optima employed in the experiments. However, we think the relatively low values chosen by us for this parameter correspond to a realistic application scenario. In the future, it should be advisable to also incorporate features as the uniformity of the sample. A challenge in this regard is to find indicators yielding results that are comparable between different numbers of dimensions. Additionally, we repeat the recommendation to always use a sampling as uniform as possible, because this increases the precision of the basin identification. By using a more expensive sampling algorithm, such as maximin reconstruction, it is possible to further improve the performance as against quasirandom sequences. If it is not possible to control the sampling, a basin identification method robust to outliers must be sought. TS already seems to meet this demand. It may also be beneficial to somehow combine the concepts of NBC and TS into one method. Using Manhattan (or maybe even non-metric) distances also might be a simple way to further improve the stability of both methods [3].

One also has to consider that in global optimization algorithms, where basin identification methods are typically employed, it may be advisable to carry out at least a minimum number of local searches per iteration, in recognition of the uncertainty generally associated with the basin identification. In this case, the decision how many points to select would not be left completely to the basin identification component. This can be realized easy by using NBC with rule 1, which can also provide a ranking of the sample, from which a decision maker can select an arbitrary number of points according to his preference. In the case of TS, the number of neighbors would have to be adapted iteratively to approximate a given number of points to select, which is more cumbersome.

References

1. Addis, B., Locatelli, M., Schoen, F.: Disk packing in a square: a new global optimization approach. INFORMS J. Comput. **20**(4), 516–524 (2008)
2. Addis, B., Cassioli, A., Locatelli, M., Schoen, F.: A global optimization method for the design of space trajectories. Comput. Optim. Appl. **48**(3), 635–652 (2011)
3. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) Database Theory — ICDT. Lecture Notes in Computer Science, vol. 1973, pp. 420–434. Springer, Berlin, Heidelberg (2001)
4. Ali, M.M., Storey, C.: Topographical multilevel single linkage. J. Global Optim. **5**(4), 349–358 (1994)
5. Beasley, D., Bull, D.R., Martin, R.R.: A sequential niche technique for multimodal function optimization. Evol. Comput. **1**(2), 101–125 (1993)
6. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Buneman, P. (eds.) Database Theory — ICDT'99. Lecture Notes in Computer Science, vol. 1540, pp. 217–235. Springer, Berlin, Heidelberg (1999)
7. Das, S., Maity, S., Qu, B.Y., Suganthan, P.N.: Real-parameter evolutionary multimodal optimization – a survey of the state-of-the-art. Swarm and Evol. Comput. **1**(2), 71–88 (2011)
8. De Rainville, F.M., Gagné, C., Teytaud, O., Laurendeau, D.: Evolutionary optimization of low-discrepancy sequences. ACM Trans. Model. Comput. Simul. **22**(2), 9:1–9:25 (2012)
9. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms and Their Application, pp. 41–49. Lawrence Erlbaum Associates, Inc., London (1987)
10. Illian, J., Penttinen, A., Stoyan, H., Stoyan, D.: Statistical Analysis and Modelling of Spatial Point Patterns. Wiley, New York (2008)
11. Johnson, M.E., Moore, L.M., Ylvisaker, D.: Minimax and maximin distance designs. J. Stat. Plan. Infer. **26**(2), 131–148 (1990)
12. León, T., Sanmatías, S., Vercher, E.: A multi-local optimization algorithm. Top **6**(1), 1–18 (1998)
13. López Redondo, J., Martínez Ortigosa, P., Žilinskas, J.: Multimodal evolutionary algorithm for multidimensional scaling with city-block distances. Informatica **23**(4), 601–620 (2012)
14. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
15. Parmee, I.C.: The maintenance of search diversity for effective design space decomposition using cluster-oriented genetic algorithms (COGA) and multi-agent strategies (GAANT). In: Proceedings of 2nd International Conference on Adaptive Computing in Engineering Design and Control, ACEDC '96, pp. 128–138. University of Plymouth, Plymouth (1996)
16. Preuss, M.: Multimodal Optimization by Means of Evolutionary Algorithms. Springer, Berlin (2015)
17. Preuss, M., Naujoks, B., Rudolph, G.: Pareto set and EMOA behavior for simple multimodal multiobjective functions. In: Runarsson, T., Beyer, H.G., Burke, E., Merelo-Guervós, J., Whitley, L., Yao, X. (eds.) Parallel Problem Solving from Nature - PPSN IX. Lecture Notes in Computer Science, vol. 4193, pp. 513–522. Springer, Berlin (2006)
18. Price, C.J.: Non-linear semi-infinite programming. Ph.D. thesis, University of Canterbury, Department of Mathematics, Christchurch, New Zealand (1992)
19. Pronzato, L., Müller, W.G.: Design of computer experiments: space filling and beyond. Stat. Comput. **22**(3), 681–701 (2012)
20. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
21. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1382–1389 (2004)

22. Törn, A., Viitanen, S.: Topographical global optimization. In: Floudas, C.A., Pardalos, P.M. (eds.) Recent Advances in Global Optimization. Princeton Series in Computer Sciences, pp. 384–398. Princeton University Press, Princeton (1992)
23. Törn, A., Viitanen, S.: Topographical global optimization using pre-sampled points. J. Global Optim. **5**(3), 267–276 (1994)
24. Törn, A., Viitanen, S.: Iterative topographical global optimization. In: Floudas, C., Pardalos, P. (eds.) State of the Art in Global Optimization. Nonconvex Optimization and Its Applications, vol. 7, pp. 353–363. Springer, Berlin (1996)
25. Törn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989)
26. Törn, A., Ali, M.M., Viitanen, S.: Stochastic global optimization: problem classes and solution techniques. J. Global Optim. **14**(4), 437–447 (1999)
27. Wessing, S.: The multiple peaks model 2. Algorithm Engineering Report TR15-2-001, Technische Universität Dortmund (2015). https://ls11-www.cs.uni-dortmund.de/_media/techreports/tr15-01.pdf
28. Wessing, S.: Two-stage methods for multimodal optimization. Ph.D. thesis, Technische Universität Dortmund (2015). http://hdl.handle.net/2003/34148
29. Žilinskas, A.: On statistical models for multimodal optimization. Mathematische Operationsforschung und Statistik. Series Statistics **9**, 255–266 (1978)

Part II
Applications of Global Optimization

Cloud Computing Approach for Intelligent Visualization of Multidimensional Data

Jolita Bernatavičienė, Gintautas Dzemyda, Olga Kurasova, Virginijus Marcinkevičius, Viktor Medvedev, and Povilas Treigys

Abstract In this paper, a Cloud computing approach for intelligent visualization of multidimensional data is proposed. Intelligent visualization enables to create visualization models based on the best practices and experience. A new Cloud computing-based data mining system DAMIS is introduced for the intelligent data analysis including data visualization methods. It can assist researchers to handle large amounts of multidimensional data when executing resource-expensive and time-consuming data mining tasks by considerably reducing the information load. The application of DAMIS is illustrated by the visual analysis of medical streaming data.

Keywords intelligent visualization • cloud computing • dimensionality reduction • medical streaming data

Introduction

One of the significant steps in the process of knowledge discovery in datasets is multidimensional data visualization. The visualization is grounded on the idea to present data in such a visual form that allows to gain insight of the data, and to influence a further process of data mining and decision making [1, 11]. The perception and extraction of useful information from the graphical representation of data is much easier than it could be done from raw domain data. It is the benefit of visualization. Ability to visualize multidimensional data is one of the basic functionalities for the data analysis tools. Despite the fact that most of the multidimensional data analysis methods were proposed a few decades ago, they remain popular and are continually improved in order to create more effective ways of dimensionality reduction-based visualization. Another direction of researches is a

J. Bernatavičienė • G. Dzemyda • O. Kurasova (✉) • V. Marcinkevičius
V. Medvedev • P. Treigys
Institute of Mathematics and Informatics, Vilnius University, Akademijos str. 4, LT-08663 Vilnius, Lithuania
e-mail: olga.kurasova@mii.vu.lt

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_5

software development for multidimensional data visualization. Recent technological possibilities of Cloud computing enable to speed up time-consuming data analysis. However, convenient, powerful, and user-friendly tools for such an analysis are still missing. The goal of this paper is to introduce a Cloud computing approach for intelligent visualization of multidimensional data. As noted in [3], the intelligent analysis is a carefully planned and considered process of deciding what will be most useful and revealing. This approach is illustrated by the analysis of medical streaming data.

Multidimensional Data Visualization

In this paper, we focus on visualization methods that reduce data dimensionality from the original high-dimensional space to the target dimension (2D in the visualization case). The goal of dimensionality reduction is to represent the input dataset in a lower-dimensional space so that certain properties (e.g., clusters and outliers) of the structure of this dataset were preserved as faithfully as possible. Another reason for reducing the dimensionality is to reduce the computational load for further data processing. Today's big multidimensional datasets contain a huge amount of data so that it becomes almost impossible to analyze them by conventional ways with a view to extract valuable information. We require more effective ways to display, analyze, and interpret the information contained within them.

The data from the real world can usually be described by an array of features $x_1, x_2, \ldots, x_n$. A combination of values of all features characterizes a particular data object $X_j = (x_{j1}, x_{j2}, \ldots, x_{jn}), j \in \{1, \ldots, m\}$, from the whole set $X_1, X_2, \ldots, X_m$, where n is the number of features and m is the number of analyzed objects. If $X_1, X_2, \ldots, X_m$ are described by more than one feature, the data are called multidimensional data, if n is large enough, then the data are called big multidimensional data. Often X_j, $j = 1, \ldots, m$, are interpreted as points in the multidimensional space.

Visualization Methods Based on Dimensionality Reduction

Several approaches have been proposed for transforming nonlinear high-dimensional structures into a lower-dimensional space. A comprehensive review of the dimensionality reduction-based visualization methods is presented in [8, 11].

Principal component analysis (PCA) [14] is a well-known method for dimensionality reduction. It can be used to display the data as a linear projection in a subspace of the original data space so that it preserves the variance of the data at best. However, the interpretation of the principal components can be difficult at times. Moreover, PCA cannot embrace nonlinear structures, consisting of arbitrarily shaped clusters or curved manifolds, since it describes the data in terms of a linear subspace.

An alternative approach to dimensionality reduction is multidimensional scaling (MDS) [5]. It is a classical approach which maps the original high-dimensional space to a lower-dimensional one by using the information on the distances between the objects in the original space so that the distances of the corresponding data points in a lower-dimensional space are preserved. Thus, the so-called Stress function (projection error) is minimized. There exists a multitude of variants of MDS with different Stress functions and their optimization algorithms [5, 24–26]. Commonly MDS Stress function is minimized using the SMACOF algorithm based on iterative majorization. It is a popular optimization algorithm suitable for solving such type of minimization problems. The method is simple and powerful, because it guarantees a monotonic convergence of the Stress function. Diagonal majorization algorithm (DMA) is a modification of the SMACOF algorithm. DMA attains a slightly worse MDS projection error than SMACOF, but computations are faster and require less computer memory resources. Therefore, the DMA algorithm can be used for visualizing big datasets. It is worth to mention that MDS does not offer a possibility to project new and previously unseen points on the existing set of the mapped points. To get a mapping that presents the previously mapped points together with the new ones requires a complete re-run of the MDS algorithm on the newly formed set of new and mapped data points. The relative MDS can be used for visualizing the new data points on the fixed mapping as well as for visualizing big datasets. The relative MDS algorithm gives a precise mapping and saves much computing time as compared with the MDS algorithm. Other local and global optimization methods for minimizing MDS Stress are investigated and discussed in [11, 24–26].

Artificial neural networks may also be used for multidimensional data visualization. Several neural network-based methods for visualizing big multidimensional datasets have been proposed, including SAMANN [19, 21] and SOM [15]. Mao and Jain have proposed in [19] a neural network implementation for multidimensional data visualization. The back propagation-like learning rule SAMANN has been developed to allow a feed-forward artificial neural network to learn MDS-based Sammon's mapping in an unsupervised way. After training the neural network is able to project previously unseen points using the obtained generalized mapping rule. It has been concluded in [19] that the SAMANN network preserves the data structure, cluster shape, and interpattern distances better than a number of other multidimensional visualization methods.

Self-organizing map (SOM) is another type of neural networks suitable for multidimensional data visualization [15]. It is also used for data clustering. SOM is a set of neurons, connected one to another via a rectangular or hexagonal topology. Each neuron is defined by its place in SOM and by the so-called codebook vectors. At each learning step, a multidimensional point X_j, $j = \{1, \ldots, m\}$, is presented to the neural network. The codebook vectors of neurons are adapted according to the defined learning rule. The neuron, whose the codebook vector is with the minimal Euclidean distance to X_j, is designated as a winner. After SOM learning phase, the analyzed data $X_1, X_2, \ldots, X_m$ are presented to SOM and winning neurons are found for each object. In such a way, the objects are distributed on SOM and some data clusters can be observed. Besides, according to the position on the grid, the neurons

are characterized by n-dimensional codebook vectors. An intuitive idea comes to apply the dimensionality reduction methods to additional mapping of the codebook vectors of the winning neurons on the plane. The MDS method may be used to this end. Moreover, the number of winning neurons is smaller than the number of data points, so a smaller dataset should be visualized by MDS than in the case where the whole dataset is analyzed by MDS. The ways of combining SOM and MDS have been proposed and investigated in [7, 18].

Cloud Computing Tools for Multidimensional Data Visualization

Over the years data mining software has been developed with the view to facilitate solving data mining problems such as classification, clustering, and prediction. Many of the solutions are open sourced and available for free, therefore they have become very popular among researchers. Usually only PCA, MDS, and SOM are implemented in the open source data mining tools: WEKA [12], Orange [6], KNIME [4], RapidMiner [13], and R. Commercial statistics software, such as Statistica (StatSoft), SAS/STAT, IBM SPSS Modeler (Clementine) as well as MATLAB, have possibilities to make data analysis by PCA and MDS.

Cloud-based technologies become available to assist in creating scalable, extensible, interoperable, modular, and easy-to-use data mining tools and to apply them not only in solving data mining problems, but also in intelligent visualization of multidimensional data. The new sophisticated instrument, the so-called scientific workflow, can be used to form and execute a series of data analysis and computation procedures in scientific application. The usage of scientific workflows allows researchers to compose convenient platforms for experiments by retrieving data from databases and data warehouses and running data mining algorithms in the Cloud infrastructure. Another way to make the visualization intelligent is to use a web service, which describes a set of packed functions with a single public interface. If this public interface is published in the network, it might be used by other applications through a standard protocol. Such approach allows to integrate heterogeneous platforms and applications while keeping it simple to the end user.

Orange4WS [22] is an extension of the well-known data mining toolbox Orange [6], in which some new features, including web services, are implemented. Other open source tools for data mining are provided by KNIME Analytics Platform [4]. KNIME also provides commercial extensions such as Cluster Execution and Big Data Extensions in order to increase the performance by integrating the advanced hardware and infrastructure capabilities.

ClowdFlows is an open source Cloud-based web application platform for composition, execution, and sharing of interactive machine learning and data mining workflows [16, 17]. A part of data mining algorithms from Orange and WEKA is implemented as local services. DAME (DAta Mining and Exploration) is an innovative web-based, distributed data mining infrastructure, specialized in exploration of big datasets by various data mining methods [20]. DAME provides

a user-friendly and standardized scientific gateway to ease the access, exploration, processing, and understanding of big datasets.

From the point of view of intelligent visualization, the aforementioned data mining tools have the following drawbacks:

- There is a lack of intelligent visualization approaches. It is useful to have a wider set of implemented multidimensional data visualization methods of various nature and additional utilities collected in one toolbox. The set of implementations, organized in such a way, would allow to get the diverse knowledge on the research object.
- Most of the data mining tools are not web applications and there is no possibility to use them directly from the web. The advantage of web applications is that they are used and controlled by a web browser integrating advantages provided by Cloud technologies.
- There are no user-friendly instruments to exploit benefits provided by Cloud computing for solving time- and resource-consuming visualization problems without specific knowledge on parallel and distributed computing.

DAMIS: Implementation of Intelligent Visualization

To create an approach for intelligent visualization of multidimensional data with intention to avoid drawbacks of the existing data mining tools, a new Cloud-based web application, called DAMIS (DAta MIning System), is being developed. The initial idea for visualization of large-scale multidimensional data, using web service technologies, has been proposed in [9, 10]. The open source DAMIS software (http://www.damis.lt) implements data mining solution as a service for the end user and has a graphical user-friendly interface which allows researchers to carry out data analysis tasks, to visualize multidimensional data, to investigate multidimensional data projections and data item similarities as well as to identify the influence of individual features and their relationships by applying various data mining algorithms and taking the advantage of Cloud computing. DAMIS is also available in the Lithuanian national open access scientific information archive MIDAS (https://www.midas.lt) and serves as the data analysis toolbox for MIDAS users.

The components of DAMIS and the relations between them are presented in Fig. 1. The relation between the implemented data mining algorithms and the graphical user interface is assured by usage of SOAP protocol-based web service. The algorithms for multidimensional data preprocessing, clustering, classification, and dimensionality reduction-based visualization have been implemented. To analyze the data by the implemented data mining algorithms, the user initializes and manages an experiment by constructing scientific workflows, i.e. the order in which the data mining algorithms are executed. The user can modify the created workflow by adding or removing nodes and reuse the created workflow for other data analysis.

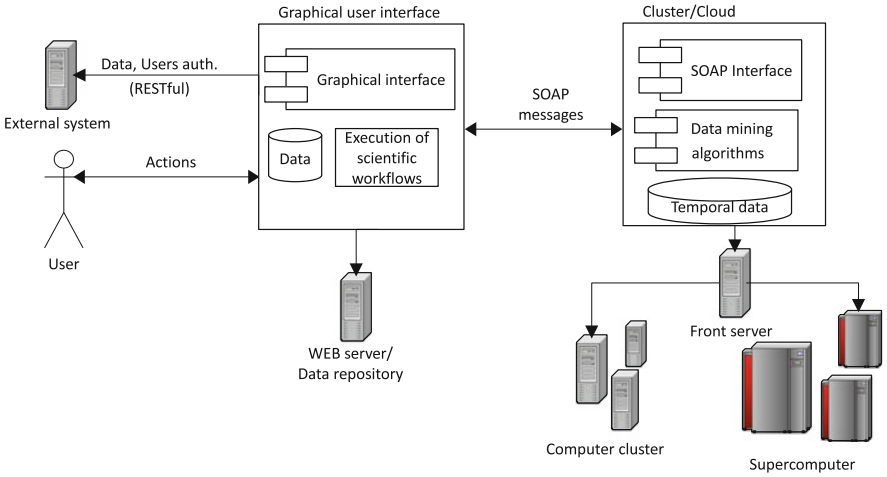

Fig. 1 DAMIS architecture

DAMIS provides Cloud computing solutions useful to solve complicated and time-consuming data mining problems. For big data analysis the user can select computing resources from the proposed alternatives and utilize them on demand. With the view to ensure the mobility and the independence of workplaces of researchers, a data repository on the web is available. After uploading data files, they become a part of the data web repository, and the user has a possibility to manage these files and perform different experiments with the same data. The results of data mining can be saved either in a user's computer or in other external system such as the Lithuanian national open access scientific information archive MIDAS. DAMIS gains an advantage over the competitive data mining tools and can be an attractive alternative for users involved in data mining.

The proposed data mining system aggregates a set of the methods for data preprocessing, clustering, classification, and visualization. The following dimensionality reduction-based methods for multidimensional data visualization are implemented:

- Principal component analysis (PCA),
- Multidimensional scaling (SMACOF-MDS),
- Diagonal majorization algorithm (DMA),
- Relative MDS,
- Artificial neural network for Sammon's mapping (SAMANN),
- Combination of the self-organizing map with multidimensional scaling (SOM-MDS).

Thus, DAMIS is a tool for intelligent visualization, which can help researchers to considerably reduce the information load and to handle large amounts of multidimensional data when solving time-consuming and resource-expensive data

mining tasks. Such an intelligent visualization allows to look inside the data and to create the visualization model, based on the best practices and recommendations.

DAMIS Application for Visual Analysis of Medical Streaming Data

The range of applications of DAMIS is wide and this system can be useful for researchers of various scientific fields. For the purpose to illustrate the DAMIS system functionality to visualize multidimensional data, the application for visual analysis of medical streaming data is presented. The similarity search in multivariate physiological time series which consists of medical observations over a period of time is investigated. Data are collected using sensors by recording some predefined personal medical features. The goal of the research is to compare a subsequence of the multivariate time series, that corresponds to the current health state of a patient, with chronologically collected historical data and to find the most similar one [2].

The data from the PhysioNet/Computing in Cardiology Challenge are used in this research (http://www.physionet.org/challenge/2012/). All the records were collected in the intensive care unit. It was selected a set Y^a, containing multivariate time series of 50 patients of the same age. Thus, we have 50 different multivariate time series $Y_i^a, i = 1, \ldots, 50$, each of them consists of 48 observations ($T^a = 50 \times 48$) of $p = 4$ features: non-invasive diastolic arterial blood pressure, non-invasive systolic arterial blood pressure, heart rate, and temperature, $Y^a = \{Y_i^a, i = 1, \ldots, 50\}$. The analyzed dataset can be downloaded from the open access scientific information archive MIDAS: http://dx.doi.org/10.18279/MIDAS.duomenys.zip.1814.

To detect events in multivariate time series, it is necessary to compare time series using the appropriate similarity measure. Different techniques and similarity measures are introduced [23] and used for comparison of multivariate time series of different nature. In this research, five similarity measures are used: Frobenius norm, Matrix Correlation Coefficient, PCA similarity factor, multidimensional dynamic time warping (MDTW), and Extended Frobenius norm (EROS). All these similarity measures are described in detail in [2].

Let us have:

- a multivariate time series Y^a of p features and T^a observations;
- a sample Y^b of p features and T^b observations—a subsequence of the multivariate time series—which corresponds to the current health state of a patient; and
- n similarity measures which will be used for comparison of subsequences.

The procedure which prepares multidimensional data from multivariate physiological time series for visual analysis can be generalized as follows [2]:

1. The sample Y^b is compared with all the subsequences of Y^a by using m similarity measures. The subsequences are obtained by moving the time window in Y^a from beginning to end. The content of such a window is a matrix of n rows. Denote the

matrix by Y^w. The width of the window (the number of columns of Y^w) is adapted to the sample Y^b, its width is equal to T^b. For each measure, k subsequences are chosen most similar to the sample. Therefore, the total number of subsequences for a further analysis is equal to kn.

2. Each comparison of the sample with a subsequence, chosen in the way defined in the step above, produces an n-dimensional point $X_s = (x_{s1}, x_{s2}, \ldots, x_{sn})$, where $s = 1, \ldots, kn$. Let us derive two additional points: $X_0 = (x_{01}, x_{02}, \ldots, x_{0n})$ is the array of values of all similarity measures, computed for the subsequence, that is ideally coincident with the sample (the array of the best values of n similarity measures); $X_c = (x_{c1}, x_{c2}, \ldots, x_{cn})$ is the weight center of $X_s, s = 1, \ldots, kn$. Therefore, the total number of n-dimensional points for discovering subsequences most similar to the sample is $m = kn + 2$.
3. Each point X_s is marked by a class label that indicates the similarity measure according to which the subsequence falls among the most similar ones. The class label of the point X_c is marked as "weight center," and that of the point X_0—"ideally coincident."

After such data preparation, the obtained multidimensional points, which correspond to different comparisons of the sample with other subsequences, are analyzed using the DAMIS software. In accordance with each of the 5 similarity measures, 10 subsequences, most similar to the sample, are chosen for a further analysis, $k = 10$. Therefore, the total number of X_s subsequences is equal to 50, $s = 1, \ldots, 50$. With the two additional points X_0 and X_c, the total number of five-dimensional points is $m = 52$. Thus, the whole set of multidimensional data is $X = \{X_1, X_2, \ldots, X_{50}, X_0, X_c\}$.

A scientific workflow for the medical data mining, which includes visualization, is formed and presented in Fig. 2. The workflow consists of the connected nodes. Each node corresponds to either data preprocessing or data mining algorithm. The nodes for data file uploading and viewing the results are also used for workflow construction.

The multidimensional points are normalized by z-score and visualized on the plane using PCA (Fig. 3), MDS (Fig. 4), and SAMANN (Fig. 5). A point on the plane represents projection result of a five-dimensional point. It depicts the comparison result of the sample with the particular subsequence by five similarity measures. In total there are 52 points on the plane. Each point is colored according to the class it represents.

It is assumed that the subsequence, the corresponding point of which on a plane is closest to the projection of X_0, is most similar to the sample X_0. Moreover, the subsequences are assumed to be similar to the sample, if the Euclidean distance

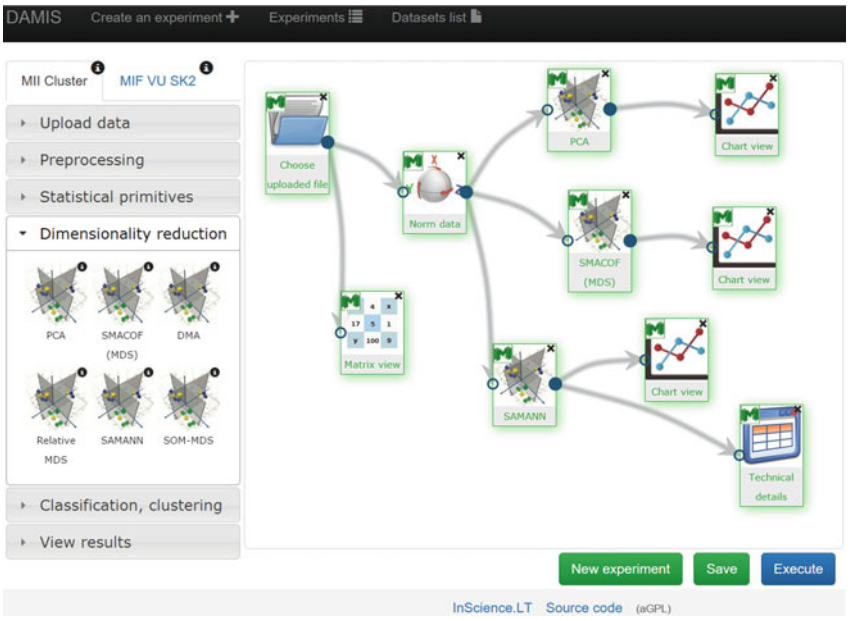

Fig. 2 DAMIS scientific workflow for medical data mining

between the projections, corresponding to such subsequences, and the projection of X_0 is less than the distance between the projections of X_0 and X_C. In each of Figs. 3, 4, and 5, there is a part of the red, green, and yellow points, the distances of which to the blue point (X_0) are smaller than the distance between the blue (X_0) and violent (X_C) points. In the case of MDS (Fig. 4), this part of points is framed. It may state that the subsequences, corresponding to these points, are most similar to the sample considering a combination of all the five similarity measures. Most of the points in the frame are yellow, and there are no light blue points in Fig. 4. It means that the PCA similarity factor demonstrates the worst results, while the Extended Frobenius norm (EROS) shows the best ones. Such coloring allows to compare the different similarity measures and disclose which are more effective.

The results obtained by different visualization methods and presented in Figs. 3, 4, and 5 are quite similar; however, some differences can be observed. Such information can be useful for researchers with a view to gain a deeper insight into the analyzed data and to evaluate the similarity of subsequences with the selected sample from different standpoints.

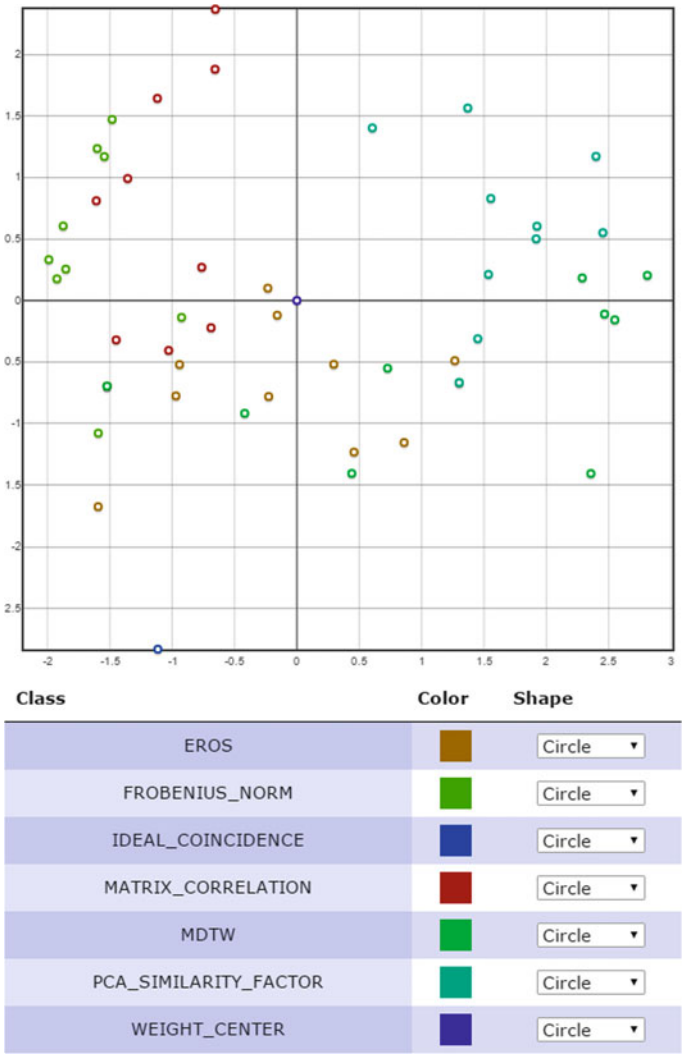

Fig. 3 Visualization results of the medical streaming data, obtained by PCA

Conclusions

The Cloud computing ideas and concepts may be successfully spread to intelligent visualization of multidimensional data. With the view to assist researchers by considerably reducing the information load and to allow them to handle large amounts of multidimensional data, when solving time-consuming tasks, a Cloud

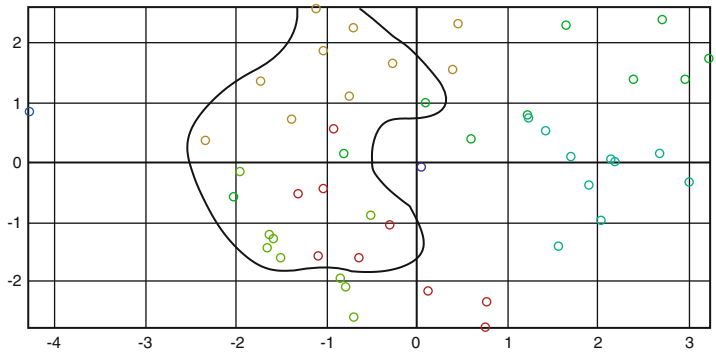

Fig. 4 Visualization results of the medical streaming data, obtained by MDS

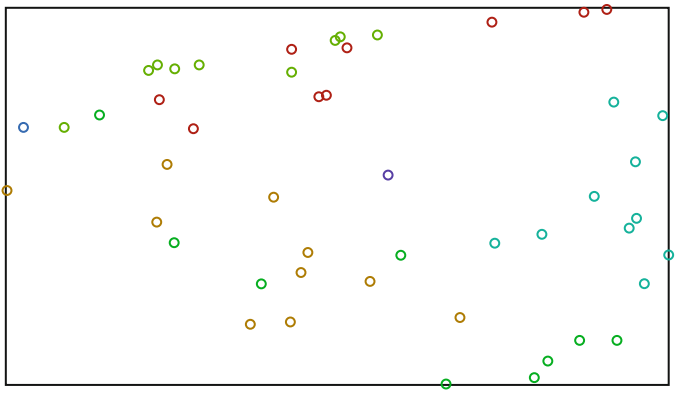

Fig. 5 Visualization results of the medical streaming data, obtained by SAMANN

computing-based data mining system DAMIS has been developed. The system puts emphasis on the service for the intelligent visualization. The researcher can choose the desired computing resources and utilize them on demand. One more advantage is that its interface meets the scientific workflow construction principle.

The visual analysis of medical streaming data has been made by applying the Cloud computing-based intelligent visualization. As a result, the possibility to visually compare a subsequence of the multivariate time series is disclosed, which corresponds to the current health state of a patient, with chronologically collected historical data. Such a visual analysis allows to detect several subsequences in the time series, similar to the current health state in accordance with several (in our case—five) measures what leads to a better final decision.

References

1. Bernatavičienė, J., Dzemyda, G., Kurasova, O., Marcinkevičius, V., Medvedev, V.: The problem of visual analysis of multidimensional medical data. In: Törn, A., Žilinskas, J. (eds.) Models and Algorithms for Global Optimization. Optimization and Its Applications, vol. 4, pp. 277–298. Springer, New York (2007). doi:10.1007/978-0-387-36721-7_17
2. Bernatavičienė, J., Dzemyda, G., Bazilevičius, G., Medvedev, V., Marcinkevičius, V., Treigys, P.: Method for visual detection of similarities in medical streaming data. Int. J. Comput. Commun. Control **10**(1), 8–21 (2015). doi:10.15837/ijccc.2015.1.1310
3. Berthold, M.R., Hand, D.J. (eds.): Intelligent Data Analysis: An Introduction, 2nd edn. Springer, Berlin (2003). doi:10.1007/978-3-540-48625-1
4. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz information Miner. In: Studies in Classification, Data Analysis, and Knowledge Organization. Springer, Berlin (2007). doi: 10.1007/978-3-540-78246-9_38
5. Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer, New York (2005). doi:10.1007/0-387-28981-X
6. Demšar, J., Curk, T., Erjavec, A., Gorup, C., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B.: Orange: data mining toolbox in Python. J. Mach. Learn. Res. **14**, 2349–2353 (2013)
7. Dzemyda, G., Kurasova, O.: Heuristic approach for minimizing the projection error in the integrated mapping. Eur. J. Oper. Res. **171**(3), 859–878 (2006). doi:10.1016/j.ejor.2004.09.011
8. Dzemyda, G., Kurasova, O., Medvedev, V.: Dimension reduction and data visualization using neural networks. In: Maglogiannis, I., Karpouzis, K., Wallace, M., Soldatos, J. (eds.) Emerging Artificial Intelligence Applications in Computer Engineering. Frontiers in Artificial Intelligence and Applications, vol. 160, pp. 25–49. IOS Press, Amsterdam (2007)
9. Dzemyda, G., Marcinkevičius, V., Medvedev, V.: Large-scale multidimensional data visualization: a web service for data mining. In: Abramowicz, W., Llorente, I., Surridge, M., Zisman, A., Vayssière, J. (eds.) Towards a Service-Based Internet. Lecture Notes in Computer Science, vol. 6994, pp. 14–25. Springer, Berlin/Heidelberg (2011). doi:10.1007/978-3-642-24755-2_2
10. Dzemyda, G., Marcinkevičius, V., Medvedev, V.: Web application for large-scale multidimensional data visualization. Math. Model. Anal. **16**(2), 273–285 (2011). doi:10.3846/13926292.2011.580381
11. Dzemyda, G., Kurasova, O., Žilinskas, J.: Multidimensional Data Visualization: Methods and Applications. Springer, Berlin (2013). doi:10.1007/978-1-4419-0236-8
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). doi:10.1145/1656274.1656278
13. Hofmann, M., Klinkenberg, R.: RapidMiner: Data Mining Use Cases and Business Analytics Applications. Chapman and Hall/CRC, Boca Raton (2013)
14. Jolliffe, I.: Principal Component Analysis. Springer, Berlin (1986). doi:10.1007/b98835
15. Kohonen, T.: Overture. In: Self-Organizing Neural Networks: Recent Advances and Applications, pp. 1–12. Springer, New York (2002)
16. Kranjc, J., Podpecan, V., Lavrac, N.: Clowdflows: A cloud based scientific workflow platform. In: Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 7524, pp. 816–819. Springer, Berlin/Heidelberg (2012). doi:10.1007/978-3-642-33486-3_54
17. Kranjc, J., Smailovič, J., Podpečan, V., Grčar, M., Žnidaršič, M., Lavrač, N.: Active learning for sentiment analysis on data streams: methodology and workflow implementation in the ClowdFlows platform. Inf. Process. Manag. **51**(2), 187–203 (2014). doi:10.1016/j.ipm.2014.04.001

18. Kurasova, O., Molytė, A.: Quality of quantization and visualization of vectors obtained by neural gas and self-organizing map. Informatica **22**(1), 115–134 (2011)
19. Mao, J., Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Netw. **6**(2), 296–317 (1995). doi:10.1109/72.363467
20. Massimo, B., Giuseppe, L., Castellani, M., Cavuoti, S., D'Abrusco, R., Laurino, O.: DAME: a distributed web based framework for knowledge discovery in databases. Memorie Soc. Astron. Ital. Suppl. **19**, 324–329 (2012)
21. Medvedev, V., Dzemyda, G., Kurasova, O., Marcinkevičius, V.: Efficient data projection for visual analysis of large data sets using neural networks. Informatica **22**(4), 507–520 (2011)
22. Podpečan, V., Zemenova, M., Lavrač, N.: Orange4WS environment for service-oriented data mining. Comput. J. **55**, 82–98 (2012). doi:10.1093/comjnl/bxr077
23. Ye, N.: The Handbook of Data Mining. LEA, New Jersey/London (2003)
24. Žilinskas, A., Žilinskas, J.: Two level minimization in multidimensional scaling. J. Glob. Optim. **38**(4), 581–596 (2007). doi:10.1007/s10898-006-9097-x
25. Žilinskas, A., Žilinskas, J.: A hybrid method for multidimensional scaling using city-block distances. Math. Meth. Oper. Res. **68**(3), 429–443 (2008). doi:10.1007/s00186-008-0238-5
26. Žilinskas, A., Žilinskas, J.: Branch and bound algorithm for multidimensional scaling with city-block metric. J. Glob. Optim. **43**(2-3), 357–372 (2009). doi:10.1007/s10898-008-9306-x

Comparative Study of Different Penalty Functions and Algorithms in Survey Calibration

Gareth Davies, Jonathan Gillard, and Anatoly Zhigljavsky

Abstract The technique of calibration in survey sampling is a widely used technique in the field of official statistics. The main element of the calibration process is an optimization procedure, for which a variety of penalty functions can be used. In this chapter, we consider three of the classical penalty functions that are implemented in many of the standard calibration software packages. We present two algorithms used by many of these software packages, and explore the properties of the calibrated weights and the corresponding estimates when using these two algorithms with the classical calibration penalty functions.

Keywords Survey calibration • Optimization • g-Weights

Introduction

Calibration of sample surveys is one of the key operations of official statistics. The problem of calibration can be defined informally as follows. Suppose some initial weights $d_1, \ldots, d_n$ are assigned to n objects of a survey. Suppose further that there are m auxiliary variables whose values on the sample are known, either exactly or approximately. The calibration problem seeks to improve the initial weights $d_1, \ldots, d_n$, by finding new weights $w_1, \ldots, w_n$ that incorporate this auxiliary information. The sample size n can be large; the number of auxiliary variables m can also be large although it is usually much smaller than n.

There are three main motivations for the use of calibration in the practice of official statistics (see, for example, [3, 27]). The first of these is to produce estimates consistent with other sources of data. Indeed, when a statistical office publishes the same statistics via two sources, the validity will be questioned if there are

G. Davies (✉) • J. Gillard
Cardiff School of Mathematics, Cardiff University, Cardiff, UK
e-mail: DaviesGP2@cardiff.ac.uk; GillardJW@cardiff.ac.uk

A. Zhigljavsky
Cardiff School of Mathematics, Cardiff University, Cardiff, UK
e-mail: ZhigljavskyAA@cardiff.ac.uk

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_6

contradictions between sources for the same statistics. We consider consistency as the primary motivation for the use of calibration throughout. The second reason for the use of calibration is to reduce the sampling variance of estimates. The inclusion of additional calibration information can lead to a reduction in the variance of estimates (see, for example, [22]). We shall see this in Example 2 from section "Example 2: Investigation of Calibrated Weights at Each Iteration of Algorithms 1 and 2". The third argument for the use of calibration is a reduction of the coverage and/or non-response bias of the survey estimates. See [19] for a more detailed discussion. This application of calibration will not be considered here.

In this chapter, we consider the use of Lagrange multipliers and Newton's method [38] for solving the calibration problem as motivated in [9]. Two of the most common algorithms will be presented. We show that the Jacobian matrix used in these algorithms has the form $A'H^{(s)}A$, with a specific choice of the matrix $H^{(s)}$. We explore various choices for the matrix $H^{(s)}$ and investigate convergence properties of the calibration algorithms in each case.

We consider solving the calibration problem for three of the classical functions, namely the quadratic, raking, and logit functions. The case of bounds on the g-weights will be of particular interest. For the logit function, these bounds are automatically satisfied by definition of the function. However, for the quadratic and raking functions, projections are required to satisfy these imposed bounds.

The structure of the chapter is as follows. Section "Calibration as an Optimization Problem" gives an introduction to the calibration problem and presents survey calibration as an optimization problem. We define the Lagrangian for this calibration problem and derive an iterative method for finding the Lagrange multipliers based on Newton's method. Section "Algorithms" presents two of the main algorithms described in the calibration literature and implemented in the statistical packages. We explore the convergence properties and the behavior of these algorithms for several examples in section "Examples". Section "Calibration Packages" gives a brief overview of the existing software packages that implement the calibration algorithms described in section "Algorithms". We conclude the chapter in section "Conclusion".

Calibration as an Optimization Problem

Mathematically, calibration is a large-scale convex optimization problem with linear constraints, amenable to modern methods of optimization [36, 40, 44, 45]. In this section, we formulate survey calibration as an optimization problem and consider various choices of the objective function that can be used. We present an approach for solving this calibration problem that uses Lagrange multipliers together with the Newton–Raphson method.

Formulation of the Problem

A vector of initial sample weights $D = (d_1, \ldots, d_n)'$ is given. The initial weights d_i are always assumed to be positive: $d_i > 0$ for all i. The aim of calibration is to adjust these initial weights in view of some additional information. The vector of calibrated weights will be denoted by $W = (w_1, \ldots, w_n)'$. For a more detailed discussion of calibration as an optimization problem, see [8]. The ratios of the weights w_i and d_i are usually considered rather than the weights w_i themselves. Hence, in this chapter, we deal with the so-called g-weights $g_i = w_i/d_i$. Denote the vector of g-weights by $G = (g_1, \ldots, g_n)'$.

Notation

We use the following key notation throughout the chapter:

$D = (d_1, \ldots, d_n)'$	Vector of initial weights,
$W = (w_1, \ldots, w_n)'$	Vector of calibrated weights,
$G = (g_1, \ldots, g_n)'$	Vector of the g-weights $g_i = w_i/d_i$ $(i = 1, \ldots, n)$,
$L = (l_1, \ldots, l_n)'$	Vector of lower bounds for the g-weights,
$U = (u_1, \ldots, u_n)'$	Vector of upper bounds for the g-weights,
$A = (a_{ij})_{i,j=1}^{n,m}$	Given $n \times m$ matrix,
$T = (t_1, \ldots, t_m)'$	Arbitrary $m \times 1$ vector,
$\mathbb{G}$	Feasible domain in the calibration problem,
$\mathbf{0} = (0, \ldots, 0)'$	n-vector of zeros,
$\mathbf{1} = (1, \ldots, 1)'$	n-vector of ones,
I_n	$n \times n$ identity matrix.

The Main Constraint

Let $X = (x_{ij})_{i,j=1}^{n,m}$ be a matrix of realizations of m auxiliary variables. The (i,j)-th entry x_{ij} of X denotes the value which the i-th member of the sample takes on the j-th auxiliary variable. Formally, X is an arbitrary $n \times m$ matrix.

Given the vector $T = (t_1, \ldots, t_m)'$, exact (hard) constraints can be written as $X'W = T$. These constraints are used for calibration of the weights. As $d_i > 0$ for all i, the hard constraints $X'W = T$ can be written in the form $A'G = T$, where the matrix $A = (a_{ij})_{i,j=1}^{n,m}$ has elements $a_{ij} = d_i x_{ij}$.

Sometimes, soft constraints $X'W \simeq T$ (or, equivalently, $A'G \simeq T$) are also used in the practice of calibration. We do not consider this case for two reasons: firstly, soft constraints are much less popular in the practice of official statistics and secondly, the optimization issues are very similar to the ones in the hard constraints case, see [8]. For an overview of calibration using soft constraints see [3].

Additional g-Weight Constraints

There are more constraints on G, in addition to $A'G = T$, that can be imposed. It is desirable for the calibrated weights to be non-negative; that is, $g_i \geq 0$ for all i (see, for example, [1, 2, 13]). Moreover, much of the calibration literature, including [4, 34], recommends imposing stricter constraints on the g-weights G of the form $L \leq G \leq U$, where $L = (l_1, \ldots, l_n)'$ and $U = (u_1, \ldots, u_n)'$ are some given $n \times 1$ vectors such that $0 \leq l_i < 1 < u_i \leq \infty$ for all i. That is, the g-weights should satisfy $l_i \leq g_i \leq u_i$ for some sets of lower and upper bounds l_i and u_i. If $l_i = 0$ and $u_i = \infty$ for all i, then the constraint $l_i \leq g_i \leq u_i$ coincides with the simple non-negativity constraint $g_i \geq 0$. In the majority of practical problems, $l_i = l$ and $u_i = u$ for all i with $0 \leq l < 1 < u \leq \infty$, where strict inequalities $l > 0$ and $u < \infty$ are very common.

The three possible choices (the most commonly used) of the vectors $L = (l_1, \ldots, l_n)'$ and $U = (u_1, \ldots, u_n)'$ are

(a) no constraints: $l_i = -\infty$ and $u_i = \infty$ for all i;
(b) non-negativity constraint: $l_i = 0$ and $u_i = \infty$ for all i; and
(c) general constraints: $0 \leq l_i < 1 < u_i \leq \infty$.

Statement of the Problem

The feasibility domain $\mathbb{G}$ for the vector of g-weights G is defined to be

$$\mathbb{G} = \{G = (g_1, \ldots, g_n)' : L \leq G \leq U \text{ and } A'G = T\}, \tag{1}$$

where $L, U \in \mathbb{R}^n$, $T \in \mathbb{R}^m$, and $A \in \mathbb{R}^{n \times m}$ are all given. Note that if the bounds $L \leq G \leq U$ for G are too narrow, then the feasible domain $\mathbb{G}$ may be empty.

In the process of calibration, the weights W have to stay as close as possible to the initial weights D. Equivalently, the g-weights G have to stay as close as possible to the n-vector of ones, denoted by $\mathbf{1} = (1, \ldots, 1)'$. To measure the closeness of G and $\mathbf{1}$, it is customary to use the function

$$\Phi(G) = \Phi(g_1, \ldots, g_n) = \sum_{i=1}^{n} q_i \phi(g_i), \tag{2}$$

where $\phi(\cdot)$ is a univariate, strictly convex function with $\phi(1) = 0$, and $q_1, \ldots, q_n$ are given non-negative numbers; typically, $q_i = d_i$ for all i.

The function (2) plays the role of the objective function in the following optimization problem:

$$\Phi(G) \to \min_{G \in \mathbb{G}}, \text{ where } \Phi(\cdot) \text{ and } \mathbb{G} \text{ are defined in (2) and (1), respectively.} \tag{3}$$

This optimization problem is exactly the problem of calibration we are interested in. This problem will be fully defined if we specify the function ϕ in (2). This is done in the next section.

Choice of the Function ϕ in (6.2)

There are two natural conditions on the function ϕ in (2):

(a) for a given l and u, $\phi(\cdot)$ is twice differentiable and strictly convex on its domain and
(b) $\phi(1) = 0$ and $\phi'(1) = 0$, where the derivatives are taken with respect to the argument of the function $\phi(g)$.

Effectively, these are the conditions also given in [10].

The Function ϕ

In the practice of calibration in official statistics, the following three functions ϕ are most commonly used:

(1) Quadratic:

$$\phi^{(Q)}(g) = \frac{1}{2}(g-1)^2;$$

(2) Raking:

$$\phi^{(R)}(g) = g\ln(g) - g + 1;$$

(3) Logit:

$$\phi^{(L)}(g;l,u) = \frac{1}{C}\left[(g-l)\ln\left(\frac{g-l}{1-l}\right) + (u-g)\ln\left(\frac{u-g}{u-1}\right)\right];$$

where $C = \frac{u-l}{(1-l)(u-1)}$.

In this chapter, the derivatives and inverse of the derivatives of the functions ϕ will also be important. Firstly, let us consider the derivatives of each of the functions listed above:

(1) Quadratic:

$$\phi'^{(Q)}(g) = g - 1;$$

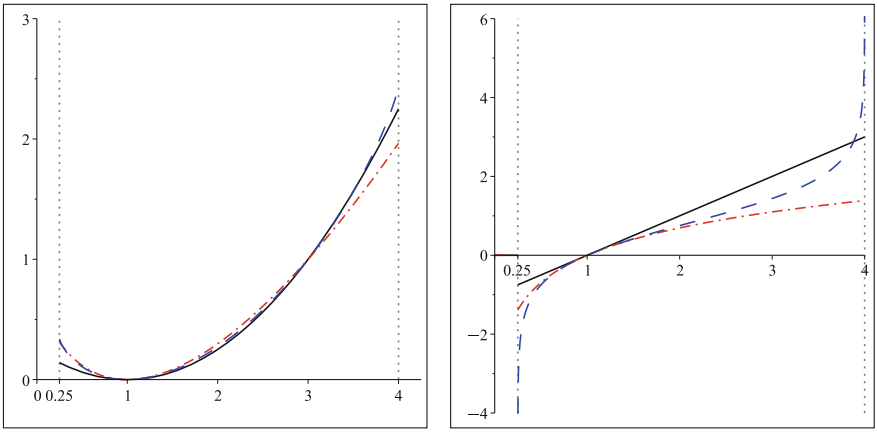

Fig. 1 Classical calibration penalty functions and their derivatives. (**a**) Functions $\phi^{(Q)}$ (*line*), $\phi^{(R)}$ (*dash*), and $\phi^{(L)}$ (*dot-dash*). (**b**) Derivatives of the functions $\phi^{(Q)}$ (*line*), $\phi^{(R)}$ (*dash*), and $\phi^{(L)}$ (*dot-dash*)

(2) Raking:

$$\phi'^{(R)}(g) = \ln(g);$$

(3) Logit:

$$\phi'^{(L)}(g;l,u) = \frac{1}{C}\left[\ln\left(\frac{g-l}{1-l}\right) - \ln\left(\frac{u-g}{u-1}\right)\right].$$

In Fig. 1, we plot each of the functions $\phi^{(Q)}$, $\phi^{(R)}$, and $\phi^{(L)}$ and their corresponding derivatives.

We have argued in [8] that there are other functions ϕ that satisfy the conditions (a) and (b) above, and have some additional attractive properties that can be more natural for use in (3) than the classical functions (1), (2), and (3). In particular, the function

$$\phi(g;l,u) = \frac{(g-1)^2}{(g-l)(u-g)}$$

has a very attractive property of equally penalizing g and $1/g$ in the case $l = 1/u$ (which is a common case in practice). Recall that $g_i = w_i/d_i$ is the ratio of the calibrated weight w_i to the initial weight d_i. Therefore, the multiplicative scale for measuring deviations of g_i from 1 is the most appropriate.

In this chapter, however, we do not pursue these arguments further. Whilst it could be argued that the quadratic function $\phi^{(Q)}(g) = \frac{1}{2}(g-1)^2$ is the most popular

and commonly used calibration function, there are recommendations against it. The function may lead to negative and/or extreme weight adjustments (see, for example, [6, 17]). On the other hand, while using the logit function, the g-weight constraints $L \leq G \leq U$ can be taken into account automatically.

The shape of the quadratic function $\phi^{(Q)}$ is arguably unnatural for the optimization problem (3). When using the quadratic function with general constraints $L \leq G \leq U$, the function does not take these constraints into account. The shape of the function results in a lack of penalty for extreme weights. This is due to the finite values of both the function $\phi^{(Q)}$ and its derivative at l_i and u_i. Therefore, optimization algorithms for solving the problem (3) have to be modified in this case to take these constraints into account. Projection algorithms are used to satisfy these constraints. The comments made here can also be applied to the function $\phi^{(R)}$ in the case of general constraints.

By contrast, the derivatives of the logit function $\phi^{(L)}$ tend to negative and positive infinity as one approaches l_i and u_i, respectively. Despite having a finite function value at l_i and u_i, the infinite derivatives do not allow optimization algorithms to give g-weights outside of the range $L \leq G \leq U$. However, it could be argued that a more natural optimization function would be such that both its value and derivative tend to infinity at l_i and u_i, respectively. See, for example, functions $\phi^{(7)}$ and $\phi^{(8)}$ in [8].

The Function h

Let $h(x) = (\phi')^{-1}(x)$, i.e. h is the inverse of ϕ'. From the strong convexity of ϕ, we have that $\phi'(g)$ is strictly increasing and so the inverse function h is uniquely defined. It can be seen from Fig. 1b, that for each of the functions ϕ considered in section "The Function ϕ", their derivatives are strictly increasing.

For the functions $\phi^{(Q)}$, $\phi^{(R)}$, and $\phi^{(L)}$ introduced in section "The Function ϕ", the corresponding h-functions are

(1) Quadratic:

$$h^{(Q)}(x) = 1 + x;$$

(2) Raking:

$$h^{(R)}(x) = \exp(x);$$

(3) Logit:

$$h^{(L)}(x; l, u) = \frac{l(u-1) + u(1-l)\exp(Cx)}{(u-1) + (1-l)\exp(Cx)};$$

where C is defined as in section "The Function ϕ".

Lagrangian for the Calibration Problem (6.3)

Returning to the calibration problem (3), let $\Lambda = (\lambda_1 \ldots, \lambda_m)'$ be the m-vector of Lagrange multipliers. We can write the Lagrangian for the problem (3) with function (2) and constraints (1) as

$$\mathscr{L}(G,\Lambda) = \Phi(G) - \Lambda'(A'G - T) = \sum_{i=1}^{n} \phi(g_i) - \sum_{j=1}^{m} \lambda_j \left(\sum_{i=1}^{n} a_{ij} g_i - t_j \right).$$

Set $\phi'(g_i) = \frac{\partial \phi(g)}{\partial g}\Big|_{g=g_i}$. Differentiating $\mathscr{L}(G,\Lambda)$ with respect to g_i $(i = 1,\ldots,n)$ we have

$$\frac{\partial \mathscr{L}(G,\Lambda)}{\partial g_i} = \phi'(g_i) - \sum_{j=1}^{m} \lambda_j a_{ij}.$$

Let $\mathbf{a}_i$ denote the i-th row of A, $i = 1,\ldots,n$ so that $A = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{pmatrix}$ and $\mathbf{a}_i = (a_{i1},\ldots,a_{im})$. Setting $\frac{\partial \mathscr{L}(G,\Lambda)}{\partial g_i} = 0$ gives

$$\phi'(g_i) = \sum_{j=1}^{m} \lambda_j a_{ij} = \mathbf{a}_i \Lambda. \tag{4}$$

Recall from section "The Function $\boldsymbol{h}$" that $h(x) = (\phi')^{-1}(x)$. Then (4) implies that the g-weights g_i $(i = 1,\ldots,n)$ corresponding to the vector Λ of Lagrange multipliers are given by

$$g_i = h(\mathbf{a}_i \Lambda) \quad \text{for } i = 1,\ldots,n. \tag{5}$$

Let us consider the calibration constraint $A'G = T$. Using (5) we can write

$$A'G = T \iff \sum_{i=1}^{n} \mathbf{a}_i' g_i = T \iff \sum_{i=1}^{n} \mathbf{a}_i' h(\mathbf{a}_i \Lambda) = T. \tag{6}$$

We wish to solve the equation

$$\sum_{i=1}^{n} \mathbf{a}_i' h(\mathbf{a}_i \Lambda) = T, \tag{7}$$

with respect to Λ. Upon solving for Λ, we can derive the calibrated weights $G = (h(\mathbf{a}_1\Lambda), \ldots, h(\mathbf{a}_n\Lambda))'$ as given by (5).

We consider algorithms for solving (7) with respect to Λ that use a Newton–Raphson method [38] as described in [9]. The algorithms we shall consider have the form

$$\Lambda^{(s+1)} = \Lambda^{(s)} + (A'H^{(s)}A)^{-1}(T - A'G^{(s)}), \quad \text{for } s = 0, 1, 2, \ldots, \tag{8}$$

where $\Lambda^{(0)} = \mathbf{0}$, $G^{(s)}$ denotes the updated vector of g-weights at iteration s, and $H^{(s)}$ is an $n \times n$ diagonal matrix to be specified (see section "The Matrix $H^{(s)}$").

First Iteration

Setting $s = 0$, the first iteration in (8) can be written as

$$\Lambda^{(1)} = \Lambda^{(0)} + (A'H^{(0)}A)^{-1}(T - A'G^{(0)}).$$

For the algorithms we shall consider in this chapter, $\Lambda^{(0)} = \mathbf{0}$, $H^{(0)} = I_n$, and $G^{(0)} = \mathbf{1}$. Thus we can write the first iteration as

$$\Lambda^{(1)} = (A'A)^{-1}(T - A'\mathbf{1}). \tag{9}$$

We remark that $\Lambda^{(1)}$ is independent of the choice of the function ϕ. It is easy to see that (9) coincides with the vector Λ that solves (7) in the case of function $h^{(Q)}$ with no constraints (that is, $l_i = -\infty$ and $u_i = \infty$ for all i). Also, it is well documented that the calibrated weights $W = (d_1 h^{(Q)}(\mathbf{a}_1\Lambda^{(1)}), \ldots, d_n h^{(Q)}(\mathbf{a}_n\Lambda^{(1)}))'$ are equivalent to the weights obtained using the generalized regression estimator (GREG), see [28].

Jacobian

Let $J(\Lambda)$ denote the Jacobian of $\sum_{i=1}^{n} \mathbf{a}_i' h(\mathbf{a}_i\Lambda)$ considered as a function of Λ. It is easy to see that $J(\Lambda^{(s)}) = A'H^{(s)}A$, where $H^{(s)} = \mathrm{diag}(h'(\mathbf{a}_1\Lambda^{(s)}), \ldots, h'(\mathbf{a}_n\Lambda^{(s)}))$ and $h'(\mathbf{a}_i\Lambda) = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\mathbf{a}_i\Lambda}$. Applying the Newton–Raphson method [38] to solve (7) for Λ gives the iterative procedure:

$$\Lambda^{(s+1)} = \Lambda^{(s)} + (J(\Lambda^{(s)}))^{-1}(T - A'G^{(s)}). \tag{10}$$

This is a particular case of (8) with $H^{(s)}$ defined as above. This Newton–Raphson approach is proposed in [9]. In the next section, we consider various choices of the matrix $H^{(s)}$.

The Matrix $H^{(s)}$

The following three forms of $H^{(s)}$ appear in the literature and existing software.

Choices of the Matrix $H^{(s)}$

Newton–Raphson $H^{(s)} = \text{diag}(h'(\mathbf{a}_1\Lambda^{(s)}),\ldots,h'(\mathbf{a}_n\Lambda^{(s)}))$ as described in section "Jacobian". The method (10) requires computation of $J(\Lambda^{(s)}) = A'H^{(s)}A$ at each iteration. The method converges quickly in an ideal situation when computation is exact, but in practice this choice of $H^{(s)}$ can lead to an unstable algorithm.

Identity Matrix $H^{(s)} = I_n$ for all s, which means that, at each iteration, the initial Jacobian $J(\Lambda^{(0)}) = A'A$ is used.

Matrix with g-Weights on the Diagonal $H^{(s)} = \text{diag}(h(\mathbf{a}_1\Lambda^{(s)}),\ldots,h(\mathbf{a}_n\Lambda^{(s)}))$; the diagonal entries of $H^{(s)}$ are simply $G^{(s)}$, the values of the calibrated weights G at the s-th iteration. Since these weights are computed as part of the algorithm at each iteration, the matrix $H^{(s)}$ does not require additional computations, unlike the first method.

In section"Example 3: Convergence Properties of Algorithms 1 and 2 for Various Choices of $H^{(s)}$", we consider the convergence properties of the iterative procedure (8) for these three forms of the matrix $H^{(s)}$.

$H^{(s)}$ for Various Calibration Functions

We now consider the forms of the matrix $H^{(s)}$ for the three calibration functions outlined in section "The Function ϕ". We consider the quadratic function $\phi^{(Q)}$ in the case of no constraints and general constraints, the raking function $\phi^{(R)}$ in the case of non-negativity constraints and general constraints, and the logit function $\phi^{(L)}$ in the case of general constraints.

Function $\phi^{(Q)}$ with No Constraints In this case, the iterative method (8) converges in one iteration. The g-weights derived from this method correspond to the weights using the generalized regression estimator (see [28]). Note that, in this case, the Newton–Raphson and identity matrix are equivalent, since $h'^{(Q)}(x) = 1$ for $x \in \mathbb{R}$. Since $h^{(Q)}(0) = 1$, the matrix with g-weights on the diagonal is equivalent to the identity matrix for the first iteration. Since we only perform one iteration for $\phi^{(Q)}$ with no constraints, this means that these two cases are also equivalent. Hence, for $\phi^{(Q)}$ with no constraints, all three cases of $H^{(s)}$ outlined in section "Choices of the Matrix $H^{(s)}$" are equivalent.

Function $\phi^{(Q)}$ with General Constraints In this case, the iterative method (8) may not converge in one iteration. A projection algorithm is used to ensure any g-weights g_i $(i = 1,\ldots,n)$ that fall outside of the interval $[l_i, u_i]$ are projected back to this interval. Informally, if $g_i < l_i$, we project it such that $g_i = l_i$; similarly, if $g_i > u_i$,

we project it such that $g_i = u_i$. The algorithm is then updated to account for this projection and the calibration is continued. See section "Algorithm 2: Projections to the Additional Constraints Required" for a full description of the algorithm. Note that in this case, the Newton–Raphson and identity matrix are equivalent, since $h'^{(Q)}(x) = 1$ for $x \in \mathbb{R}$. However, as we now perform more than one iteration, the matrix with g-weights on the diagonal will be different to the other matrices from the second iteration onwards.

Function $\phi^{(R)}$ with Non-Negativity and General Constraints The comments made in this section regarding $H^{(s)}$ for the function $\phi^{(R)}$ apply in both the cases of non-negativity constraints and general constraints. Observe that for $x \in [0, \infty)$ we have $h'^{(R)}(x) = h^{(R)}(x) = \exp(x)$, hence the Newton–Raphson matrix and matrix with g-weights on the diagonal are equivalent. However, since $\exp(x) \neq 1$ for all $x \in [0, \infty)$, the identity matrix form of $H^{(s)}$ will be different to the other matrices from the second iteration onwards (note that, since $\exp(0) = 1$, all the matrices will be equal to the identity matrix at the first iteration). Due to the function definition, the non-negativity constraints are automatically taken into account with function $\phi^{(R)}$. However, for general constraints, the projection algorithm outlined in section "Algorithm 2: Projections to the Additional Constraints Required" is required.

Function $\phi^{(L)}$ with General Constraints We remark that, for the function $\phi^{(L)}$, the corresponding function $h^{(L)}(x)$ is such that $h'^{(L)}(x) \neq h^{(L)}(x)$ for all $x \in [l_i, u_i]$, hence the Newton–Raphson matrix and matrix with g-weights on the diagonal are not equivalent. Note also that $h'^{(L)}(x)$ and $h^{(L)}(x)$ are not equal to 1 for all $x \in [l_i, u_i]$, hence the identity matrix form of $H^{(s)}$ will be different to the other two forms from the second iteration onwards. Note that $h'^{(L)}(0) = h^{(L)}(0) = 1$, hence all three matrices will be equivalent for the first iteration.

Table 1 summarizes this section. We give the functions, the constraint cases considered, and the relationships between the various forms of the matrix $H^{(s)}$.

We shall consider convergence properties of (8) for all forms of the matrix $H^{(s)}$ in section "Example 3: Convergence Properties of Algorithms 1 and 2 for Various Choices of $H^{(s)}$". However, before considering several examples, we fully describe the algorithms for solving (7) using the iterative procedure (8).

Table 1 Comparisons of the three forms of the matrix $H^{(s)}$ for the classical calibration functions in various constraint cases

Function	Constraints	Newton–Raphson (I), Identity (II), and matrix with g-weights (III)
$\phi^{(Q)}$	No	$(I) = (II) = (III)$
$\phi^{(Q)}$	General	$(I) = (II) \neq (III)$
$\phi^{(R)}$	Non-negativity	$(I) = (III) \neq (II)$
$\phi^{(R)}$	General	$(I) = (III) \neq (II)$
$\phi^{(L)}$	General	$(I) \neq (II) \neq (III)$

Algorithms

In this section, we consider specific algorithms for solving the Eq. (7) using the iterative procedure (8).

The Main Algorithms

In this chapter, we consider two main algorithms for the calibration problem (3). Some alternative algorithms that we do not present in this chapter are briefly described in section "Other Algorithms".

Any specific algorithm of the form (8) will be characterized by the following:

- Choice of the function ϕ;
- Choice of the matrix $H^{(s)}$;
- Choice of the constraints on G additional to the main constraint $A'G = T$ (that is, choice of L and U); and
- Choice of an appropriate stopping rule.

There are three main choices for the function ϕ; see section "The Function ϕ". There are also three choices for the matrix $H^{(s)}$; see section "The Matrix $H^{(s)}$". The three choices of the vectors $L = (l_1, \ldots, l_n)'$ and $U = (u_1, \ldots, u_n)'$ were described in section "Additional $\boldsymbol{g}$-Weight Constraints". The stopping rule used in all the algorithms described in this section is of the following type: STOP if either $A'G^{(s)} = T$ to within a pre-specified accuracy or the maximum number of iterations is met.

We consider two algorithms used in many of the standard calibration packages (see section "Calibration Packages" for a further discussion of these packages). Algorithm 1 is applicable to the function $\phi^{(L)}$ in the case of general constraints, and to function $\phi^{(R)}$ in the case of non-negativity constraints. Algorithm 2 is applicable to the functions $\phi^{(Q)}$ and $\phi^{(R)}$ in the case of general constraints.

Algorithm 1: No Projections Needed

Input Matrix A, vector T, vectors $L = (l_1, \ldots, l_n)'$ and $U = (u_1, \ldots, u_n)'$, the form of the matrix $H^{(s)}$, and the function ϕ (either $\phi^{(L)}$ or $\phi^{(R)}$). For the function $\phi^{(R)}$, we can only use $l_i = 0$ and $u_i = \infty$ for all $i = 1, \ldots n$. For the function $\phi^{(L)}$, $l_i \geq 0$ and $u_i < \infty$ for all $i = 1, \ldots n$.

Output The vector $G^{(s+1)}$ computed at the final iteration; this vector is the (approximate) solution of the calibration problem (3) for the chosen function ϕ.

Algorithm 1

1. Set $s = 0, \Lambda^{(0)} = \mathbf{0}, G^{(0)} = \mathbf{1}$, and $H^{(0)} = I_n$.
2. Compute $\Lambda^{(s+1)} = (\lambda_1^{(s+1)}, \ldots, \lambda_m^{(s+1)})'$ using (8).
3. Compute $G^{(s+1)} = (g_1^{(s+1)}, \ldots, g_n^{(s+1)})'$ by $g_i^{(s+1)} = h(\mathbf{a}_i \Lambda^{(s+1)})$.
4. Compute $H^{(s+1)}$ as outlined in section "Jacobian" using one of the three forms of $H^{(s)}$.
5. STOP if the stopping criterion is satisfied. Otherwise, set $s \to s+1$ and return to Step 2.

Algorithm 1 is characterized by the following: an input matrix A, an input vector T, the form of the matrix $H^{(s)}$, the function $\phi^{(L)}$ or $\phi^{(R)}$, the vectors L and U in the case of function $\phi^{(L)}$, and the constraints for g_i used. This algorithm cannot be used for $\phi^{(Q)}$ (unless we do not impose any constraints on g_i and in this case the solution is obtained in the first iteration, see section "Algorithm 1a: Quadratic Function $\phi^{(Q)}$ with No Constraints"). For the function $\phi^{(R)}$, Algorithm 1 can only be used if the required constraint is the non-negativity constraint.

Remark. Algorithm 1 cannot be used for $\phi^{(Q)}$ unless we do not impose any constraints on g_i, see case (a) in section "Additional $\boldsymbol{g}$-Weight Constraints"; in this case, the solution is obtained in the first iteration, see Algorithm 1a below.

Algorithm 1a: Quadratic Function $\phi^{(Q)}$ with No Constraints

This method is non-iterative and contains the following two steps only:

1. Compute $\Lambda^{(1)}$ from (8) using $G^{(0)}$ and $H^{(0)}$.
2. Compute $G^{(1)} = \mathbf{1} + A'\Lambda^{(1)}$.

For the functions $\phi^{(Q)}$ and $\phi^{(R)}$ with general constraints, Algorithm 2 described below should be used.

Algorithm 2: Projections to the Additional Constraints Required

Input Matrix A, vector T, vectors $L = (l_1, \ldots, l_n)'$ and $U = (u_1, \ldots, u_n)'$ such that $0 \le l_i < 1 < u_i \le \infty$ for all $i = 1, \ldots n$, form of the matrix $H^{(s)}$, and function ϕ (either $\phi^{(Q)}$ or $\phi^{(R)}$).

Output The vector $G^{(s+1)}$ computed at the final iteration; this vector is the (approximate) solution of the calibration problem (3) for the chosen function ϕ.

Algorithm 2

1. Set $s = 0, n_0 = n, \Lambda^{(0)} = \mathbf{0}, G^{(0)} = \mathbf{1}, H^{(0)} = I_n, A^{(0)} = A, T^{(0)} = T, L^{(0)} = L$, and $U^{(0)} = U$.
2. Compute $\Lambda^{(s+1)} = (\lambda_1^{(s+1)}, \ldots, \lambda_m^{(s+1)})'$ by

$$\Lambda^{(s+1)} = \Lambda^{(s)} + ((A^{(s)})' H^{(s)} A^{(s)})^{-1} (T^{(s)} - (A^{(s)})' G^{(s)}), \tag{11}$$

which is the formula (8) with $A = A^{(s)}$ and $T = T^{(s)}$.
3. Use Algorithm 2' below to compute n_{s+1}, matrix $A^{(s+1)}$ of size $n_{s+1} \times m$, vector $T^{(s+1)}$ of size m, and vectors $L^{(s+1)}$, $U^{(s+1)}$, and $G^{(s+1)}$ of size n_{s+1}.
4. Compute the matrix $H^{(s+1)}$ of size $n_{s+1} \times n_{s+1}$ as outlined in section "Jacobian" with $A = A^{(s+1)}$ and $n = n_{s+1}$. For $s > 0$, the matrix $H^{(s)}$ is computed as outlined in section "The Matrix $H^{(s)}$" using one of the three forms of $H^{(s)}$. Here $\mathbf{a}_i^{(s)}$ denote i-th rows of $A^{(s)}$ so that $\mathbf{a}_i^{(s)} = (a_{i1}, \ldots, a_{im})$, $i = 1, \ldots, n_s$.
5. STOP if the stopping criterion is satisfied. Otherwise, set $s \to s+1$ and return to Step 2.

Algorithm 2': Performing Step 3 in Algorithm 2

Input Matrix $A^{(s)} = (a_{ik}^{(s)})_{i,k}$ of size $n_s \times m$, vector $T^{(s)}$ of size m, the vectors $L^{(s)} = (l_1^{(s)}, \ldots, l_{n_s}^{(s)})'$ and $U^{(s)} = (u_1^{(s)}, \ldots, u_{n_s}^{(s)})'$ of size n_s.

Output Integer $n_{s+1} \le n_s$, matrix $A^{(s+1)}$ of size $n_{s+1} \times m$, vector $T^{(s+1)}$ of size m, vectors $G^{(s+1)}$, $L^{(s+1)}$, and $U^{(s+1)}$ of size n_{s+1}.

Algorithm 2'

1. Compute vector $\widetilde{G}^{(s+1)} = (\tilde{g}_1^{(s+1)}, \ldots, \tilde{g}_{n_s}^{(s+1)})'$ of size n_s by $\tilde{g}_i^{(s+1)} = h(\mathbf{a}_i^{(s)} \Lambda^{(s+1)})$, $i = 1, \ldots, n_s$.
2. If $l_i^{(s)} \le \tilde{g}_i^{(s+1)} \le u_i^{(s)}$ for all $i = 1, \ldots, n_s$, then set $n_{s+1} = n_s$, $A^{(s+1)} = A^{(s)}$, $G^{(s+1)} = \widetilde{G}^{(s+1)}$, $T^{(s+1)} = T^{(s)}$, $L^{(s+1)} = L^{(s)}$, and $U^{(s+1)} = U^{(s)}$. Otherwise go to the next step.
3. Define

$$\gamma_i^{(s+1)} = \begin{cases} \tilde{g}_i^{(s+1)} & \text{if } l_i^{(s)} \le \tilde{g}_i^{(s+1)} \le u_i^{(s)}; \\ l_i^{(s)} & \text{if } \tilde{g}_i^{(s+1)} < l_i^{(s)}; \\ u_i^{(s)} & \text{if } \tilde{g}_i^{(s+1)} > u_i^{(s)}. \end{cases} \tag{12}$$

The map (12) produces a split of the set of indices $\Omega^{(s)} = \{1, 2, \ldots, n_s\}$ into three subsets: $\Omega_l^{(s)}$, $\Omega_u^{(s)}$, and $\Omega_m^{(s)}$.

Define n_{s+1} to be the number of times the equality $\gamma_i^{(s+1)} = \tilde{g}_i^{(s+1)}$ holds. Let $\Omega_m^{(s)} = \{i_1, i_2, \ldots, i_{n_{s+1}}\}$ be the ordered set of indices such that the equality $\gamma_i^{(s+1)} = \tilde{g}_i^{(s+1)}$ holds for $i = i_j, j = 1, \ldots, n_{s+1}$. The indices in $\Omega_m^{(s)}$ are ordered so that $i_j < i_{j+1}$ for all j.

Similarly, $\Omega_l^{(s)}$ and $\Omega_u^{(s)}$ are defined as the ordered sets of indices such that the inequalities $\tilde{g}_i^{(s+1)} < l_i^{(s)}$ or $\tilde{g}_i^{(s+1)} > u_i^{(s)}$ hold, respectively, for the indices in $\Omega_l^{(s)}$ and $\Omega_u^{(s)}$.

4. Define the matrix $A^{(s+1)} = (a_{jk}^{(s+1)})_{j,k}$ of size $n_{s+1} \times m$ by $a_{jk}^{(s+1)} = a_{i_j k}^{(s)}$ for $j = 1, \ldots, n_{s+1}$ such that $i_j \in \Omega_m^{(s)}$. Similarly, we compute vectors $G^{(s+1)} = (g_j^{(s+1)})_j$, $L^{(s+1)} = (l_j^{(s+1)})_j$, and $U^{(s+1)} = (u_j^{(s+1)})_j$ of size n_{s+1} as follows: $g_j^{(s+1)} = \gamma_{i_j}^{(s+1)}$, $l_j^{(s+1)} = l_{i_j}^{(s)}$, and $u_j^{(s+1)} = u_{i_j}^{(s)}$ for $j = 1, \ldots, n_{s+1}$ such that $i_j \in \Omega_m^{(s)}$.
5. Let $\tilde{n}_s = n_s - n_{s+1}$. Form $\Omega_l^{(s)} \cup \Omega_u^{(s)} = \{i_1, i_2, \ldots, i_{\tilde{n}_s}\}$, the set of indices such that either of the inequalities $\tilde{g}_i^{(s+1)} < l_i^{(s)}$ or $\tilde{g}_i^{(s+1)} > u_i^{(s)}$ hold for $i = i_l$, $l = 1, \ldots, \tilde{n}_s$.
6. Define matrix $\tilde{A}^{(s+1)} = (a_{lk}^{(s+1)})_{l,k}$ of size $\tilde{n}_s \times m$ by $a_{lk}^{(s+1)} = a_{i_l k}^{(s)}$ for $l = 1, \ldots, \tilde{n}_s$ such that $i_l \in \Omega_l^{(s)} \cup \Omega_u^{(s)}$. Similarly, we compute vector $\tilde{G}^{(s+1)} = (g_l^{(s+1)})_l$ of size $\tilde{n}_s$ as $g_l^{(s+1)} = \gamma_{i_l}^{(s+1)}$ for $l = 1, \ldots, \tilde{n}_s$ such that $i_l \in \Omega_l^{(s)} \cup \Omega_u^{(s)}$.
7. Compute $T^{(s+1)} = T^{(s)} - (\tilde{A}^{(s+1)})' \tilde{G}^{(s+1)}$.

Other Algorithms

Whilst we have considered the algorithms that are arguably most commonly used in the calibration software, there are several algorithms that we have not considered in this chapter. The first of these is the scale modified quadratic algorithm. This algorithm uses projections to satisfy the calibration constraints $A'G = T$ at each iteration, continuing until the range restrictions $L \leq G \leq U$ are met to within a pre-specified accuracy. The algorithm is applied only in the case of function $\phi^{(Q)}$ with general constraints. This algorithm is attributed to [16]. The method is outlined in [32] (see Method 3) with further information in Sect. 2.2 of [26]. This algorithm is used in the calibration software BASCULA (see section "Details of the Calibration Packages" for further information).

The shrinkage minimization method uses a similar algorithm to the scaled modified quadratic algorithm, but we do not consider it here as it is not used by any of the packages considered in this chapter. See Method 4 of [32] and Sect. 2.4 of [26] for more information on the shrinkage minimization method.

Finally, the so-called projection method algorithm, attributed to [14] with details outlined in [11], is another algorithm that is only used for the case of the quadratic function $\phi^{(Q)}$ with general constraints. This algorithm is used in Statistics Canada's GES software (see section "Details of the Calibration Packages" for further details).

In the next section, we present several examples that use Algorithms 1 and 2 described in section "The Main Algorithms".

Examples

We consider various examples to explore the convergence properties of the algorithms described in section "The Main Algorithms". We begin by considering

Algorithm 1 for function $\phi^{(L)}$ with general constraints. We shall show that, for this case, there are three types of convergence. In the second example, we explore properties of the calibrated weights at subsequent iterations of Algorithms 1 and 2. We conclude by exploring convergence properties of both Algorithms 1 and 2 for all possible combination of functions with various choices of the matrix $H^{(s)}$.

Example 1: Convergence Properties of Algorithm 1 for $\phi^{(L)}$ with General Constraints

In this section, we explore the behavior of Algorithm 1 for the function $\phi^{(L)}$. We shall illustrate that there are three interesting cases:

1. If the constraint $A'G = T$ cannot be satisfied, then the algorithm does not converge and gives bad or no results.
2. If the constraint $A'G = T$ can be satisfied but some of the weights g_i tend to l_i or u_i, then the algorithm gives slow or no convergence.
3. If $A'G = T$ can be satisfied and all the weights g_i remain well within the bounds imposed by l_i and u_i, then the algorithm converges quickly.

We illustrate each of these cases using a small example. We fix $T = (6.4, 12.2)'$ and $l_1 = l_2 = l = 0.5$. We use Algorithm 1 with the function $\phi^{(L)}$ and general constraints (type (c) in section "Additional $\boldsymbol{g}$-Weight Constraints"). We shall only consider the Newton–Raphson form of the matrix $H^{(s)}$ in this example. We take the matrix $A = (\mathbf{a}_1, \mathbf{a}_2)'$ where $\mathbf{a}_1 = (2, 6)$ and $\mathbf{a}_2 = (3, 5)$. We shall vary the value of $u_1 = u_2 = u$ to illustrate each of the three cases described above.

For comparison purposes, we shall also consider the convergence properties of Algorithm 2 for the function $\phi^{(Q)}$ in the case of general constraints. However, as we shall see, for Algorithm 2 using $\phi^{(Q)}$ there are only two cases—convergence or no convergence.

Case 1: Fast Convergence We begin by taking $u = 2$. After five iterations of Algorithm 1, the vector $A'G - T$ has both elements of order 10^{-8}, thus after five iterations we have essentially satisfied the constraints. The algorithm continues to improve in accuracy, and after 11 iterations of Algorithm 1, $A'G - T$ has infinitesimally small elements. The determinant of the final Jacobian matrix is approximately 8.55. Since the g-weights do not approach l or u, there are no issues with infinite values appearing in the Jacobian matrix (recall that the derivative of $\phi^{(L)}$ is infinite at l and u). For this case, the weights converge to $G = (0.575, 1.75)'$ very quickly. We plot the weights for each iteration of Algorithm 1 in Fig. 2a (circle, line, black). Contrast this to the weights in Fig. 2b, where we see that Algorithm 2 for $\phi^{(Q)}$ has converged after one iteration.

Case 2: Slow Convergence We now change the value of u to 1.75, the value to which the larger weight converged in Case 1. Re-running Algorithm 1 in this case gives much slower convergence. After five iterations of Algorithm 1, the vector

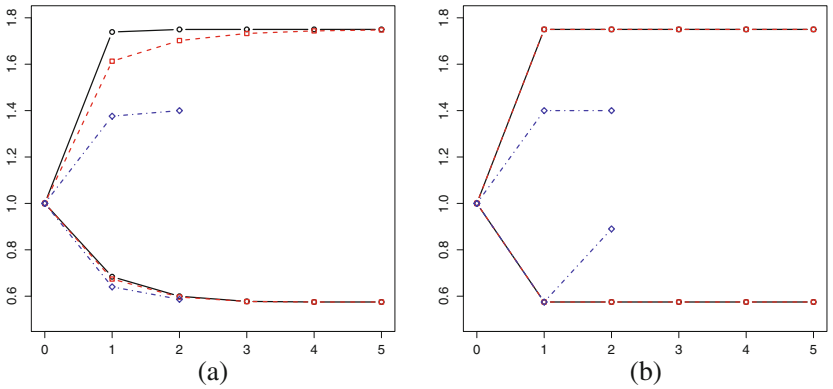

Fig. 2 Plot of g-weights for the first five iterations of Algorithm 1 for $\phi^{(L)}$ and Algorithm 2 for $\phi^{(Q)}$ in different cases of convergence; convergence when $u = 2$ (*black, circle, line*), convergence when $u = 1.75$ (*red, square, dash*) and convergence when $u = 1.4$ (*blue, diamond, dot-dash*). (**a**) g-weights using Algorithm 1 for $\phi^{(L)}$. (**b**) g-weights using Algorithm 2 for $\phi^{(Q)}$

$A'G - T$ has elements -0.007 and -0.012. The determinant of the Jacobian matrix is 0.319, which is much smaller than the corresponding value in Case 1. After 12 iterations of Algorithm 1, $G = (0.575, 1.749)'$. The elements of $A'G - T$ are of order 10^{-5}, larger than the values of $A'G - T$ after five iterations in Case 1. The determinant of the Jacobian matrix after 12 iterations is 2.91×10^{-4}. This continues to decrease with subsequent iterations, reaching an infinitesimally small value after 50 iterations. The algorithm continues to move slowly towards the solution $G = (0.575, 1.75)'$; however, the algorithm fails to reach the upper bound of 1.75, since reaching this upper bound would lead to an infinite value in the Jacobian matrix (recall that the derivative of $\phi^{(L)}$ is ∞ at u). We plot the weights for the first five iterations of Algorithm 1 in Fig. 2a (square, dash, red). Contrast this with the weights for Algorithm 2 with function $\phi^{(Q)}$, plotted in Fig. 2b, where once again the algorithm converges in one iteration.

Case 3: No Convergence We now change the value of u to 1.4. In this case, the algorithm runs for three iterations. At the third iteration, the value of the weights are $G = (0.576, 1.400)'$; however, the entries of $A'G - T$ are -1.049 and -1.746. Therefore, the calibration constraints are not satisfied. The determinant of the Jacobian matrix after three iterations is 2.75×10^{-7}. This small value of the determinant, together with the upper weight virtually reaching the upper bound of 1.4, causes the algorithm to fail when trying to invert the Jacobian matrix. The matrix will have an infinitely large entry as well as an infinitesimally small determinant, both of which cause the algorithm to fail. We plot the trajectory of the weights in this case in Fig. 2a (diamond, dot-dash, blue). Figure 2b shows the trajectory of the weights for Algorithm 2 with function $\phi^{(Q)}$, which also runs for three iterations but fails to converge. We observe that the weights at the third iteration for Algorithm 2 with function $\phi^{(Q)}$ are different to those for the third iteration of Algorithm 1 with function $\phi^{(L)}$. The algorithms are behaving differently

Table 2 Convergence properties of Algorithm 1 for $\phi^{(L)}$ in Example 1 for various values of u with $l = 0.575$

Case	u	No. of iter	$A'G - T$
Fast convergence	2	5	10^{-8}
Slow convergence	1.75	11	10^{-5}
No convergence	1.4	3 (then stop)	1

in an attempt to find a solution to the calibration problem which, in this case, does not have a solution.

Table 2 summarizes the key points from this example. We conclude this section with some brief remarks regarding Algorithms 1 and 2.

Multi-Start We started Algorithms 1 and 2 for various $\Lambda^{(0)}$ in both Cases 1 and 2 above. For suitable choices of $\Lambda^{(0)}$ (in particular, such that $h(\mathbf{a}_i\Lambda^{(0)})$ gave weights g_i in the range $l_i \leq g_i \leq u_i$, $i = 1, \ldots, n$), the algorithms converged to the same solution. The final g-weights g_i ($i = 1, \ldots, n$) and the Lagrange multipliers Λ were the same in all cases (to within computer accuracy).

Derivatives The identities $\phi'(g_i) = \mathbf{a}_i\Lambda$ and $g_i = h(\mathbf{a}_i\Lambda)$ were confirmed for the final value of the Lagrange multipliers Λ and the calibrated weights g_i given by Algorithms 1 and 2. This allows us to conclude that the algorithms have converged to a (local) minimum.

Example 2: Investigation of Calibrated Weights at Each Iteration of Algorithms 1 and 2

In this example, we further explore the convergence properties of Algorithms 1 and 2 by considering the calibrated weights at each iteration of the algorithms.

Data

In this example, we consider the Belgian municipalities dataset included in the "sampling" package in R (see [35] for more details). The dataset provides information about the Belgian population on July 1st 2004 compared with July 1st 2003, and includes financial information about the municipality incomes at the end of 2001. Data is available for the 589 municipalities in Belgium. There are 17 variables in the dataset, including the municipality name and province number. However, the 8 variables of interest in this example are the number of men on July 1st 2003, the number of women on July 1st 2003, the difference in the number of men on July 1st 2003 and July 1st 2004, the difference in the number of women on July 1st 2003 and July 1st 2004, total taxable income in Euros in 2001, total taxation in Euros

in 2001, average of the income-tax return in Euros in 2001, and the median of the income-tax return in Euros in 2001.

We take a simple random sample of size 200 and assign initial weights $d_i = N/n$ where N is the size of the population and n is the sample size (in this example $N = 589$ and $n = 200$). These would be the weights used in the Horvitz–Thompson estimator [15]. The values of the 8 variables of interest for each of the 200 sample members are used to form the 200×8 matrix X. The matrix A is formed using the relationship $a_{ij} = d_i x_{ij}$ as introduced in section "The Main Constraint". Using Algorithm 1, we wish to calibrate this sample to the known totals for each of the 8 variables. These known totals are used to form the 8×1 vector T. We take $l_i = l = 0.73$ and $u_i = u = 1.3$ for $i = 1, \ldots, 200$.

Calibrated Weights

Figure 3 shows histograms and the corresponding density plots of the weights at iteration 1 (see Fig. 3a), iteration 5 (see Fig. 3b), and the final, eighth, iteration (see Fig. 3c) of Algorithm 2 (and 2') using the quadratic function $\phi^{(Q)}$ with general constraints. The values of $||A'G - T||_F$ and $\phi^{(Q)}$ at that iteration are included below the plots. For the first iteration of Algorithm 2, most of the g-weights stay close to their initial value of 1. Figure 3a shows a uni-modal distribution of the g-weights after the first iteration with the mode approximately 1.

However, there are small peaks at both ends of the histogram. This is due to some of the calibrated weights being projected to the bounds in Algorithm 2'. As the number of iterations increases, more weights are projected to the bounds of $l = l_i = 0.73$ or $u = u_i = 1.3$ (for all $i = 1, \ldots, n$), leading to a bi-modal distribution with modes at l and u. After the first iteration, there are approximately 5 % of the weights at each bound; however, there are over 30 % of the weights at each bound by the final iteration. As the number of iterations increases, there are fewer weights between l and u. Observe that the value of $||A'G - T||_F$ decreases over subsequent iterations whilst the value of $\phi^{(Q)}$ increases.

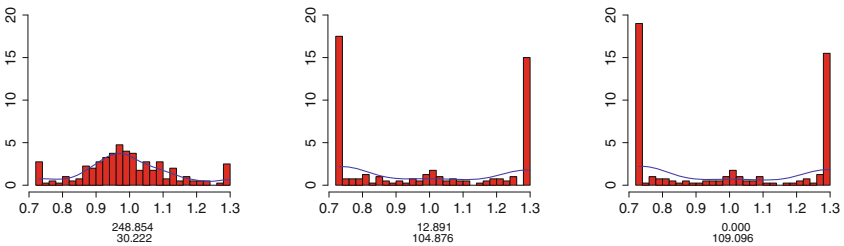

Fig. 3 Histogram (*red*) and density plot (*blue line*) of g-weights for the quadratic function $\phi^{(Q)}$ at iterations 1, 5, and 8 with the values of $||A'G - T||_F$ and $\phi^{(Q)}$ at that iteration. (**a**) g-weights at the first iteration. (**b**) g-weights at the fifth iteration. (**c**) g-weights at the final (eighth) iteration

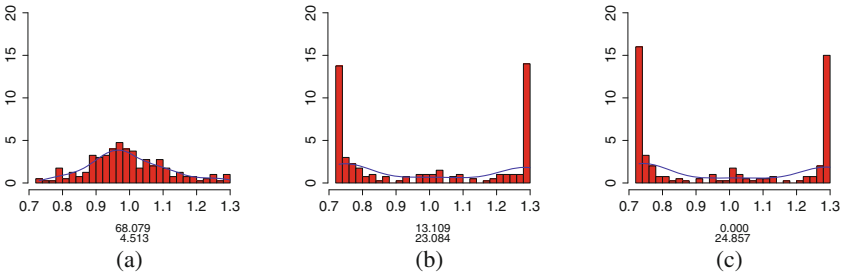

Fig. 4 Histogram (*red*) and density plot (*blue line*) of g-weights for the logit function $\phi^{(L)}$ at iterations 1, 5, and 9 with the values of $||A'G - T||_F$ and $\phi^{(L)}$ at that iteration. (**a**) g-weights at the first iteration. (**b**) g-weights at the fifth iteration. (**c**) g-weights at the final (ninth) iteration

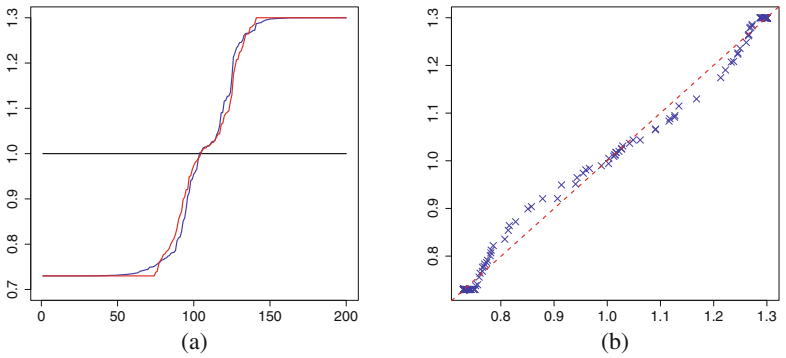

Fig. 5 Investigating g-weights for the quadratic and logit functions in the case of general constraints. (**a**) Plot of g-weights for $\phi^{(Q)}$ (*red*) and $\phi^{(L)}$ (*blue*) against 1 (*black*). (**b**) Scatter-plot of g-weights for $\phi^{(Q)}$ against g-weights for $\phi^{(L)}$

Figure 4 shows histograms and the corresponding density plots of the g-weights at iteration 1 (see Fig. 4a), iteration 5 (see Fig. 4b) and the final, ninth, iteration (see Fig. 4c) of Algorithm 1 using the logit function $\phi^{(L)}$ with general constraints. The values of $||A'G - T||_F$ and $\phi^{(L)}$ at that iteration are included below the plots. After the first iteration, most of the g-weights stay close to their initial value of 1. The density plot for the first iteration shows a uni-modal distribution with the mode approximately 1.

The plot in Fig. 4a is similar to that in Fig. 3b. However, observe that there are fewer weights near the bounds l and u when using the logit function compared to using the quadratic function. For the logit function, no projections to the constraints were required. As we perform subsequent iterations of Algorithm 1, the weights begin to move towards the lower and upper bounds l and u, respectively. This leads to a bi-modal distribution with modes near l and u. Note that the modes are not at l and u, since the infinite derivative of $\phi^{(L)}$ at l and u prevents the g-weights from reaching those values. Once again, we observe that the value of $||A'G - T||_F$ decreases over subsequent iterations whilst the value of $\phi^{(L)}$ increases.

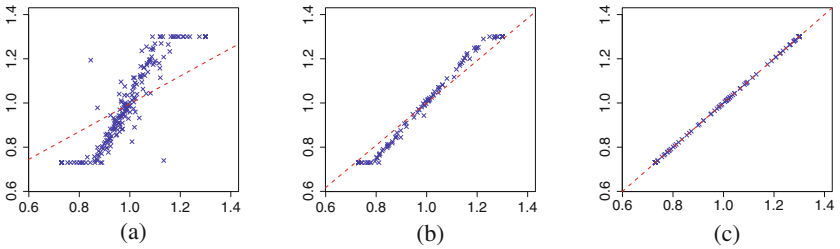

Fig. 6 Scatter-plot of weights for the quadratic function $\phi^{(Q)}$ at the ith iteration compared with the $(i-1)$-th iteration ($i = 2, 6, 8$). (**a**) g-weights at the second iteration against the first iteration. (**b**) g-weights at the sixth iteration against the fifth iteration. (**c**) g-weights at the eighth iteration against the seventh iteration

In Fig. 5a, we plot the weights for the quadratic function $\phi^{(Q)}$ and the logit function $\phi^{(L)}$ ordered by size. The red curve in Fig. 5a corresponds to the weights (ordered by size) obtained using the quadratic function $\phi^{(Q)}$ with Algorithm 2 in the case of general constraints. Observe the horizontal lines at $l = 0.73$ and $u = 1.3$. These show the weights that have been projected to the bounds. In contrast, the blue curve in Fig. 5a, corresponding to the weights obtained using the logit function $\phi^{(L)}$ with Algorithm 1, has $l = 0.73$ and $u = 1.3$ as asymptotes (recall that using Algorithm 1 for $\phi^{(L)}$ with general constraints cannot give weights at the bounds).

Figure 5b shows a scatter-plot of the calibrated weights for the quadratic function $\phi^{(Q)}$ with general constraints using Algorithm 2, against the calibrated weights for the logit function $\phi^{(L)}$ with general constraints using Algorithm 1. These are the weights given in Figs. 3c and 4c, respectively. Observe that there is a cluster of points that are horizontal in the upper right and lower left of the plot. This shows the weights that were projected to the bounds for the quadratic function $\phi^{(Q)}$, but were not able to reach these bounds when the logit function $\phi^{(L)}$ was used.

To further investigate how Algorithms 1 and 2 perform, let us consider how the calibrated weights behave between subsequent iterations of the algorithms. We remark here that scatter-plots of g-weights from the first iteration against the initial g-weights would give a vertical line of points at 1 on the horizontal axis. This is due to the fact that $G^{(0)} = \mathbf{1}$.

Figure 6 shows scatter-plots of the weights using Algorithm 2 (and 2') for the quadratic function $\phi^{(Q)}$ with general constraints at the ith iteration against the weights obtained at the $(i-1)th$ iteration for $i = 2, 6, 8$ (note that 0th iteration here refers to the initial weights). Observe the horizontal lines in Fig. 6a, b. These correspond to weights that were projected to the boundary values of l and u at the i-th iteration, which were not necessarily at the bounds at the $(i-1)$-th iteration. There are many of these weights in Fig. 6a, with fewer in Fig. 6b, and none in Fig. 6c. During the final iteration, there are few changes in the weights, as the algorithm has almost converged to the true solution.

Figure 7 shows scatter-plots of the weights using Algorithm 1 for the logit function $\phi^{(L)}$ with general constraints at the ith iteration against weights at the

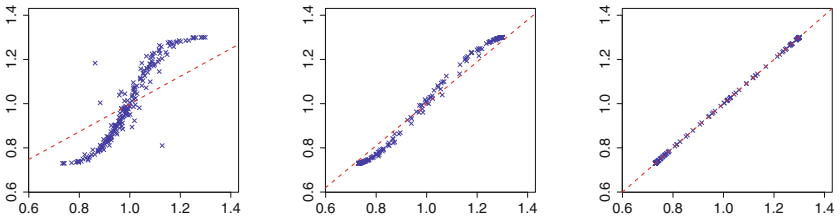

Fig. 7 Scatter-plot of weights for the logit function $\phi^{(L)}$ at the ith iteration compared with the $(i-1)$-th iteration ($i = 2, 6, 9$). (**a**) g-weights at the second iteration against the first iteration. (**b**) g-weights at the sixth iteration against the fifth iteration. (**c**) g-weights at the ninth iteration against the eighth iteration

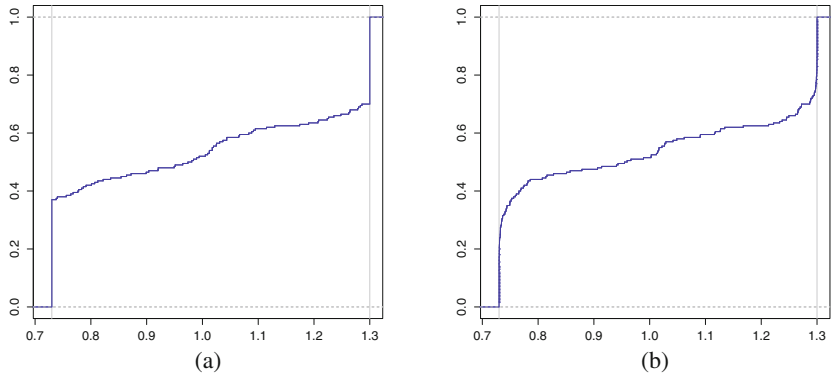

Fig. 8 Empirical cumulative distribution function (ECDF) of the g-weights for functions $\phi^{(Q)}$ and $\phi^{(L)}$. (**a**) ECDF of the g-weights for $\phi^{(Q)}$ using Algorithm 2. (**b**) ECDF of the g-weights for $\phi^{(L)}$ using Algorithm 1

$(i-1)th$ iteration for $i = 2, 6, 9$ (note again that 0th iteration here refers to the initial weights). Unlike the scatter-plots in Fig. 6, the scatter-plots in Fig. 7 do not have horizontal lines of weights. Instead, the arrangement of points resembles an "elongated-S" (see Fig. 7a, b). This curvature of the points shows the weights that have moved nearer the bounds than in the previous iteration. However, these weights have not reached the boundary due to the infinite derivatives of the function $\phi^{(L)}$ at l and u. There is little change in the calibrated weights as the algorithm nears convergence. See Fig. 6c where almost all of the points lie on the main diagonal (see Fig. 7c).

Figure 8a shows the empirical cumulative distribution function (ECDF) for the weights in Fig. 3c; that is, the weights at the final iteration of Algorithm 2 for the quadratic function $\phi^{(Q)}$ with general constraints. Note that the distribution is discontinuous at the lower and upper bounds of $l = 0.73$ and $u = 1.3$, respectively. This is because many of the weights have been projected to the bounds.

In contrast, consider the ECDF in Fig. 8b corresponding to the weights in Fig. 4c for the final iteration of Algorithm 1 using the logit function $\phi^{(L)}$ with general constraints. There is a continuous distribution from l to u, with the distribution curve becoming steeper near the bounds of l and u. This plot further shows that, in the case of the logit function, the weights are tending to but cannot reach the bounds (due to the infinite value of the derivative of the function at the bounds).

Comparison of Calibrated Weights When Projected to the Lower and Upper Bounds

We have seen that the calibrated weights obtained using the logit function $\phi^{(L)}$ often tend towards, but cannot reach, the lower and upper bounds. This is in contrast to the weights obtained using Algorithm 2 for $\phi^{(Q)}$ with general constraints where the algorithm may lead to g-weights that do not satisfy the constraints $L \leq G \leq U$. In this case, Algorithm 2' is used to project any weights back to the nearest bound, before adjusting the remaining weights to satisfy the constraint $A'G = T$.

Let us consider a similar procedure for the logit function $\phi^{(L)}$. In this case, we explore the effect of projecting some of the calibrated weights to the lower bound l and the upper bound u. We then re-calibrate the weights that have not been projected to satisfy the constraint $A'G = T$ (as in Algorithm 2'). In Table 3, we give the function value $\phi^{(L)}$, the Frobenius norm of the distance from constraints $||A'G - T||_F$, and the coefficient of variation (CoV) of calibrated weights in various cases of projection. We remark here that the CoV of the calibrated weights is related to variance of the corresponding calibration estimators. See, for example, [18] for further details.

From Table 3, we can see that the value of the objective function $\phi^{(L)}$ has increased in all cases of projections. Since the function $\phi^{(L)}$ has infinite derivative at l and u, the function increases sharply near the bounds. Hence, any projection of weights is likely to increase the value of the objective function. However, the CoV of the weights is smaller when projections are used than in the case of no projection. As the projection moves the projected weights to the bounds, the remaining weights

Table 3 Values of objective function $\phi^{(L)}$, Frobenius distance from constraints $||A'G - T||_F$, and coefficient of variation (CoV) for various cases of projections of weights to the lower and upper bounds

Projection	$\phi^{(L)}(G)$	$\|\|A'G - T\|\|_F$	CoV
No projection	91.74	3.04×10^{-19}	50.74
The 45 largest and 45 smallest weights	92.43	1.04×10^{-17}	50.50
Weights within 0.05 of l and u	92.94	4.73×10^{-19}	50.38
The lower and upper 25 % of the weights	92.69	9.75×10^{-17}	50.67
Weights below $l/0.975$ and above $0.975u$	92.04	6.11×10^{-12}	50.62

that are re-calibrated move nearer to 1 to account for this adjustment. This leads to a reduced variance and hence reduced CoV of the weights.

Estimating the Total Number of People

Whilst the g-weights play an important role in the calibration problem, the main use of these weights is to estimate some quantity of interest. The properties of this estimator are often more interesting to practitioners than the properties of the weights themselves. We are going to use the calibrated weights to estimate the number of people living in Belgium in 2004. To do this, we take 10,000 random samples from the Belgian municipalities dataset. We shall consider three estimators:

1. The Horvitz–Thompson estimator $Y'D$;
2. The calibration estimator $Y'W$ for the calibrated weights W using Algorithm 2 with the quadratic function $\phi^{(Q)}$ in the case of general constraints; and
3. The calibration estimator $Y'W$ for the calibrated weights W using Algorithm 1 with the logit function $\phi^{(L)}$ in the case of general constraints.

For each estimator, we consider properties of the estimates obtained when taking simple random samples of size 75, 100, and 200.

Figure 9 shows the distribution of the estimates for the true value of 10,417,122 when using the Horvitz–Thompson estimator $Y'D$ using samples of size $n = 75$ (Fig. 9a), $n = 100$ (Fig. 9b), and $n = 200$ (Fig. 9c). As expected, the distribution has a smaller variance as the sample size is increased. Observe that the distribution of estimates using the Horvitz–Thompson estimator is skewed to the left, with the mode of the distribution to the left of the true value. This skewness reduces as the sample size increases.

From the estimates shown in Fig. 9, we compute the mean, bias, median, variance, and MSE of the estimates. These values are given in Table 4. Surprisingly,

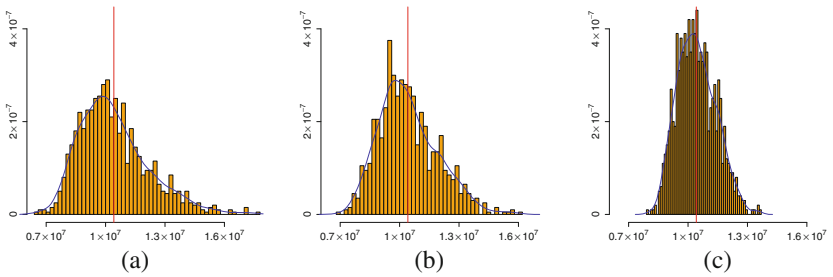

Fig. 9 Histogram (*orange*) and density plot (*blue line*) of estimates of the total number of people in Belgium in 2004 using g-weights from 10,000 random samples of size 75, 100, and 200 using the Horvitz–Thompson estimator; actual total = 10,417,122 (*red line*). (**a**) Sample size $n = 75$. (**b**) Sample size $n = 100$. (**c**) Sample size $n = 200$

Table 4 Values of mean, bias, variance, and mean-squared error (MSE) for the estimates of the total number of people in Belgium in 2004 using 10,000 random samples of size 75, 100, and 200 using the Horvitz–Thompson estimator; actual total = 10,417,122

Horvitz–Thompson	$n = 75$	$n = 100$	$n = 200$
Mean	10,425,081	10,439,499	10,437,165
Bias	7959	22,377	20,043
Median	10,146,880	10,236,770	10,355,954
Variance	3.10475×10^{12}	2.21299×10^{12}	9.38258×10^{11}
MSE	3.10481×10^{12}	2.21349×10^{12}	9.38660×10^{11}

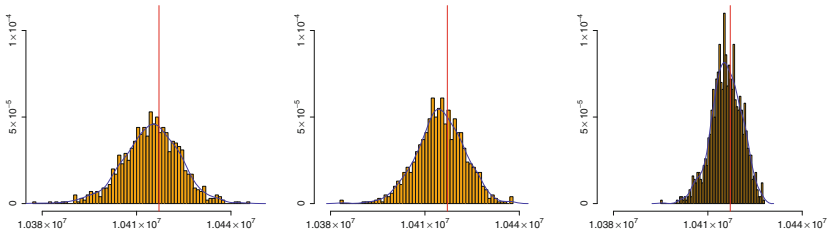

Fig. 10 Histogram (*orange*) and density plot (*blue line*) of estimates of the total number of people in Belgium in 2004 using g-weights from 10,000 random samples of size 75, 100, and 200 for the quadratic function $\phi^{(Q)}$; actual total = 10,417,122 (*red line*). (**a**) Sample size $n = 75$. (**b**) Sample size $n = 100$. (**c**) Sample size $n = 200$

the bias is smallest when using the smallest sample size. However, the variance and MSE of the estimates decrease as the sample size increases, as expected. The median of the weights moves closer to the true value as the sample size increases.

Figure 10 shows the distribution of the calibrated estimates for the true value of 10,417,122 using Algorithm 2 for the quadratic function $\phi^{(Q)}$ with general constraints. We take 10,000 random samples of size $n = 75$ (Fig. 10a), $n = 100$ (Fig. 10b), and $n = 200$ (Fig. 10c). The range of the estimates in this case is approximately 170 times smaller than the range of the estimates using the Horvitz–Thompson estimator. Again, the distribution has a smaller variance as the sample size is increased. The distribution of the estimates in this case is less skewed than for the Horvitz–Thompson estimator, with the mode of the estimates close to the true value in all cases.

From the estimates shown in Fig. 10, we compute the mean, bias, median, variance, and MSE of the estimates. These values are given in Table 5. The bias, variance, and MSE all decrease as the sample size increases. These values are smaller than the corresponding values for the Horvitz–Thompson estimator. The median of the estimates is also near the true value in all cases, providing a better estimate of the true value than the corresponding medians using the Horvitz–Thompson estimator.

Table 5 Values of mean, bias, variance, and mean-squared error (MSE) for the estimates of the total number of people in Belgium in 2004 using 10,000 random samples of size 75, 100, and 200 for the quadratic function $\phi^{(Q)}$; actual total = 10,417,122

$\phi^{(Q)}$	$n = 75$	$n = 100$	$n = 200$
Mean	10,414,658	10,414,928	10,415,853
Bias	−2464	−2194	−1269
Median	10,414,893	10,415,008	10,415,837
Variance	8.14258×10^7	5.83535×10^7	2.38957×10^7
MSE	8.74984×10^7	6.31651×10^7	2.55059×10^7

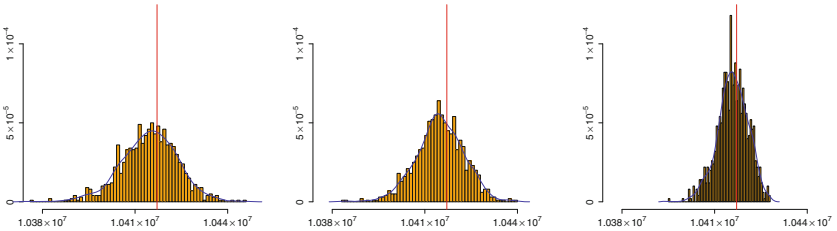

Fig. 11 Histogram (*orange*) and density plot (*blue line*) of estimates of the total number of people in Belgium in 2004 using g-weights from 10,000 random samples of size 75, 100, and 200 for the quadratic function $\phi^{(L)}$; actual total = 10,417,122 (*red line*). (**a**) Sample size $n = 75$. (**b**) Sample size $n = 100$. (**c**) Sample size $n = 200$

Figure 11 shows the distribution of the calibrated estimates for the true value of 10,417,122 using Algorithm 1 for the logit function $\phi^{(L)}$ with general constraints. We take 10,000 random samples of size $n = 75$ (Fig. 11a), $n = 100$ (Fig. 11b), and $n = 200$ (Fig. 11c). Again, the range of the estimators in this case is approximately 170 times smaller than the range for the estimates using the Horvitz–Thompson estimator. The range of the estimates is also slightly smaller than the range of the estimates for the quadratic function $\phi^{(Q)}$. The figures show that the distribution of the estimates has smaller variance as the sample size is increased. The distribution of the estimates in this case is less skewed than for the Horvitz–Thompson estimator, with the mode of the estimates close to the true value in all cases.

From the estimates shown in Fig. 11, we compute the mean, bias, median, variance, and MSE of the estimates. This information is given in Table 6. The bias, variance, and MSE all decrease as the sample size increases. These values are also smaller than the corresponding values for the Horvitz–Thompson estimator. The median of the estimates is also closer to the true value in all cases, being a much better estimate of the true value than the corresponding medians using the Horvitz–Thompson estimator. We remark that the values for the mean, bias, median, variance, and MSE are very similar to the values obtained using the estimates for the quadratic function $\phi^{(Q)}$. This agrees with the assertion in [9] that,

Table 6 Values of mean, bias, variance, and mean-squared error (MSE) for the estimates of the total number of people in Belgium in 2004 using 10,000 random samples of size 75, 100, and 200 for the quadratic function $\phi^{(L)}$; actual total = 10,417,122

$\phi^{(L)}$	$n = 75$	$n = 100$	$n = 200$
Mean	10,414,669	10,414,918	10,415,888
Bias	−2453	−2204	−1234
Median	10,415,042	10,414,850	10,415,894
Variance	8.27775×10^7	5.93664×10^7	2.40143×10^7
MSE	8.87961×10^7	6.42240×10^7	2.55367×10^7

under certain regularity conditions, the calibration estimators are asymptotically equivalent independent of the chosen penalty function.

In this section, we have seen that the Horvitz–Thompson estimator gives estimates with a larger variance than for the calibration estimators. This supports the second motivation for calibration from the introduction that the use of calibration can lead to a reduction in the sampling variance of the estimates. In this example, the range of the Horvitz–Thompson estimates was approximately 170 times larger than the range of the calibrated estimates. The calibrated estimates generally perform better for estimating the true value than the estimates using the Horvitz–Thompson estimator (despite the Horvitz–Thompson estimator being unbiased, see [15]). However, in this example, we saw that the choice of the penalty function had little effect on the properties of the resulting estimates.

Example 3: Convergence Properties of Algorithms 1 and 2 for Various Choices of $H^{(s)}$

The labour force survey (LFS) is arguably one of the most important social surveys conducted by the Office for National Statistics (ONS). It is a household survey, where participants are asked about labor force characteristics and related topics. One labor force characteristic of interest is employment status, since this key information is used to calculate estimates of the UK employment and unemployment rate, statistics that are of interest to government, businesses, and even the general public.

Since 1992, the LFS has been carried out quarterly. Each selected household remains part of the sample for five consecutive quarters. This means one fifth of the sample needs to be replaced every quarter. The original purpose of the survey was to investigate characteristics for cross-sectional data. However, since households are retained in the survey for five consecutive quarters, it was recognized that the LFS could also be used to investigate characteristic changes of individuals across

quarters. Sample members who respond in all five consecutive quarters can be linked and combined to give the so-called five-quarterly longitudinal LFS datasets.

The quarterly LFS began, in its current form, during the Spring of 1992. However, the rotating panel design of the sample (i.e., retaining households for five consecutive quarters and updating a fifth of the sample each quarter) was not established until Spring 1993, with the first five-quarterly LFS dataset considering households over the five-quarter period from Spring 1993 to Spring 1994. Every quarter, a new five-quarterly LFS dataset is produced as another cross-sectional dataset becomes available.

However, combining five consecutive quarters' worth of cross-sectional datasets, and including only those sample members who responded in all five quarters, can lead to methodological issues and result in a sample that is unrepresentative of the true population. These issues can be classified by two main problems: firstly, this linking of the five datasets can result in bias due to non-response and attrition of the sample; secondly, there is the issue of bias that can occur due to response errors, since these can have a major effect on the estimates of changes in the characteristics of interest. See [33] for details of existing methodologies to deal with these issues.

The primary purpose of the longitudinal datasets is to produce estimates of flows, i.e. changes in characteristics over the five-quarterly period. In particular, the labor force flows are of particular interest, since these show patterns of people moving between states of employment, unemployment, and inactive as well as highlighting changes in the numbers of those who are of working age. For the five-quarterly LFS datasets, the flow characteristics can be considered as measuring the change in characteristics over a 12-month period.

As the dataset in consideration is used for forming estimates related to the working age population, that is, all males and females who are aged 16–69, only sample members who responded to all five waves of the LFS who were aged between 15 and 69 at wave 1 are included.

In this example, we consider the five-quarterly longitudinal dataset for the five quarters from April 2012 to June 2013 (see [25]). The working age population is estimated to be 44,443,746 from census data. There are 4538 sample members in the dataset. For each of these sample members, we calibrate on 61 constraints. These constraints include satisfying known population totals in 28 age–sex categories, 18 region groups, and ensuring that the estimate of numbers of people in the three employment statuses (employed, unemployed, and inactive) matches each of the totals for each of the five quarters used to form the dataset.

Therefore, for this dataset, we have $n = 4538$ and $m = 61$. The main purpose of the calibration here is consistency between the estimates of the population totals for each of the 61 constraints and their known totals. However, calibration can also help to deal with biases arising from non-response and sample design (see, for example, [22]).

Forming estimates of changes in employment status forms an important part of government policy, since the statistics produced highlight how employment has changed over the 12-month period. The estimates also show the numbers of those

moving into and out of working age, which forms a key basis for government policy on pensions and retirement age.

To obtain estimates of the changes in employment status, calibration is used to assign an appropriate weight to each sample member. Extreme weights are undesirable here, since that could lead to certain flow rates being over-estimated if certain sample members dominate. Much time is spent in designing the survey and sampling scheme to ensure that the sample is as representative as possible of the true population. Allowing weights to move close to 0 means that the corresponding sample member's contribution to the survey is removed. It would, effectively, have not been worth the cost of interviewing them and observing their characteristics, since they contribute very little to the subsequent estimates. This argument can also be applied to negative weights.

Therefore, taking all of this into account, we wish to estimate the employment flows such that

1. There is a close match between the sample estimates and known population totals;
2. There are non-extreme weights, i.e. not too large so that sample members dominate, and not too small or negative so that sample members are effectively unimportant to the estimates; and
3. The estimate of the flows is reliable, in a sense that we shall define later.

To begin, let us consider all possible combinations of algorithms and functions we have introduced throughout this chapter. We consider the following five cases:

1. Algorithm 1a for function $\phi^{(Q)}$ with no constraints;
2. Algorithm 1 for function $\phi^{(R)}$ with non-negativity constraints;
3. Algorithm 2 (and 2') for function $\phi^{(Q)}$ with general constraints;
4. Algorithm 2 (and 2') for function $\phi^{(R)}$ with general constraints; and
5. Algorithm 1 for function $\phi^{(L)}$ for general constraints.

For this example, we consider general constraints with $l_i = 0.5$ and $u_i = 2.4$ for all $i = 1, \ldots, n$.

In Table 7, we consider convergence of the algorithms in all five cases listed above when using the Newton–Raphson form of the matrix $H^{(s)}$. We give the number of iterations (Iter), the minimum and maximum values of the g-weights obtained, as well as the mean, standard deviation (SD), coefficient of variation (CoV), skewness (Skew.), and kurtosis (Kurt.) of the weights. We also list the 1st-percentile (1st %), the median (med), and the 99th-percentile (99th %) of the weights in each case.

Observe that there are negative g-weights for the function $\phi^{(Q)}$ with no constraints. There were six negative weights in this case. This is undesirable since we do not want sample members to be under-represented in the estimates. Use of the raking function with non-negativity constraints has resulted in g-weights as large as 4.85. This is undesirable since we do not want individual sample members to dominate in the estimates. We observe that Algorithm 2 for $\phi^{(R)}$ with general constraints failed to converge (after running the algorithm for 100,000 iterations). The general

Table 7 Table comparing convergence properties of the calibrated g-weights using the Newton–Raphson version of $H^{(s)}$

ϕ	Constr.	Iter	Min g	Max g	Mean	SD	CoV	Skew.	Kurt.	1st %	Med.	99th %
$\phi^{(Q)}$	No	1	−0.087	3.173	0.989	0.491	49.629	1.262	1.654	0.213	0.910	2.428
$\phi^{(R)}$	Non-neg	4	0.305	4.849	0.990	0.499	50.479	1.952	5.463	0.419	0.892	2.572
$\phi^{(Q)}$	Gen.	5	0.500	2.400	0.990	0.498	50.329	1.303	1.167	0.500	0.887	2.400
$\phi^{(R)}$	Gen.					No convergence						
$\phi^{(L)}$	Gen.	6	0.502	2.399	0.990	0.505	50.981	1.389	1.036	0.509	0.837	2.367

constraints $l_i = 0.3$ and $u_i = 4$ were required so that Algorithm 2 converged in this case. In practice, the general constraints for the raking function need to be less "severe" than for the quadratic or raking functions, in the sense of a smaller lower bound and/or a larger upper bound. This is due to the nature of the derivative of $\phi^{(R)}$ (as plotted in Fig. 1b). The resulting h-function is such that its domain is smaller than for the quadratic and logit functions, despite having the same range.

There are several weights at the bounds in the case of the quadratic function, since the minimum and 1st-percentiles, and maximum and 99th-percentile are equal to the lower and upper bounds, respectively. In contrast, the minimum value of the g-weights for the logit function is greater than the lower bound, and the maximum g-weight is less than the upper bound.

In Table 8, we consider convergence of the algorithms in all five cases listed above when using the g-weights form of the matrix $H^{(s)}$. We give the number of iterations (Iter), the minimum and maximum values of the g-weights obtained, as well as the mean, standard deviation (SD), coefficient of variation (CoV), skewness (Skew.), and kurtosis (Kurt.) of the weights. We also list the 1st-percentile (1st %), the median (med), and the 99th-percentile (99th %) of the weights in each case.

Observe that the cases of the quadratic function with no constraints and the raking function with non-negative constraints lead to the same results as in Table 7. However, convergence for both the quadratic and logit functions with general constraints took more iterations than in the case of the Newton–Raphson form of $H^{(s)}$. The algorithm has converged to the same place and the solutions are virtually identical in both cases.

In Table 9, we consider convergence of the algorithms in all five cases listed above when using the g-weights form of the matrix $H^{(s)}$. We give the number of iterations (Iter), the minimum and maximum values of the g-weights obtained, as well as the mean, standard deviation (SD), coefficient of variation (CoV), skewness (Skew.), and kurtosis (Kurt.) of the weights. We also list the 1st-percentile (1st %), the median (med), and the 99th-percentile (99th %) of the weights in each case.

In this case, the quadratic function with no constraints gives the same results as in Tables 7 and 8. The quadratic function with general constraints converged in the same number of iterations as for the Newton–Raphson method considered in Table 7, since the matrices $H^{(s)}$ are equivalent in both cases. However, the algorithm failed to converge for the raking function in both the non-negativity and general constraint cases. The logit function with general constraints took longer to converge than for the Newton–Raphson form of the matrix $H^{(s)}$ considered in Table 7; however, using the identity matrix form of $H^{(s)}$ took less iterations to converge than for the g-weights form of $H^{(s)}$ considered in Table 8.

For a general discussion on measuring numerical complexity of an optimization problem see [23, 42, 43]. For a discussion on choosing test functions for optimization problems see [41].

In Fig. 12, we plot a box-plot of the calibrated weights considered in Table 7. We also include the calibrated weights for the quadratic and logit functions using the general constraints $l_i = 0.3$ and $u_i = 3$ (the raking function with general constraints failed to converge in this case).

Table 8 Table comparing convergence properties of the calibrated g-weights using the g-weights version of $H^{(s)}$

ϕ	Constr.	Iter	Min g	Max g	Mean	SD	CoV	Skew.	Kurt.	1st %	Med.	99th %
$\phi^{(Q)}$	No	1	−0.087	3.173	0.989	0.491	49.629	1.262	1.654	0.213	0.910	2.428
$\phi^{(R)}$	Non-neg	4	0.305	4.849	0.990	0.499	50.479	1.952	5.463	0.419	0.892	2.572
$\phi^{(Q)}$	Gen.	15	0.500	2.400	0.989	0.4983	50.329	1.303	1.167	0.500	0.887	2.400
$\phi^{(R)}$	Gen.				No convergence							
$\phi^{(L)}$	Gen.	75	0.502	2.399	0.990	0.505	50.979	1.389	1.036	0.509	0.837	2.367

Table 9 Table comparing convergence properties of the calibrated g-weights using the identity matrix version of $H^{(s)}$

ϕ	Constr.	Iter	Min g	Max g	Mean	SD	CoV	Skew.	Kurt.	1st %	Med.	99th %
$\phi^{(Q)}$	No	1	−0.087	3.173	0.989	0.491	49.629	1.262	1.654	0.213	0.909	2.428
$\phi^{(R)}$	Non-neg				No convergence							
$\phi^{(Q)}$	Gen.	5	0.500	2.400	0.990	0.498	50.329	1.303	1.167	0.500	0.887	2.400
$\phi^{(R)}$	Gen.				No convergence							
$\phi^{(L)}$	Gen.	55	0.502	2.399	0.990	0.505	50.978	1.389	1.036	0.509	0.837	2.367

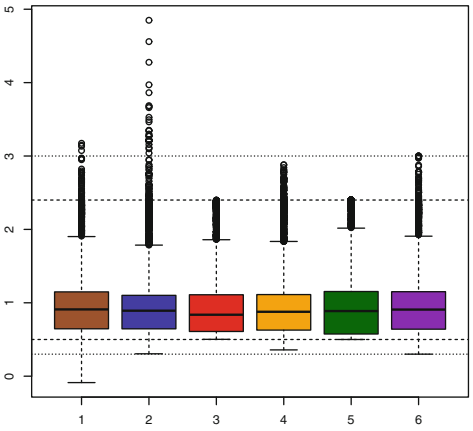

Fig. 12 Box-plot of g-weights for: *1.* $\phi^{(Q)}$ with no constraints, *2.* $\phi^{(R)}$ with non-negativity constraints, *3.* $\phi^{(L)}$ with general constraints ($l = 0.5$ and $u = 2.4$), *4.* $\phi^{(L)}$ with general constraints ($l = 0.3$ and $u = 3$), *5.* $\phi^{(Q)}$ with general constraints ($l = 0.5$ and $u = 2.4$), and *6.* $\phi^{(Q)}$ with general constraints ($l = 0.3$ and $u = 3$)

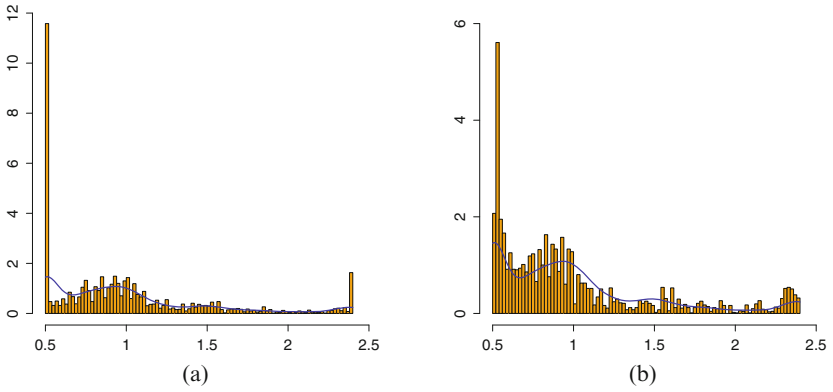

Fig. 13 Histogram (*orange*) and density plot (*blue line*) of the calibrated weights for the truncated quadratic function $\phi^{(Q)}$ and the logit function $\phi^{(L)}$ for the general constraints $l_i = 0.5$ and $u_i = 2.4$. (**a**) g-weights using Algorithm 2 with $\phi^{(Q)}$. (**b**) g-weights using Algorithm 1 with $\phi^{(L)}$

In Fig. 13, we plot histograms of the weights obtained using Algorithm 2 for function $\phi^{(Q)}$ (Fig. 13a) and Algorithm 1 for function $\phi^{(L)}$ (Fig. 13b) in the case of the general constraints $l_i = 0.5$ and $u_i = 2.4$. These correspond to cases 5 and 3, respectively, in Fig. 12. From Fig. 13a, we can see that many of the weights have been projected to the boundaries. In contrast, Fig. 13a shows that many g-weights approach, but do not reach, the bounds. This supports our comments from Example 2 in section "Example 2: Investigation of Calibrated Weights at Each Iteration of Algorithms 1 and 2".

In Fig. 14, we plot histograms of the weights using Algorithm 2 for function $\phi^{(Q)}$ (Fig. 14a) and Algorithm 1 for function $\phi^{(L)}$ (Fig. 14b), both in the case of

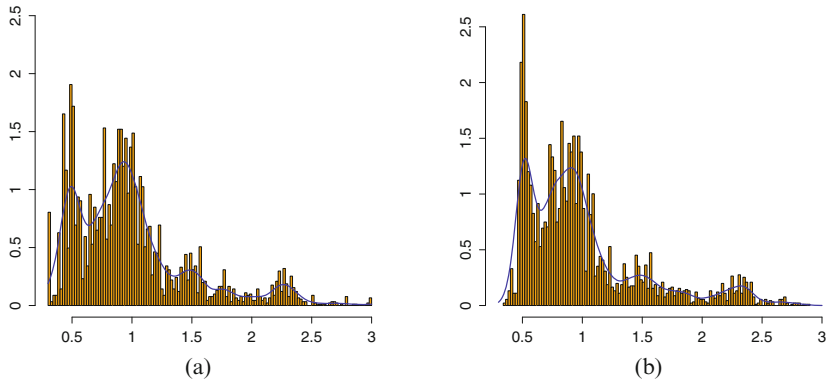

Fig. 14 Histogram (*orange*) and density plot (*blue line*) of the calibrated weights for the truncated quadratic function $\phi^{(Q)}$ and the logit function $\phi^{(L)}$ for the general constraints $l_i = 0.3$ and $u_i = 3$. (**a**) g-weights using Algorithm 2 with $\phi^{(Q)}$. (**b**) g-weights using Algorithm 1 with $\phi^{(L)}$

the general constraints $l_i = 0.3$ and $u_i = 3$. These correspond to cases 6 and 4, respectively, in Fig. 12. The histogram in Fig. 14b shows that there are fewer weights with values at the lower and upper bounds. Contrast this to Fig. 14a, where the histogram shows that there are many weights with values at the lower and upper bounds.

For the histograms in Fig. 13, most of the weights are at or near the boundaries, with few weights in between. However, as the difference between the bounds is increased, fewer weights move towards these bounds. The resulting distribution is uni-modal or bi-modal depending on the nature of the imposed bounds.

We can summarize this example as follows:

Algorithm 1a for Function $\phi^{(Q)}$ with No Constraints All three forms of $H^{(s)}$ are equivalent and the algorithm gives the same solution.

Algorithm 1 for Function $\phi^{(R)}$ with Non-negativity Constraints The Newton–Raphson and g-weights form of the matrix $H^{(s)}$ are the same and the algorithms converge to the same solution in the same number of iterations for both cases. However, the algorithms failed to converge in the case of the identity matrix.

Algorithm 2 (and 2') for Function $\phi^{(Q)}$ with General Constraints The algorithm converged in the fewest iterations in the case of the Newton–Raphson and identity forms of $H^{(s)}$. The algorithm converged to the same solution using the g-weights form of $H^{(s)}$, but took more iterations to converge.

Algorithm 2 (and 2') for Function $\phi^{(R)}$ with General Constraints There was no convergence for any case of $H^{(s)}$. This is likely to be due to the general constraints being too restrictive for this choice of function.

Algorithm 1 for Function $\phi^{(L)}$ for General Constraints Converged in the fewest iterations for the Newton–Raphson version of $H^{(s)}$. The identity matrix version of

$H^{(s)}$ was the second fastest in terms of the number of iterations, with the g-weights version of $H^{(s)}$ requiring the most iterations to converge.

Calibration Packages

We now present details of the various packages that perform calibration, along with the algorithms and calibration functions they implement.

Summary of the Calibration Packages

Table 10 gives a summary of some of the packages that implement the calibration algorithms presented in this chapter. In the table below, we refer to the three functions ϕ presented in section "The Function ϕ", namely the quadratic function $\phi^{(Q)}$, the raking function $\phi^{(R)}$, and the logit function $\phi^{(L)}$. For the various functions, the packages implement the different cases of the g-weight constraints outlined in section "Additional $\boldsymbol{g}$-Weight Constraints". In Table 10, we refer to each of the constraint cases by the letters (a), (b), and (c), corresponding to the labels used in the list given in section "Additional $\boldsymbol{g}$-Weight Constraints". The case (a) refers to no constraints, case (b) refers to non-negativity constraints, and case (c) refers to general constraints. We also refer to the three versions of the matrix $H^{(s)}$ as introduced in section "The Matrix $H^{(s)}$". Recall that the three versions are the Newton–Raphson matrix, identity matrix, and matrix with g-weights on the diagonal (g-weights matrix).

Details of the Calibration Packages

This section expands on the information presented in Table 10 of section "Summary of the Calibration Packages". All packages give the option to use Algorithm 1a (see section "Algorithm 1a: Quadratic Function $\phi^{(Q)}$ with No Constraints") for the quadratic function with no constraints. We provide an overview of each of the programs and the algorithms they implement.

Calib This is a function that is part of the "sampling" package [35] in R. It uses Algorithm 1 for the functions $\phi^{(L)}$ and $\phi^{(R)}$, and Algorithm 2 (and 2') for the function $\phi^{(Q)}$. The algorithm uses the g-weights form of $H^{(s)}$ for the logit and raking functions and the identity matrix form of $H^{(s)}$ in the case of the quadratic function.

Calibrate This is an R function contained within the "survey" package [21]. The package has the options to use Algorithm 1 for the functions $\phi^{(L)}$ and $\phi^{(R)}$ in the case

Table 10 Summary of the main calibration packages and the algorithms they implement

Package	Program	Functions	Constraints	Matrix H
CALIB	R	$\phi^{(Q)}$	(a) and (c)	Identity matrix
	(sampling)	$\phi^{(R)}$	(b)	g-Weights matrix
		$\phi^{(L)}$	(c)	g-Weights matrix
CALIBRATE	R	$\phi^{(Q)}$	(a) and (c)	Newton–Raphson
	(survey)	$\phi^{(R)}$	(b) and (c)	Newton–Raphson
		$\phi^{(L)}$	(c)	Newton–Raphson
ReGenesees	R	$\phi^{(Q)}$	(a) and (c)	Newton–Raphson
		$\phi^{(R)}$	(b)	Newton–Raphson
		$\phi^{(L)}$	(c)	Newton–Raphson
CALMAR	SAS	$\phi^{(Q)}$	(a) and (c)	Newton–Raphson
CALMAR 2		$\phi^{(R)}$	(b)	Newton–Raphson
		$\phi^{(L)}$	(c)	Newton–Raphson
g-CALIB-S	SPSS	$\phi^{(Q)}$	(a) and (c)	Newton–Raphson
		$\phi^{(R)}$	(b)	Newton–Raphson
		$\phi^{(L)}$	(c)	Newton–Raphson
GES	SAS	$\phi^{(Q)}$	(a) and (c)	Projection method algorithm
BASCULA	Blaise	$\phi^{(Q)}$	(a) and (c)	Scale modified quadratic

of general constraints and non-negativity constraints, respectively, and Algorithm 2 (and 2') for the functions $\phi^{(Q)}$ and $\phi^{(R)}$ in the case of general constraints. The Newton–Raphson form of $H^{(s)}$ is implemented in all cases. There is the option for the user to input his/her own calibration functions; however, this requires stating the function h, i.e. the inverse of the derivative of the function ϕ.

ReGenesees This is an R package [39] developed by the Italian Statistical Office, ISTAT, that implements the function ecalibrate. The user may choose from the quadratic, raking, and logit functions. Algorithm 1 is performed for functions $\phi^{(L)}$ and $\phi^{(R)}$ in the case of general constraints and non-negativity constraints, respectively, whilst Algorithms 2 (and 2') are used for the quadratic and raking functions $\phi^{(Q)}$ and $\phi^{(L)}$ with general constraints. The Newton–Raphson form of the matrix $H^{(s)}$ is implemented for all functions.

CALMAR The software CALMAR (CALibration of MARgins) is used by the Office for National Statistics in the UK, the Central Statistical Office in Ireland, and many other statistical offices throughout the world. The package was developed by Sautory [29] at the French National Institute for Statistics and Economic Studies (INSEE). The software uses a SAS Macro [30] to perform Algorithm 1 for functions $\phi^{(L)}$ and $\phi^{(R)}$ in the case of general constraints and non-negativity constraints, respectively, as well as Algorithms 2 (and 2') for function $\phi^{(Q)}$ with general constraints. The Newton–Raphson form of $H^{(s)}$ is used for all functions.

CALMAR2 This is a modified version of the SAS macro CALMAR introduced above. Sautory [20] enhanced several aspects of the original CALMAR code,

allowing the user to perform simultaneous calibration at different levels of a survey, and use generalized calibration adjustment for total non-response (see [31] for further information). CALMAR2 also includes a new function referred to as the hyperbolic sine function. It is not widely used in practice, and so has not been discussed in this chapter. CALMAR2 is used at INSEE, as well as several other statistical offices worldwide. With the exception of the new hyperbolic sine function (which we do not discuss here), the functions and algorithms implemented are as described for CALMAR.

g-CALIB-S This is an SPSS package developed at Statistics Belgium. The package is very similar to CALMAR, in that Algorithm 1 may be used for functions $\phi^{(L)}$ and $\phi^{(R)}$ in the case of general constraints and non-negativity constraints, respectively, as well as Algorithms 2 (and 2') for function $\phi^{(Q)}$ with general constraints. The Newton–Raphson form of $H^{(s)}$ is used for all functions. This package is well documented in [37].

GES The generalized estimation software (GES) was developed by Statistics Canada, and uses a projection method algorithm (see section "Other Algorithms") for the quadratic function $\phi^{(Q)}$ with general constraints (see [12] for a more detailed discussion). The package is used by the Office for National Statistics (ONS) in the analysis of quarterly LFS datasets. We note, however, that for the longitudinal five-quarterly datasets as considered in section "Example 3: Convergence Properties of Algorithms 1 and 2 for Various Choices of $H^{(s)}$", the ONS currently uses CALMAR rather than GES.

The paper of Brodie and Cotterell [5] compares the GES algorithm with a so-called new algorithm. This algorithm is equivalent to Algorithm 2 (and 2'). It is shown in [5] that the "new" algorithm converges in fewer iterations than the GES algorithm. They also show that, when the bounds are tightened, the time taken for the "new" algorithm to converge decreases. This is a consequence of the algorithm setting more weights to the bounds at earlier iterations.

It is argued in [5] that an "obvious" advantage of the CALMAR algorithm is that it gives a solution that lies entirely within or on the boundary values. Therefore, the bounds will be satisfied exactly. However, for the projection method algorithm used by GES, the bounds are not met exactly and are only satisfied to within the convergence level specified by the user. Despite this, it is argued that when the calibration problem with bounds is not solvable, the projection method algorithm is better than the "new" algorithm, in that the projection method algorithm will give an approximate solution. However, caution should be taken here as the algorithm may not have given a valid approximation to the solution of the calibration problem in this case.

It could also be argued that soft calibration is a useful method of obtaining an approximate solution when the hard calibration problem cannot be solved. See Sect. 5 of [8] for further information regarding soft calibration.

BASCULA This program implements the scale modified quadratic algorithm attributed to [16] for the function $\phi^{(Q)}$ with general constraints. This algorithm is

equivalent to Method 3 considered in [32]. The package was developed by Statistics Netherlands and is well documented in the Bascula 4.0 User Guide [24]. There has been criticism of the convergence properties of this algorithm (and hence the BASCULA package), see, for example, the technical paper [7].

Conclusion

In this chapter, we have considered two of the most common algorithms for performing calibration in survey sampling. In the case of no constraints, the quadratic function can lead to negative and/or extreme weights. In the case of non-negative constraints, the raking function can lead to extreme weights. This motivates the need for general constraints on the g-weights. The logit function $\phi^{(L)}$ can automatically take these constraints into account. However, in practice, algorithms using this function may converge slowly when the weights approach the bounds imposed by these constraints.

For the quadratic and raking functions, Algorithm 2 is required in the case of general constraints. This involves projecting any of the g-weights not satisfying the constraints so that they satisfy these constraints. We have seen that Algorithm 2 for function $\phi^{(R)}$ with general constraints does not perform well in terms of convergence, performing much worse than Algorithm 2 when using the quadratic function $\phi^{(Q)}$ with general constraints. The algorithms using the quadratic or logit functions often converge for more restrictive bounds than for the raking function. This is due to the shape of the functions and their derivatives.

We have considered various versions of the Jacobian matrix used in the method for determining the Lagrange multipliers. It is not surprising that the true Jacobian gave the fastest convergence in all cases. However, the g-weights form of the Jacobian was no better than using the initial Jacobian in the case of the quadratic function with general constraints. The g-weights form of the Jacobian performed worse than simply using the initial Jacobian in the case of the logit function with general constraints. It is therefore recommended that the g-weights version of the Jacobian matrix should not be used as a substitute for the Newton–Raphson matrix.

Acknowledgements Work of the first author was supported by the Engineering and Physical Sciences Research Council, project number 1638853 "Examination of approaches to calibration and weighting for non-response in cross-sectional and longitudinal surveys." The work of the third author was supported by the Russian Science Foundation, project No. 15-11-30022 "Global optimization, supercomputing computations, and application".

References

1. Bankier, M., Rathwell, S., Majkowski, M., et al.: Two step generalized least squares estimation in the 1991 Canadian census. In: Proceedings of the Workshop on Uses of Auxiliary Information in Surveys (1992)

2. Bardsley, P., Chambers, R.: Multipurpose estimation from unbalanced samples. Appl. Stat. **33**(3), 290–299 (1984)
3. Bocci, J., Beaumont, C.: Another look at ridge calibration. Metron **66**(1), 5–20 (2008)
4. Brewer, K.: Cosmetic calibration with unequal probability sampling. Surv. Methodol. **25**(2), 205–212 (1999)
5. Brodie, P., Cotterell, B.: Reducing or eliminating the effects of extreme calibration weights in social surveys. In: Proceedings, 3rd European Conference on Quality in Survey Statistics, Cardiff (2006)
6. Chambers, R.: Robust case-weighting for multipurpose establishment surveys. J. Off. Stat. **12**(1), 3–32 (1996)
7. Chauvet, G., Deville, J., El Haj Tirari, M., Le Guennec, J.: Evaluation de trois logiciels de calage: g-CALIB 2.0, CALMAR 2 et BASCULA 4.0. Tech. rep., Statistics Belgium (2005)
8. Davies, G., Gillard, J., Zhigljavsky, A.: Calibration in survey sampling as an optimization problem. In: Optimization, Control, and Applications in the Information Age, pp. 67–89. Springer, Cham (2015)
9. Deville, J.C., Särndal, C.E.: Calibration estimators in survey sampling. J. Am. Stat. Assoc. **87**(418), 376–382 (1992)
10. Deville, J.C., Särndal, C.E., Sautory, O.: Generalized raking procedures in survey sampling. J. Am. Stat. Assoc. **88**(423), 1013–1020 (1993)
11. Estevao, V.: Calculation of g-weights under calibration and bound constraints. Tech. rep., Statistics Canada (1994)
12. Estevao, V., Hidiroglou, M., Särndal, C.: Methodological principles for a generalized estimation system at statistics Canada. J. Off. Stat. **11**(2), 181–204 (1995)
13. Fuller, W., Loughin, M., Baker, H.: Regression weighting in the presence of nonresponse with application to the 1987–1988 national food consumption survey. Surv. Methodol. **20**, 75–85 (1994)
14. Han, S.P.: A successive projection method. Math. Program. **40**(1), 1–14 (1988)
15. Horvitz, D.G., Thompson, D.J.: A generalization of sampling without replacement from a finite universe. J. Am. Stat. Assoc. **47**(260), 663–685 (1952)
16. Huang, E., Fuller, W.: Nonnegative regression estimation for sample survey data. In: Proceedings of the Social Statistics Section, vol. 21, pp. 300–305. American Statistical Association, Washington, D.C. (1978)
17. Kalton, G., Flores-Cervantes, I.: Weighting methods. J. Off. Stat. **19**(2), 81–98 (2003)
18. Kish, L.: Weighting for unequal pi. J. Off. Stat. **8**(2), 183 (1992)
19. Kott, P.S.: Using calibration weighting to adjust for nonresponse and coverage errors. Surv. Methodol. **32**(2), 133–142 (2006)
20. Le Guennec, J., Sautory, O.: La macro CALMAR2: redressement d'un échantillon par calage sur marges. Tech. rep., INSEE (2005)
21. Lumley, T.: Package 'survey'. R package version 3.30-3 (2015)
22. Lundström, S., Särndal, C.E.: Calibration as a standard method for treatment of nonresponse. J. Off. Stat. **15**, 305–327 (1999)
23. Mathar, R., Žilinskas, A.: A class of test functions for global optimization. J. Glob. Optim. **5**(2), 195–199 (1994)
24. Nieuwenbroek, N., Boonstra, H.: Bascula 4.0 Reference Manual. Statistics Netherlands, The Hague/Heerlen (2002)
25. Office for National Statistics. Social Survey Division: Labour Force Survey Five-Quarter Longitudinal Dataset, April 2012–June 2013. UK Data Service. SN: 7379. http://dx.doi.org/10.5255/UKDA-SN-7379-2 (2015)
26. Rao, J., Singh, A.: Range restricted weight calibration for survey data using ridge regression. Pak. J. Stat. **25**(4), 371–384 (2009)
27. Särndal, C.: The calibration approach in survey theory and practice. Surv. Methodol. **33**(2), 99–119 (2007)
28. Särndal, C.E., Swensson, B., Wretman, J.: Model Assisted Survey Sampling. Springer, New York (2003)

29. Sautory, O.: Redressements d'échantillons d'enquêtes auprès des ménages par calage sur marges. Tech. rep., INSEE (1991)
30. Sautory, O.: La macro CALMAR. Redressement d'un échantillon par calage sur marges. Tech. rep., INSEE (1993)
31. Sautory, O.: CALMAR2: a new version of the CALMAR calibration adjustment program. In: Proceedings of Statistics Canada Symposium (2003)
32. Singh, A., Mohl, C.: Understanding calibration estimators in survey sampling. Surv. Methodol. **22**(2), 107–116 (1996)
33. Tate, P.F.: Utilising longitudinally linked data from the British labour force survey. Surv. Methodol. **25**, 99–104 (1999)
34. Théberge, A.: Calibration and restricted weights. Surv. Methodol. **26**(1), 99–108 (2000)
35. Tillé, Y., Matei, A.: Package 'sampling'. R package version 2.6 (2015)
36. Torn, A., Žilinskas, A.: Global Optimization. Springer, New York (1989)
37. Vanderhoeft, C.: Generalised calibration at Statistics Belgium: SPSS module g-CALIB-S and current practices. Tech. rep., Statistics Belgium (2001)
38. Ypma, T.J.: Historical development of the Newton-Raphson method. SIAM Rev. **37**(4), 531–551 (1995)
39. Zardetto, D.: Package 'ReGenesees'. R package version 1.7 (2015)
40. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer, Berlin (2008)
41. Žilinskas, A.: A review of statistical models for global optimization. J. Glob. Optim. **2**(2), 145–153 (1992)
42. Žilinskas, A.: On similarities between two models of global optimization: statistical models and radial basis functions. J. Glob. Optim. **48**(1), 173–182 (2010)
43. Žilinskas, A.: A statistical model-based algorithm for 'black-box' multi-objective optimisation. Int. J. Syst. Sci. **45**(1), 82–93 (2014)
44. Žilinskas, A., Žilinskas, J.: Branch and bound algorithm for multidimensional scaling with city-block metric. J. Glob. Optim. **43**(2), 357–372 (2009)
45. Žilinskas, A., Žilinskas, J.: P-algorithm based on a simplicial statistical model of multimodal functions. Top **18**(2), 396–412 (2010)

Multidimensional Scaling for Genomic Data

Audrone Jakaitiene, Mara Sangiovanni, Mario R. Guarracino, and Panos M. Pardalos

Abstract Scientists working with genomic data face challenges to analyze and understand an ever-increasing amount of data. Multidimensional scaling (MDS) refers to the representation of high dimensional data in a low dimensional space that preserves the similarities between data points. Metric MDS algorithms aim to embed inter-point distances as close as the input dissimilarities. The computational complexity of most metric MDS methods is over $O(n^2)$, which restricts application to large genomic data ($n \gg 10^6$). The application of non-metric MDS might be considered, in which inter-point distances are embedded considering only the relative order of the input dissimilarities. A non-metric MDS method has lower complexity compared to a metric MDS, although it does not preserve the true relationships. However, if the input dissimilarities are unreliable, too difficult to measure or simply unavailable, a non-metric MDS is the appropriate algorithm. In this paper, we give overview of both metric and non-metric MDS methods and their application to genomic data analyses.

Keywords metric multidimensional scaling; non-metric multidimensional scaling; data mining; genomic data

A. Jakaitiene (✉)
System Analysis Department, Institute of Mathematics and Informatics,
Vilnius University, Akademijos str. 4, LT-08663 Vilnius, Lithuania
e-mail: audrone.jakaitiene@mii.vu.lt

M. Sangiovanni • M.R. Guarracino
High Performance Computing and Networking Institute, National Research Council,
Via Pietro Castellino 111, Naples 80131, Italy
e-mail: mara.sangiovanni@na.icar.cnr.it; mario.guarracino@cnr.it

P.M. Pardalos
Center for Applied Optimization, University of Florida, 01 Weil Hall,
P.O. Box 116595, Gainesville, FL 32611-6595, USA
e-mail: pardalos@ufl.edu

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_7

Introduction

Representing high dimensional data in a low dimensional space is an important task, since it is easier to study the information structure when the dimension is greatly reduced [8, 32]. Multidimensional scaling is a method that represents measurements of (dis)similarity among pairs of objects as distances in a space of lesser dimensionality [4]. With the dimensional reduction, it is possible to cluster the data relationships exploiting their distribution in the low dimensional space, and explore significant patterns. When the data configuration is Euclidean, multidimensional scaling (MDS) is similar to principal component analysis (PCA), which can remove inherent noise with its compact representation of data [32]. However, when the data configuration is nonlinear, MDS can be applied for a better understanding of the embedded data structure [30, 32].

Torgerson [31] proposed the first MDS method based on the Euclidean distance, and a complete symmetric similarity matrix (with no missing data). In 1964, Kruskal [13] formulated MDS as a problem of minimization of the STRESS function. Since then, various MDS approaches have been developed and applied to many fields, e.g., pattern recognition, stock market analysis, molecular conformational analysis, medicine, pharmacology, and environmental monitoring [8].

Nowadays, due to the larger computational power, multidimensional scaling is widely used as a tool for visualization of genomic data. Recent high-throughput next generation sequencing (NGS) technologies are spreading at very fast pace in the fields of genomics and functional genomics. These new methodologies produce large amounts of short reads, i.e. nucleotidic (DNA) or amminoacidic (protein) sequences. These extremely large datasets should be stored and properly processed to extract relevant knowledge, and require appropriate bioinformatics and computational biology strategies. MDS, as a tool for the dimensionality reduction, is of great interest not only for the visualization of large amounts of data, but also to reveal the data structure by clustering them based on their distribution in the low dimensional space. Indeed, MDS techniques may greatly help researchers in exploring significant patterns in multidimensional data.

Figure 1 shows the application of metric MDS to visualize complex 3D data, namely a Möbius strip, a surface with only one side and only one boundary, in a space of lower dimension (2D).

Multidimensional scaling algorithms fall into two broad classes: metric algorithms, which seek an embedding with inter-point distances closely matching the input dissimilarities; and non-metric algorithms, which find an embedding respecting only the relative ordering of the input dissimilarities [1]. In this paper, we give overview of both metric and non-metric MDS methods and their application to genomic data analyses.

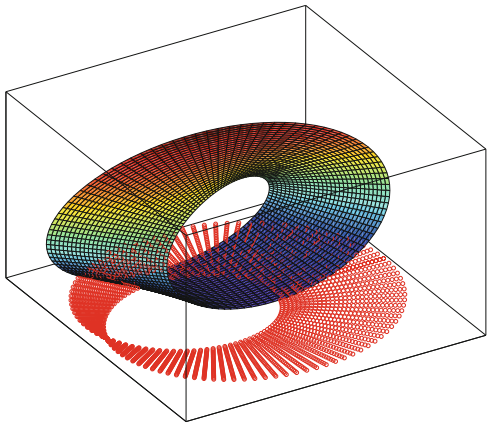

Fig. 1 2D visualization of 3D data. Here metric MDS is used on a Möbius strip with a Euclidean distance metric

Metric MDS

The objective of MDS is to find a configuration in a low dimensional space so that the distances among the points in this configuration, d_{ij}, are as close as possible in value to the distances δ_{ij}, $i,j = 1,\ldots,n$, in the original space. The input to MDS is a matrix of pairwise dissimilarities δ_{ij}, with $\delta_{ij} > 0$, symmetric and with zero diagonal. When the original data are set of points in a multidimensional space $\mathbb{R}^d$, the dissimilarities are defined as distances in that space.

For visualization purposes, the embedding space is usually chosen to be two-dimensional ($m = 2$); however, greater dimensionalities ($m < d$) might be interesting for some applications. The embedding points $x_i \in \mathbb{R}^m$ must preserve the inter-point distances of the input data. This problem might be solved in several ways [4, 6, 38]. One of the most used methodologies is based on an iterative approach in which, starting from an initial configuration, the positions in the destination space are changed until a suitable configuration is found. This goal might be reached by minimizing the STRESS objective function $f(X)$ [7] defined as follows

$$f(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(d_{ij}(X) - \delta_{ij}\right)^2, \tag{1}$$

where $X = (x_1, x_2, \ldots, x_n)$ and $d_{ij}(X)$ denotes appropriate distance between the points x_i and x_j, $i,j = 1,\ldots,n$, such as Euclidean, Manhattan (city block), Bray, or some other distance. Weights are assumed to be positive $w_{ij} > 0$.

Given that δ_{ij} and d_{ij} are zero diagonal symmetric and $w_{ij} = w_{ji}$, it is possible to sum up $i < j$ or $j < i$ elements, obtaining the formula of raw STRESS

$$f_r(X) = \sum_{i<j} w_{ij} \left(d_{ij}(X) - \delta_{ij}\right)^2 \tag{2}$$

and, respectively, the normalized STRESS

$$f_n(X) = \frac{\sum_{i<j} w_{ij} \left(d_{ij}(X) - \delta_{ij}\right)^2}{\sum_{i<j} w_{ij} \delta_{ij}^2}. \tag{3}$$

To obtain inter-point distances in lower dimensional spaces the STRESS function must be minimized. Mathematically the optimization problem to solve is the following

$$f^* = \min f(X), \tag{4}$$

where $f(X)$ is a nonlinear objective function $f : \mathbb{R}^n \to \mathbb{R}$ of continuous variables X and n is the number of variables. The goal is to find all X^* where the objective function is minimal, i.e.

$$X^* : f(X^*) = f^*. \tag{5}$$

The goodness-of-fit of the found solution(s) might be assessed by defining a relative error calculated as

$$E_M(X) = \sqrt{\frac{f(X)}{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \delta_{ij}^2}}. \tag{6}$$

Depending on the problem, it is also possible to consider an everywhere differentiable S-STRESS function

$$f_S(X) = \sum_{i<j} w_{ij} \left(d_{ij}^2(X) - \delta_{ij}^2\right)^2 \tag{7}$$

or, instead of the least-squares STRESS, the least absolute deviation (L_1-norm) STRESS function

$$f_{L1}(X) = \sum_{i<j} w_{ij} \left|d_{ij}(X) - \delta_{ij}\right|. \tag{8}$$

Although the STRESS function is defined by the analytic formula (2), which seems rather simple, it may have many local minima. The minimization problem is highly dimensional, with the number of variables equal to $n \times m$. Moreover, at some points the function $f(X)$ might be not differentiable. These features make the minimization of $f(X)$ a difficult task [35, 37]. The optimization problem without any assumption about unimodality is called global optimization problem. In Dzemyda

et al. [7] various approaches of global optimization methods applied to metric MDS are described.

The computational complexity of most metric MDS methods is over $O(n^2)$, similar to the complexity of the distance geometry problems, e.g., to the problems of molecular conformation [20]. Therefore the application of metric MDS to large genomic data ($n \gg 10^6$) might be computationally challenging, and the use of high performance parallel environment might be essential [8], as well as the application of non-metric MDS.

Non-metric MDS

Non-metric MDS has been extensively used in the psychometrics and psychophysics communities. In those sciences the magnitude of the input dissimilarities δ_{ij} could be unreliable, too difficult to measure, or simply unavailable [1]. Therefore the δ_{ij} can be interpreted only in an ordinal sense. In such cases, the dissimilarities δ_{ij} are replaced by disparities, $\hat{d}_{ij}$, and the STRESS function can be written as

$$f_d(X) = \sum_{i<j} w_{ij} \left(d_{ij}(X) - \hat{d}_{ij}\right)^2. \tag{9}$$

For the assessment of goodness-of-fit, it is possible to define the function

$$E_{NM}(X) = \sqrt{\frac{\sum_{i<j} w_{ij} \left(d_{ij}(X) - \hat{d}_{ij}\right)^2}{\sum_{i<j} d_{ij}(X)^2}}. \tag{10}$$

The aim is to find such a configuration that d_{ij} are in the same rank order as the original δ_{ij} in the ordinal or non-metric space. In metric MDS the disparities are related to the dissimilarities by a specific continuous function, whose calculation is carried out using least-squares monotonic regression.

Ordinal models typically require that if $\delta_{ij} < \delta_{kl}$, then $\hat{d}_{ij} < \hat{d}_{kl}$, with no particular order of the distances for $\delta_{ij} = \delta_{kl}$.

Nominal MDS models are obtained when the distances are presented as the qualitative distinctiveness, and the disparities are restricted by if $\delta_{ij} = \delta_{kl}$, then $\hat{d}_{ij} = \hat{d}_{kl}$. A word of caution should be given, since in nominal MDS the interpretation in which the closer the points are, the more similar objects they represent, does not hold anymore.

MDS for Genomics Data

NGS techniques are producing a tremendous amount of genomic data at an impressive pace. Storage, transmission, analyses, and visualization of these information is a daunting and challenging process [15, 33].

A typical NGS experiment produces billions of genomic sequences (reads) [17] that need Gigabytes of memory to be stored and processed. New computational infrastructure and methodologies are needed to extract meaningful information from the overwhelming amount of big data produced [19].

MDS might represent an invaluable tool to address this need, since it permits to detect complex data structures and visualize them in low dimensional (2D/3D) spaces [38].

The input to both metric and non-metric MDS is a pairwise dissimilarity matrix of size $[n \times n]$, with n number of input samples. The computational complexity of MDS is over $O(n^2)$, although it might be reduced to $O(n\sqrt{n})$ using hybrid approaches [18]. Non-metric MDS might be faster than metric MDS, but since it preserves only the order of similarities instead of their original scale it is somehow considered as not fully reliable [32].

Many novel algorithms and approaches are addressing the problem of reducing the computational complexity of MDS, to allow its application also to large genomic datasets. Hughes et al. [12] propose an interpolative approach to reduce the size of the input matrix. They use metric MDS to perform clustering of DNA sequences based on a pairwise genetic distance. They were able to rapidly process with the proposed algorithm a dataset of 10^5 sequences. Tzeng et al. [32] propose a novel method based on dividing the initial problem into smaller, easier problems. Then the global solution is built combining the results obtained for the sub-problems. They were able to build a correlation map of about 16×10^3 genes using more than 2000 annotations coming from the Gene Ontology database. Stanberry et al. [29] developed a methodology based on interpolation, to reduce the size of input data, and on parallelization, to reduce the computation time. They were able to process and visualize about 10^4 protein sequences, with the aim to support their functional annotation. Taguchi and Oono [30] describe a novel, extremely non-metric algorithm to perform unsupervised data mining on gene expression time series. They used Pearson correlations as a dissimilarity measure to process about 500 genes in 11 time points. Park et al. [21] describe CFMDS-CUDA, a parallel computing architecture for metric MDS that exploits the power of GPUs (graphical processing unit) and a divide-and-conquer approach. They tested it on mouse microarray data of size $10^4 \times 1000$.

Other improvement directions take into account the quality of the obtained solution depending on the initial condition as in Becavin et al. [3], or the use of genetic algorithms to avoid local minima as in Žilinskas and Žilinskas [36].

Table 1 lists some papers that use MDS for genomic analyses. The table shows the data size in terms of the number of inputs, n, and features, m. Thus, the dissimilarity matrix given as input to the MDS algorithm is of size $[n \times n]$. Genomic

Table 1 List of papers using MDS on genomic datasets

Paper reference	Data type and size [n × m]	Type	IDF OF	Purpose
Heinrich [11]	Genome vs SNPs $[1063 \times 35 \times 10^6]$	NM	Genotype-weighted stress	Exome genotyping accuracy assessment
Hughes [12]	DNA vs DNA alignments [100,000 × 1]	M	Genetic distance stress	Sequence clustering
Malaspinas [14]	Genomes vs SNPs [950 × 600,000]	M	Allele-sharing stress	Ancestry calling for low-depth data
McCue [16]	Genomes vs SNPs [335 × 54,000]	M	Genetic distance stress	Inbreeding evaluation
Ruan [25]	DNA vs DNA alignments [1306 × 1]	M	Percent identity stress	Clustering and 3D phylograms building
Park [22]	Cells vs gene expression [150 × 48]	NM	Spearman correlation stress	Single cell phenotype determination
Schloss [27]	OTUs vs conditions [~40 × 12]	NM	ThetaYC stress	Sequencing quality assessment
Staley [28]	OTUs vs sites [280× 10]	NM	Bray–Curtis stress	Impact of land use on bacterial communities
Stanberry [29]	Protein vs protein sequences [10,000 × 720]	M	Protein similarity Summon's loss	Visualization and annotation of proteins
Taguchi [30]	Gene expressions vs time [517 × 11]	NM	Pearson correlations Kendall statistics	Unsupervised data mining
Tzeng[32]	Gene names vs GO annotation [16,502 × 2168]	M	Euclidean stress	Individuation of gene correlation maps
Zhu [34]	DNA sequences vs SNPs [216 × 2000]	NM	Taxonomic distance stress	Improvement of association mapping results

Data type specifies the genomic data used as input (here OTU stands for operational taxonomic unit). Data size is also shown, as $[n \times m]$. The MDS type is either metric (M) or non-metric (NM). The input dissimilarity function (IDF) and the optimized function (OF) used in each paper are also shown. Lastly, a brief description of the purpose of the performed analyses is given

data range from a small number of input points and features, as in Schloss et al. [27] where MDS is used alongside other tools to visualize sequenced data and assess their quality, to very large datasets as in [12, 29, 32].

Several of this works use MDS to process DNA sequences. It is interesting to note that both metric and non-metric MDS are used, with a wide range of different dissimilarity measures. When performing genotyping, in which the DNA sequences of one or more organisms are compared against lists of known mutations (called SNPs, single nucleotide polymorphisms), Malaspinas et al. [14] and McCue et al. [16] rely on metric MDS where dissimilarity between sequences is calculated using two different, genetic-based distances. Instead, Heinrich et al. [11] and Zhu [34] use non-metric MDS with a genetic-based and, respectively, a taxonomy-based dissimilarity. In all the cases a STRESS objective function is minimized. DNA vs DNA sequence comparison is performed by both Hughes et al. [12] and Ruan et al. [25] using metric MDS, with the aim to cluster together similar sequences. In the latter a distance based on the percentage of identical nucleotides across sequences is used to build 3D philograms of the input data, whereas the former relies on a different genetic-distance based metric.

Gene expression data processing is addressed with non-metric MDS both in Park et al. [22] and in Taguchi et al. [30]. The first group use Spearman correlation as a dissimilarity measure and minimize a classic STRESS function, to assess the phenotype of single cells. Instead, Taguchi et al. rely on Pearson correlation to perform unsupervised data mining. Interestingly, they minimize an objective function based on Kendall statistics.

Another field in which MDS is widely used is the analysis of microbiomes, in which operational taxonomic units (OTUs; a microbial diversity unit of measure) are compared across different conditions or places. Non-metric MDS is used in the work of both Schloss et al. [27] and Staley et al. [28], where ThetaYC and Bray–Curtis dissimilarity measures are used, respectively. See also Gonzalez et al. [10] for an interesting review on the relationships between microbiota analysis results and different MDS approaches.

Stanberry et al. [29] propose a tool to visualize and annotate clusters of proteins. Their work is based on metric MDS in which the dissimilarity measure is given by protein sequence distance, and a Summon's loss function is minimized. Lastly, Tzeng et al. [32] use MDS to build correlation maps based on gene names and Gene Ontology (GO) annotations. They use Euclidean distance and classic STRESS function.

Software

Multidimensional scaling is becoming a tool of standard use in genomic analyses. Many software suites and packages, expressly tailored on the needs of bioinformaticians and computational biologists, offer MDS-based functions as a standard tool not only for data visualization but also to perform more sophisticated analyses such as unsupervised mining. In some cases MDS analyses are a part of a larger pipeline,

as in MOTHUR [26], an open-source software designed to support the study of microbial communities. The analyses described in Staley et al. [28] and in Schloss et al. [27] are both performed using the MOTHUR software. Similarly, in Malaspinas et al. [14] metric MDS is a part of *bammds*, a more complex tool developed to assess the ancestry of low-depth whole-genome data. PLINK [23] is a tool, largely used for whole-genome association and population-based linkage analyses, that offers the possibility to perform also MDS analyses. DACIDR [24] is a tool designed to cluster microbiota sequences into OTUs. METAGENassist [2] includes MDS as an analysis and visualization tool for metagenomics studies. METAREP [9] is a similar tool, also designed to compare and annotate metagenomic datasets. PRIMER6 [5] is a commercial software package used for analyzing species or sample abundance. Lastly, the biologists' reference language R provides two distinct packages, namely *stats* and MASS, to perform both metric and non-metric MDS analyses.

References

1. Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D.J., Belongie, S.: Generalized non-metric multidimensional scaling. In: International Conference on Artificial Intelligence and Statistics, pp. 11–18 (2007)
2. Arndt, D., Xia, J., Liu, Y., Zhou, Y., Guo, A.C., Cruz, J.A., Sinelnikov, I., Budwill, K., Nesbø, C.L., Wishart, D.S.: Metagenassist: a comprehensive web server for comparative metagenomics. Nucleic Acids Res. **40**, W88–W95 (2012)
3. Bécavin, C., Tchitchek, N., Mintsa-Eya, C., Lesne, A., Benecke, A.: Improving the efficiency of multidimensional scaling in the analysis of high-dimensional data using singular value decomposition. Bioinformatics **27**(10), 1413–1421 (2011)
4. Borg, I., Groenen, P.J.: Modern Multidimensional Scaling: Theory and Applications. Springer Series in Statistics, vol. 1. Springer, New York (2005)
5. Clarke, K., Warwick, R.: Change in Marine Communities: An Approach to Statistical Analysis and Interpretation. Primer-E Ltd., Devon (2001)
6. Cox, T.F., Cox, M.A.: Multidimensional Scaling. Chapman and HallCRC Monographs on Statistics and Applied Probability, vol. 88. Chapman and Hall/CRC Press, London/Boca Raton (2000)
7. Dzemyda, G., Kurasova, O., Žilinskas, J.: Multidimensional Data Visualization. Methods and Applications Series: Springer Optimization and its Applications, vol. 75, pp. 122. Springer, Berlin (2013)
8. Floudas, C.A., Pardalos, P.M.: Encyclopedia of Optimization, vol. 1. Springer Science and Business Media, Berlin (2008)
9. Goll, J., Rusch, D.B., Tanenbaum, D.M., Thiagarajan, M., Li, K., Methé, B.A., Yooseph, S.: METAREP: JCVI metagenomics reports—an open source tool for high-performance comparative metagenomics. Bioinformatics **26**(20), 2631–2632 (2010)
10. Gonzalez, A., Knight, R.: Advancing analytical algorithms and pipelines for billions of microbial sequences. Curr. Opin. Biotechnol. **23**(1), 64–71 (2012)
11. Heinrich, V., Kamphans, T., Stange, J., Parkhomchuk, D., Hecht, J., Dickhaus, T., Robinson, P.N., Krawitz, P.M.: Estimating exome genotyping accuracy by comparing to data from large scale sequencing projects. Genome Med. **5**, 1–11 (2013)
12. Hughes, A., Ruan, Y., Ekanayake, S., Bae, S.H., Dong, Q., Rho, M., Qiu, J., Fox, G.: Interpolative multidimensional scaling techniques for the identification of clusters in very large sequence sets. In: Proceedings from the Great Lakes Bioinformatics Conference 2011, vol. 13, p. S9. BioMed Central Ltd, London (2012)

13. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
14. Malaspinas, A.S., Tange, O., Moreno-Mayar, J.V., Rasmussen, M., DeGiorgio, M., Wang, Y., Valdiosera, C.E., Politis, G., Willerslev, E., Nielsen, R.: Bammds: a tool for assessing the ancestry of low-depth whole-genome data using multidimensional scaling (MDS). Bioinformatics **30**(20), 2962–2964 (2014)
15. Marx, V.: Biology: the big challenges of big data. Nature **498**(7453), 255–260 (2013)
16. McCue, M.E., Bannasch, D.L., Petersen, J.L., Gurr, J., Bailey, E., Binns, M.M., Distl, O., Guérin, G., Hasegawa, T., Hill, E.W., et al.: A high density SNP array for the domestic horse and extant perissodactyla: utility for association mapping, genetic diversity, and phylogeny studies. PLoS Genet. **8**(1), e1002,451 (2012)
17. Metzker, M.L.: Sequencing technologies—the next generation. Nat. Rev. Genet. **11**(1), 31–46 (2010)
18. Morrison, A., Ross, G., Chalmers, M.: Fast multidimensional scaling through sampling, springs and interpolation. Inf. Vis. **2**(1), 68–77 (2003)
19. Nekrutenko, A., Taylor, J.: Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. Nat. Rev. Genet. **13**(9), 667–672 (2012)
20. Pardalos, P.M., Shalloway, D., Xue, G., et al.: Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 23. American Mathematical Society, Providence, RI (1996)
21. Park, S., Shin, S.Y., Hwang, K.B.: CFMDS: CUDA-based fast multidimensional scaling for genome-scale data. BMC Bioinf. **13**(Suppl 17), 1–23 (2012)
22. Park, J., Brureau, A., Kernan, K., Starks, A., Gulati, S., Ogunnaike, B., Schwaber, J., Vadigepalli, R.: Inputs drive cell phenotype variability. Genome Res. **24**(6), 930–941 (2014)
23. Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M.A., Bender, D., Maller, J., Sklar, P., De Bakker, P.I., Daly, M.J., et al.: Plink: a tool set for whole-genome association and population-based linkage analyses. Am. J. Hum. Genet. **81**(3), 559–575 (2007)
24. Ruan, Y., Ekanayake, S., Rho, M., Tang, H., Bae, S.H., Qiu, J., Fox, G.: DACIDR: deterministic annealed clustering with interpolative dimension reduction using a large collection of 16s rRNA sequences. In: Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB '12, pp. 329–336. ACM, New York (2012)
25. Ruan, Y., House, G.L., Ekanayake, S., Schutte, U., Bever, J.D., Tang, H., Fox, G.: Integration of clustering and multidimensional scaling to determine phylogenetic trees as spherical phylograms visualized in 3 dimensions. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 720–729. IEEE, New York (2014)
26. Schloss, P.D., Westcott, S.L., Ryabin, T., Hall, J.R., Hartmann, M., Hollister, E.B., Lesniewski, R.A., Oakley, B.B., Parks, D.H., Robinson, C.J., et al.: Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl. Environ. Microbiol. **75**(23), 7537–7541 (2009)
27. Schloss, P.D., Gevers, D., Westcott, S.L.: Reducing the effects of pcr amplification and sequencing artifacts on 16s rRNA-based studies. PloS One **6**(12), e27,310 (2011)
28. Staley, C., Unno, T., Gould, T.J., Jarvis, B., Phillips, J., Cotner, J.B., Sadowsky, M.J.: Application of Illumina next-generation sequencing to characterize the bacterial community of the Upper Mississippi River. J. Appl. Microbiol. **115**(5), 1147–1158 (2013)
29. Stanberry, L., Higdon, R., Haynes, W., Kolker, N., Broomall, W., Ekanayake, S., Hughes, A., Ruan, Y., Qiu, J., Kolker, E., et al.: Visualizing the protein sequence universe. Concurr. Comput. Pract. Exper. **26**(6), 1313–1325 (2014)
30. Taguchi, Y.h., Oono, Y.: Relational patterns of gene expression via non-metric multidimensional scaling analysis. Bioinformatics **21**(6), 730–740 (2005)
31. Torgerson, W.S.: Multidimensional scaling: I. Theory and method. Psychometrika **17**(4), 401–419 (1952)
32. Tzeng, J., Lu, H.H., Li, W.H.: Multidimensional scaling for large genomic data sets. BMC Bioinf. **9**(1), 179 (2008)

33. Wolfe, P.J.: Making sense of big data. Proc. Natl. Acad. Sci. **110**(45), 18031–18032 (2013)
34. Zhu, C., Yu, J.: Nonmetric multidimensional scaling corrects for population structure in association mapping with different sample types. Genetics **182**(3), 875–888 (2009)
35. Žilinskas, A., Jakaitiene, A.: A conjugate gradient method for two dimensional scaling. Commun. Cognition. Monograph. **43**(3–4), 3–13 (2010)
36. Žilinskas, A., Žilinskas, J.: Parallel genetic algorithm: assessment of performance in multidimensional scaling. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 1492–1501. ACM, New York (2007)
37. Žilinskas, A., Žilinskas, J.: Two level minimization in multidimensional scaling. J. Glob. Optim. **38**(4), 581–596 (2007)
38. Žilinskas, A., Žilinskas, J.: Optimization-based visualization. In: Encyclopedia of Optimization, pp. 2785–2791. Springer, Berlin (2008)

Solving Stochastic Ship Fleet Routing Problems with Inventory Management Using Branch and Price

Ken McKinnon and Yu Yu

Abstract This chapter describes a stochastic ship routing problem with inventory management. The problem involves finding a set of least cost routes for a fleet of ships transporting a single commodity when the demand for the commodity is uncertain. Storage at supply and consumption ports is limited and inventory levels are monitored in the model. Consumer demands are at a constant rate within each time period, and in the stochastic problem, the demand rate for a period is not known until the beginning of that period. The demand situation over the time periods is described by a scenario tree with corresponding probabilities. A decomposition formulation is given and it is solved using a Branch and Price framework. A master problem (set partitioning with extra inventory constraints) is built, and the subproblems, one for each ship, are solved by stochastic dynamic programming and yield the columns for the master problem. Each column corresponds to one possible tree of actions for one ship giving its schedule loading/unloading quantities for all demand scenarios. Computational results are given showing that medium sized problems can be solved successfully.

Keywords Stochastic Dynamic Programming • Branch and Price • Ship Routing • Inventory Management.

Introduction

The marine shipping industry has experienced an unprecedented boom over the past decade. This is not only because of the rapid growth of the requirements to transfer more and more energy and commercial commodities from one location to another, but also because the characteristics of the ocean shipping industry, with its low transportation costs and huge loading capacity, are suitable for cheaply transporting huge amounts of products.

K. McKinnon (✉) • Y. Yu
School of Mathematics, University of Edinburgh, Edinburgh, UK
e-mail: k.mckinnon@ed.ac.uk; yu.yu@pi-solution.com

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_8

The classical routing and scheduling problem for vehicles and ships is important part of the general transportation problem, and has received a great deal of attention in academic research. A large number of possible solution approaches have been presented in the literature, involving either exact optimization methods or heuristic algorithms. A comprehensive review is provided in [10]. This focuses on literature about ship routing and scheduling published between the years 1990 and 2003. The survey is presented in several different parts: strategy planning problem, tactical and operational planning problems, naval problems, and other related problems. A survey of different solution methods in the literature is also presented in the review. A mixed integer programming (MIP) model is described in [25] for the problem of transporting different bulk products from a set of origins to a set of destinations by a fleet of ships. A ship has separate compartments for different products. A ship's voyage goes from a single loading port to a single discharging port. A cost-based heuristic algorithm is also presented to obtain acceptable solution quickly. Sherali et al. [26] have presented an MIP model for the Kuwait Petroleum Corporation (KPC) problem. Because of the integrality conditions and the large number of demand contract scenarios, the problem cannot be solved to optimality by the MIP model. An alternative aggregated model is then formulated and solved by a specialized rolling horizon heuristic method to make the problem solvable. In the ocean shipping industry, expert opinion is an important factor. Crary et al. [11] introduce a model integrating the expert opinion and MIP model for the problem of sizing the US destroyer fleet. MIP models for SRP are also built in [3, 24, 27]. Heuristics are developed in [19] in order to obtain an acceptable solution within reasonable time when solving the MIP model.

The Dantzig–Wolfe decomposition approach has proved to be successful for the vehicle routing problem with time windows. Desrochers et al. [14] were the first to propose a set partitioning model for the vehicle routing problem with time windows solved by column generation, and this appears to be an efficient way of finding the optimal solution. As for the ship routing problem, it is also a good solution approach. There is much literature on solving the problem by Dantzig–Wolfe decomposition. Early papers [1, 2] describe a typical tramp ship scheduling problem, which were the first works to use a Dantzig–Wolfe decomposition approach for ship routing and scheduling. The master problem is the linear relaxation of a set partitioning problem and subproblems are shortest path problems. But the algorithm presented cannot guarantee optimal integer solutions. In [5, 8, 9], the demand is regarded as a constant and Branch and Price is used to solve the problem. The problem is decomposed into a ship route subproblem for each ship and a port inventory subproblem for each port. The approach presented in this chapter is closest to that used in their paper. Compared to their papers, the present chapter deals with different inventory situations, and solves the stochastic problem rather than deterministic problem.

In realistic shipping operations, especially for ocean shipping, much of the planning data is uncertain. Deterministic models for ship routing and scheduling are sometimes inappropriate, and there is a need to develop stochastic model.

The stochastic vehicle routing problem (SVRP) without inventory constraints and with simple recourse actions is discussed extensively in the literature. A Branch and Price algorithm for vehicle routing problem with stochastic demands is presented in [7]. In this paper, the expected number of failures and the corresponding penalty cost are considered in the objective function and a two stage stochastic program with fixed recourse and capacity constraints is built. A straight-forward modification of the Clark and Wright savings algorithm for the SVRP based on a discussion of route failure is presented in [16]. In [18, 20], an integer L-shaped method is used to solve SVRP to optimality. In [4] an a priori sequence among all customers of minimal expected total length is proposed, and a variety of theoretical approaches are analyzed as well. In addition, several solution frameworks for the stochastic vehicle routing with stochastic demands are discussed in [17].

There are few references in the literature to stochastic ship routing problems with inventory. A robust ship scheduling with multiple time windows is presented in [6]. This uses a set partitioning approach with the columns found a priori to minimize the chances that ships stay idle in ports during the non-working days. A Markov decision process model of the stochastic inventory routing problem is introduced in [23], and approximation methods are used to find acceptable solutions.

This chapter considers the problem of optimizing the distribution of a single commodity by a fleet of ships when there is limited storage at the supply and consumption ports and the consumer demand is uncertain. Consumer demand is described by a scenario tree and demand is assumed to be constant within each period. A solution consists of a tree of schedules for each ship, where a schedule for a ship specifies the loading and unloading quantities at each port visited and the start time of each such operation (which we refer to as a *service*). These ship schedules must be such that the storage limits at ports are satisfied at all times. The problem is formulated as a multistage stochastic programming problem and is solved by Branch and Price—a Branch and Bound method that uses Dantzig–Wolfe decomposition to solve each node. The master problem is a set partitioning problem with extra inventory constraints. Each column in the master problem corresponds to a tree of schedules for a ship. Attractive columns are generated by a stochastic dynamic programming using a backward labelling method.

The structure of the rest of the chapter is as follows. Section “Decomposition Approach for the Stochastic Ship Routing Problem” introduces the Dantzig–Wolfe decomposition approach and describes the structure of the master and subproblems. This section also describes the techniques used to eliminate cycles. Sections “Branch and Bound” gives the Branch and Bound algorithm and “Examples and Results” present computational results, and section “Conclusion” gives the conclusion.

Decomposition Approach for the Stochastic Ship Routing Problem

Assumptions

The ocean transportation problem is too complex to consider every factor in the real world when modelling the problem. In this chapter we make the following simplifying assumptions:

- At each consumer port, the rate of demand is constant within a period, but can change between periods.
- At each port, loading and unloading rates are constant.
- At most one ship can be loading or unloading at any given time. This assumption avoids the overlap of services at a port.
- For each ship the travel time and cost between any two ports are fixed.
- A service at a port must start and finish within a period. Note, however, this is not a practical limitation as the service can continue without a break in the following period.

Solution Framework

A Branch and Price algorithm is used in this chapter. This consists of a master problem which is solved by Branch and Bound (B&B), with each node in the B&B tree being solved by Dantzig–Wolfe (DW) decomposition. Each column in the master problem corresponds to a tree of schedules for a ship. There is a huge number of these columns and if all were included explicitly the master problem would be impossible to solve. However the DW approach only generates the small subset of them that are needed, and is thus able to solve the full problem at each B&B node. In each iteration of DW a subproblem is solved for each ship to generate an attractive tree of schedules for that ship. In this chapter the subproblems are solved by stochastic dynamic programming.

At any stage in the solution of a master problem at a B&B node, a (finite) subset of the columns will have been generated. This problem, called a restricted master problem, is solved and the shadow prices of the constraints are then used to find the most negative reduced cost from among the columns that have not yet been generated. This can be done without explicitly generating any columns by solving a stochastic dynamic programming problem separately for each ship. The solution gives the tree of schedules for the ship. If this added as a column to the master problem, it would have the most negative reduced cost among all the possible columns for that ship. This procedure continues until no column with negative reduced cost can be generated, at which stage the master problem for that B&B node has been solved.

Master Problem

The detail formulation of a master problem is introduced here. A port can be visited several times within the time window of a scenario tree node, so an index for visit number is needed. In the model, many objects are indexed by the triple (Port, Visit, Scenario node) which is referred as a port visit. For any ship, there are a set of trees of schedules for it. The problem is to choose one tree of schedules for each ship. We introduce the details of master problem as below.

Indices

i	port
k	scenario tree node
$a(k)$	predecessor node of node k in scenario tree
m	the order of the visit to a port within a scenario tree node (i.e., the current visit is the mth time the port has been visited in this node)
v	ship
s	tree of schedule for one ship
(i,m,k)	a port visit

Sets

N	set of ports
V	set of ships
K	set of scenario tree nodes
K^T	set of scenario tree nodes in final period
P	set of port visits
R_v	set of tree of schedules for ship v

Parameters

A_{svimk}	the number of times in tree of schedules s for ship v that it makes port visit (i,m,k) (must be 0 or 1 to be feasible)
C_{sv}	expected cost if ship v takes the tree of schedules s
Q_{svimk}	quantity unloaded by ship v in port visit (i,m,k) if ship makes that port visit in schedule tree s, and 0 otherwise (value is negative if ship is loading)
T_{svimk}	the start service time for ship v in (i,m,k) if the ship makes that port visit in schedule tree s, and 0 otherwise
B_k	end of the time period which includes scenario tree node k
W_i	unloading rate from ship to port i (value is negative if ship is loading)
M	the maximum number of visits to any port in a scenario tree node
D_{ik}	demand rate in port i in node k (value is negative at a supply port)
S_i	initial stock level in port i
$\bar{S}_i$	upper bound for storage in port i
$\underline{S}_i$	lower bound for storage in port i

The values of parameters A_{svimk}, Q_{svimk}, and T_{svimk} are found by solving subproblems. These three parameters represent the route information and all three

are either zero or all three are nonzero. A_{svimk} is the number of times ship v makes port visit (i,m,k) in schedule s, and to be feasible this must either be 0 or 1. However during the solution process there may be cycles of port visits, and this leads to values of A_{svimk} greater than 1. Such infeasible schedules cannot be included in the final optimal solution. Parameters Q_{svimk} and T_{svimk} represent, respectively, the quantity loaded and the start service time for this port visit.

Variables

x_{sv} 1 if ship v takes schedule tree s, and 0 otherwise
y_{imk} 1 if some ship makes port visit (i,m,k), and 0 otherwise
q_{imk} amount of commodity unloaded from a ship during port visit (i,m,k) (value is negative if ship is loading)
t^S_{imk} the start of service time in port visit (i,m,k)
t^E_{imk} the end of service time in port visit (i,m,k)
h^S_{imk} the stock level at time t^S_{imk}
h^E_{imk} the stock level at time t^E_{imk}

Formulation of Master Problem

$$\min \sum_{v\in V}\sum_{s\in R_v} C_{sv}x_{sv} \tag{1}$$

$$\sum_{v\in V}\sum_{s\in R_v} A_{svimk}x_{sv} = y_{imk} \ \ \forall (i,m,k)\in P \tag{2}$$

$$\sum_{v\in V}\sum_{s\in R_v} Q_{svimk}x_{sv} = q_{imk} \ \ \forall (i,m,k)\in P \tag{3}$$

$$\sum_{v\in V}\sum_{s\in R_v} T_{svimk}x_{sv} + (1-y_{imk})B_k = t^S_{imk} \ \ \forall (i,m,k)\in P \tag{4}$$

$$\sum_{s\in R_v} x_{sv} = 1 \ \ \forall v\in V \tag{5}$$

$$x_{sv}\geq 0 \ \ \forall v\in V, s\in R_v \tag{6}$$

$$\{x_{sv} : s\in R_v\} \text{ yield a valid tree of schedules for ship } v, \ \forall v \tag{7}$$

$$y_{imk}\in\{0,1\} \ \ \forall\ (i,m,k)\in P \tag{8}$$

$$t^E_{imk} = t^S_{imk} + q_{imk}/W_i \ \ \forall (i,m,k)\in P \tag{9}$$

$$t^E_{i,m-1,k} \leq t^S_{imk} \ \ \forall (i,m,k)\in P,\ m>1 \tag{10}$$

$$y_{imk}\geq y_{i,m+1,k} \ \ \forall (i,m,k)\in P \tag{11}$$

$$h^E_{imk} = h^S_{imk} - (t^E_{imk} - t^S_{imk})D_{ik} + q_{imk} \ \ \forall (i,m,k)\in P \tag{12}$$

$$h^E_{iMk} - (B_k - t^E_{iMk})D_{ik}\geq 0 \ \ \forall i\in N,\ k\in K^T \tag{13}$$

$$h^S_{imk} = S_i - t^S_{imk}D_{ik} \ \ \forall i\in N,\ m=1,\ k=1 \tag{14}$$

$$h^S_{imk} = h^E_{i,m-1,k} - (t^S_{imk} - t^E_{i,m-1,k})D_{ik} \ \ \forall (i,m,k)\in P,\ m>1 \tag{15}$$

$$h^S_{imk} = h^E_{i,M,a(k)} - (B_{a(k)} - t^E_{i,M,a(k)})D_{i,a(k)} - (t^S_{imk} - B_{a(k)})D_{ik} \quad \forall i \in N, m = 1, k > 1 \tag{16}$$

$$\underline{S}_i \le h^S_{imk},\ h^E_{imk} \le \bar{S}_i \quad \forall (i,m,k) \in P \tag{17}$$

In (1) we minimize the total expected cost. Constraints (5) and (6) result in a convex combination of schedule trees for each ship v, and to be valid must yield a single schedule tree. This can only happen if all schedule trees for ship v corresponding to $x_{sv} > 0$ follow the same tree of routes and the cost functions are linear over the convex hull (as the cost functions are in our examples). Constraint (2) calculates the number of occurrences of a port visit and ensures that each port visit occurs at most once. The variable y_{imk} is 0 if there are fewer than m ship visits at port i in scenario node k and is 1 otherwise. Constraint (3) calculates the loading or unloading quantity and constraint (4) calculates the start of service time for each port visit. If port visit (i,m,k) occurs, then the first term in (4) gives the start time for that service and the second term is zero. If port visit (i,m,k) does not occur, then the first term will be zero and the second term will be B_k, i.e., the end of the period for node k. Constraint (9) calculates the end of service time and (10) guarantees that there is no overlap between two services, i.e., a later port visit can only be served after the service of previous visit has been finished. Constraint (11) ensures that if a port is visited $m+1$ times in a scenario node, it must be visited m times in that scenario node. Constraints (12)–(17) are the inventory constraints. They ensure that the storage level is between the upper and lower bound of the storage tank at the start and end of each service. Since all flow rates are constant within a scenario node, the inventory level will change linearly between the start and end service times. So the constraints ensure that the inventory is within the bounds all the time within the whole planning period.

Reduced Cost

After a restricted master problem is solved (i.e., a master problem with a subset of the possible columns), dual variables will be known. These dual variables are denoted by d^A_{imk}, d^Q_{imk}, d^T_{imk}, and d^S_v for constraints (2)–(5), respectively. The reduced cost $\hat{C}_{sv}$ can then be calculated as follows:

$$\begin{aligned}\hat{C}_{sv} &= C_{sv} - \sum_{i,m,k} (A_{svimk} d^A_{imk} + Q_{svimk} d^Q_{imk} + T_{svimk} d^T_{imk}) - d^S_v \\ &= \sum_{(i,m,k) \to (i',m',k') \in E_s} P_k C_{ii'v} - \sum_{(i,m,k) \in N_s} (d^A_{imk} - d^Q_{imk} Q_{svimk} + d^T_{imk} T_{svimk}) - d^S_v \end{aligned} \tag{18}$$

where P_k is the cumulative probability from start to node k that defines the scenario tree, E_s the set of edges, and N_s the set of port visits defining the tree of schedules

s, and $C_{ii'v}$ is the travelling cost along the edge $i \to i'$ for ship v. Constraint (18) expresses the reduced cost as the sum of terms over the edges and nodes in the tree of schedules.

Ship Routing Subproblems

The parameters Q_{svimk} and T_{svimk} as well as set E_s and N_s in (18) represent the route information generated by subproblems and are not given in advance. We wish to generate a column with the minimum reduced cost so we replace these parameters with variables q_{vimk} and t^S_{vimk} and also a variable route, which is specified by variable sets of edges E and nodes N that will define the schedule tree. For a ship v, the objective of the subproblem can be formulated as follows:

$$\bar{C}_v = \min_{E,N} \min_{q} \min_{t^S} \left(\sum_{(i,m,k)\to(i',m',k')\in E} P_k C_{ii'v} - \sum_{(i,m,k)\in N} (d^A_{imk} + d^Q_{imk} q_{vimk} + d^T_{imk} t^S_{vimk}) \right) - d^S_v \quad (19)$$

In formulation (19), we try to find a physical visiting sequence and the corresponding values of q_{svimk} and t^S_{svimk} for each port visit in the sequence so as to minimize the reduced cost given in (18). The d^S_v term in (18) does not need to be considered in the subproblems. It can be subtracted from the objectives after solving the subproblems.

A ship subproblem can then be formulated as a shortest tree problem and solved by stochastic dynamic programming. The solution of the shortest tree problems is a tree of schedules with the least reduced cost, and yields a column that can be added into the master problem as a column. The state in the DP is (i,m,k,g,t), where i is the port, m is the order of the visit, k is the node of scenario tree, g is the amount of commodity on board the ship v when the ship arrives at port visit (i,m,k), and t is the start service time for the port visit (i,m,k). Both start service time, t, and the quantity on board the ship, g, are continuous quantities. However within the DP step they have to be restricted to discrete values, and this may lead to slight suboptimality. If a service time is between two grid points, it will be delayed to the next grid point, and a regular grid is used for values of g so that there is no inaccuracy in accounting for the amounts on board ships. However, using discrete values for g and t does not mean that our model can only generate the solution with these discrete values. In fact, the master problem may choose several columns with the same physical tree of routes but different time and loading quantities and use the average of these columns as the solution, which may have the start service times and loading quantities different from discrete values.

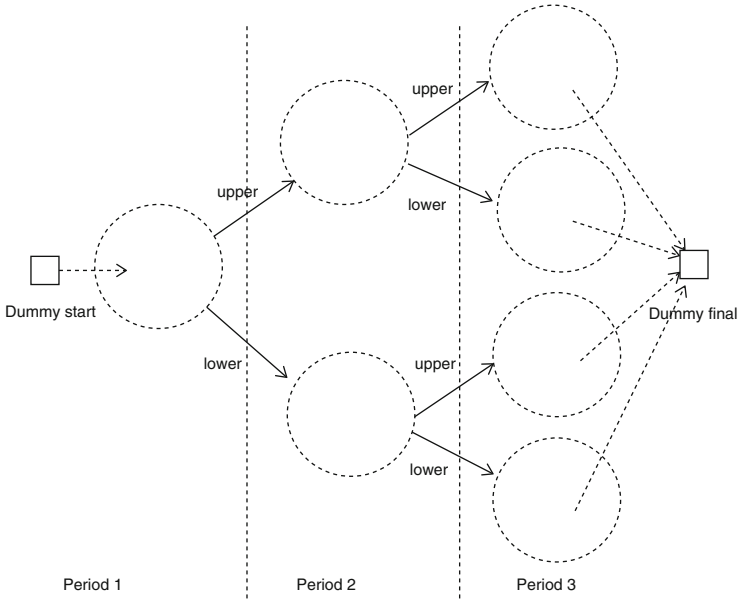

Fig. 1 Structure of a DP network showing demand scenario parts as *circles* and alternative upper and lower demand level branches in each period

Dynamic Programming Network

In this section, we describe the DP network for the ship subproblems. For each port visit in the network, there is a start service node and an end service node related to it. The costs in the objective are assigned to edges in the network. The DP network for a ship subproblem is related to the scenario tree which describes the pattern of consumer demands. We divide the network into several parts, each part, called a *demand scenario part,* represents a node in the demand scenario tree in the corresponding time period so that the DP network has the same top level structure as the demand scenario tree. See Figs. 1, 2, and 3 for examples.

In a DP network, a ship starts from the dummy start node, makes a set of port visits in different demand scenario parts of the network, and finishes the trip when it arrives at the dummy final node. When the ship is at a start service node, it makes decisions about how much to load or unload at the current port visit. When the ship is at an end service node it has a choice of three different actions: it can sail to another port visit in the same demand scenario part, it can stay at the current port visit until the future information is available before deciding which port to visit in the next period, or it can leave the current port visit immediately and sail to a port in the next period, in which case the future information about demand will be revealed during sailing but the ship will not change its destination port in response.

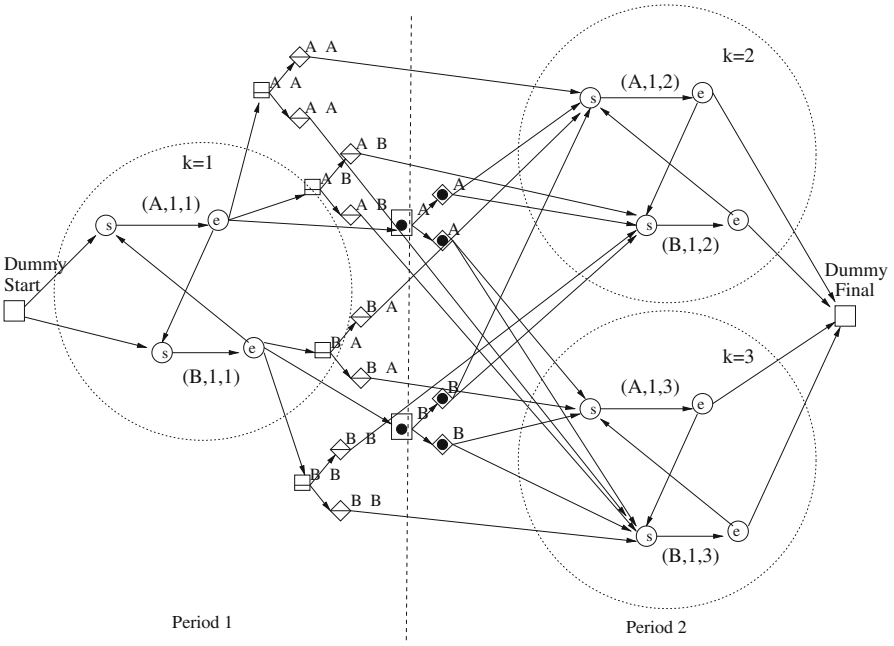

Fig. 2 DP network with two ports, A and B, and a maximum of one visit to each port in each demand scenario part

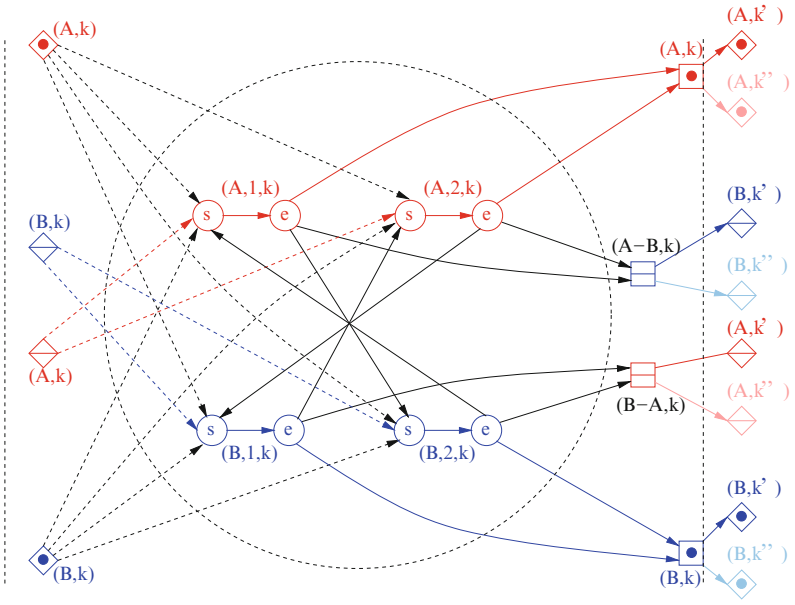

Fig. 3 Detailed demand part of DP network with two ports, A and B, and two possible port visits per port

Figure 2 is a simple example of a DP network with two time periods. There are three demand scenario parts in the network. The start node corresponds to the initial status of the ship. Its status is defined by its position (in some port or at a position at sea) and the amount of cargo on the ship. Figure 3 shows a fuller example of a single demand scenario part of a network.

Table 1 DP network node types

Node	Type	Description
Ⓢ	decision node	start service node: decision made is how much to load or unload during the port visit
ⓔ	decision node	end service node: decision made is the choice of next port visit or to delay until more information is available
⊟	sum-up node	forms the expected future value in port i at current time given the decision to sail to port j in next period
⊡	sum-up node	forms the expected future value in port i at end of current period before the decision of which port to visit next
◇	decision node	split node: decision at current time of which port visit of a given port to visit first in the next period
◈	decision node	split node: decision at end of current period of which port to visit in the next period and which is the first port visit for that port

The different types of nodes in DP networks are listed in the Table 1. Nodes Ⓢ, ⓔ and ◇ and the dummy start node are the decision nodes, and remaining nodes are sum-up nodes. Each port visit (i,m,k) has a start service node Ⓢ and an end service node ⓔ. For each boundary between two periods there is one $\boxminus^{i-j}$ node for every pair of ports i and j. This is associated with a journey from port i to port j in a later period starting before the period boundary at which the demands in the next period will be revealed. Each $\boxminus^{i-j}$ node is linked to a set of $\diamondsuit^{i-j}$ decision nodes, one for each demand scenario node in the following period, and its action is to sum up the expected costs at these nodes. Each $\diamondsuit$ is associated with one future demand scenario part and one port and selects the first port visit for that ship at that port. For example, in Fig. 2, a ship can go from end service node of port visit $(\mathrm{A},1,1)$ to sum-up node $\boxminus^{\mathrm{A-B}}$ and sail to the start service node related to physical port B in period 2 through the split nodes. (However in this example there is only one port visit per port in period 2, so unlike the more general case shown in Fig. 3, there is no choice in this example at the split node.) The horizontal line inside the node is used to signify that the time window of this node is the whole period including the node.

Table 2 Edge costs and times in DP network

Edges	Edge costs	Min edge time
Ⓢ → Ⓔ	$-d^{Q}_{imk}(g-g')-d^{T}_{imk}t^{S}_{imk}-d^{A}_{imk}$	$\lvert g-g'\rvert/W_v$
◇ → Ⓢ	$P_k C_{ii'v}$	$\tilde{T}_{ii'v}$
Ⓔ → Ⓢ	$P_k C_{ii'v}$	$\tilde{T}_{ii'v}$

Another sum-up node $\boxdot^{i}$ is on the boundary between two periods. This is associated with a journey from port i to any port in the next period after the demands in next period are known. Each $\boxdot^{i}$ node is linked to a set of $\diamondsuit^{i}$ nodes, one for each demand scenario part. The decision made of which port to sail first at a $\diamondsuit$ node depends on the future demands in its scenario, so can be different in different demand scenario parts. The dot inside the node is used to signify that the arrival time at this node is fixed on the period boundary. In the DP network the decision nodes are Ⓢ and Ⓔ and ◇: Ⓢ and Ⓔ relate to decisions within the current demand scenario part and ◇ to the initial visit decisions in the following period.

Within a demand scenario part of the network, there can be edges from end service nodes Ⓔ to start service nodes Ⓢ. These edges are the travelling edges, and they have the associated travelling times and costs. Each end service node Ⓔ (related to physical port i) is linked with several sum-up nodes $\boxminus^{i-j}$ and one sum-up node $\boxdot^{i}$. The port visits of the same physical port share the same sum-up nodes. For example, in Fig. 3, both end service nodes of port visit $(\mathrm{A},1,k)$ and $(\mathrm{A},2,k)$ are linked with sum-up nodes $\boxminus^{\mathrm{A-B}}$ and $\boxdot^{\mathrm{A}}$. Since a sum-up node $\boxdot$ is on the boundary between periods, its time is fixed so the edges from node Ⓔ to node $\boxdot$ may have nonzero transition times on them. This corresponds to ships delaying their journeys until more demand information is available.

The edge costs in the DP network are derived from the objective function (19) of the ship routing subproblem, and are listed in the Table 2. Here g is the amount of commodity on board the ship when it arrives at start service node (i,m,k), while g' is the amount of commodity on board the ship at end service node (i,m,k). So the difference between them, $|g-g'|$, is the loading or unloading quantity in port visit (i,m,k). W_v is the (constant) loading or unloading rate for ship v, P_k is the cumulative probability of reaching node k in the demand scenario tree, $C_{ii'v}$ is the travelling cost from port i to port i' by ship v, and $\tilde{T}_{ii'v}$ is the (undelayed) sailing time from port i to port i'. Other edges which are in the network but not included in the above table have zero costs and zero minimum edge times and are used to define the stochastic structure of the network.

Every node in the network has a window for its visit time, so this is a stochastic DP problem with time windows. The time window for $\boxdot$ nodes is the single time of its period boundary, and the time window of every other node is initialized to be the same as the period in which it lies. However if there are more restrictive windows

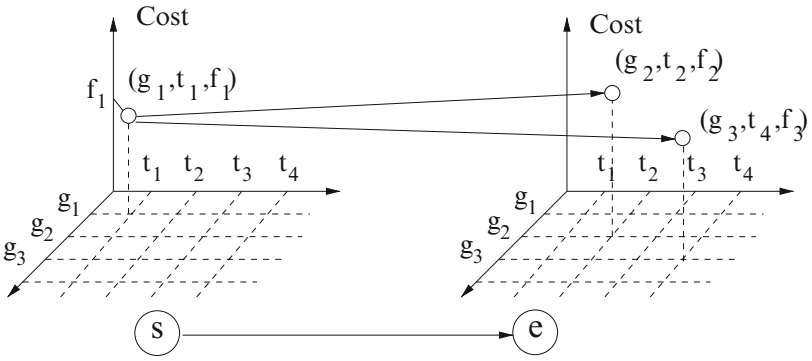

Fig. 4 Example of a service

for some nodes defined in the problem, the time windows on these nodes would be reduced, and all windows can also be reduced in the course of the solution by the B&B method.

Dynamic Programming Formulation

Since there are many different types of nodes in the DP network it is convenient to be able to refer to any node by a general single index, l. Each node l in the DP network for ship v has a function, denoted by $f_{lv}(t,g)$, giving the expected cost to the final node from node l when the node visit time is t and the amount on the ship is g. In our problem, there is a time window for the visit time at each node (i.e., $t \in [\tilde{A}_{lv}, \tilde{B}_{lv}]$). Since a sum-up node ⊡ is on the boundary between periods its time is fixed and so its time window has zero width and there is only one time point in its cost function, which therefore only depends on the quantity on board. Treating both t and g as continuous quantities makes the problem difficult to solve, so instead we restrict (t,g) to a discrete grid of values, $T \times G_v$. Consequently the loading quantity is also discrete as it is the difference in g between the start and end service nodes. An example of part of the $T \times G_v$ grid and a port service are shown in Fig. 4. This shows the expected cost f_1 in a start service node Ⓢ at start of service time t_1 with quantity g_1 on board a ship, and the state reached in the end service node ⓔ for two of the possible loadings that will be considered in calculating the best value of f_1. For instance, point (t_4, g_3) corresponds to a service lasting $t_4 - t_1$ with the ship's load increased by $g_3 - g_1$.

The direction of solving stochastic dynamic programming is from dummy final node to the dummy start node. In the dummy final node L the cost function $f_{Lv}(t,g)$ is initialized to zero and on all other nodes is initialized to infinity (however, if we

want to give a reward for a ship finishing early or having cargo on board at the end, then a more general $f_{Lv}(t,g)$ can be defined).

The values of $f_{lv}(t,g)$ are calculated recursively as follows, the calculations being for all ships v, all nodes l, all $t \in T \cap [\tilde{A}_{lv}, \tilde{B}_{lv}]$, and all $g \in G_v$.

- For l a Ⓢ start service node and l' the corresponding end service node Ⓔ:

$$f_{lv}(t,g) = \min_{g' \in G_v : g' \geq g} \left\{ f_{l'v}(\tilde{T}_{lv}(t, g'-g),\ g') \ - d_l^Q(g'-g) - d_l^T t - d_l^A \right\},$$

  where $\tilde{T}_{lv}(t,\Delta g) = \min\{t' \in T : t' \geq t + |\Delta g|/W_v\}$ is the loading time rounded to the closest higher discrete time, and assuming node l corresponds to a visit to port i , $d_l^Q = d_i^Q$, $d_l^T - d_i^T$, and $d_l^A = d_i^A$ as given in Table 2. The recurrence above refers to a supply port. For a demand port the $g' \geq g$ is replaced by $g' \leq g$.
- For l a ◇ split node or Ⓔ end service node:

$$f_{lv}(t,g) = \min_{l' : l \to l'} \ \min_{\max\{\tilde{A}_{l'v}, t + \tilde{T}_{ll'}\} \leq t' \leq \tilde{B}_{l'v}} \{ f_{l'v}(t', g) + \tilde{C}_{ll'v} \},$$

  where if node l corresponds to a visit to port i, $\tilde{C}_{ll'v}$ is the cost of edge $l \to l'$ for ship v as shown in the Table 2, and $\tilde{T}_{ll'}$ is the transition time from a Ⓢ node l to a Ⓔ node l', and for other cases is zero.
- For l a ⊟ or ⊡ sum-up node:

$$f_{lv}(t,g) = \sum_{l' : l \to l'} f_{l'v}(t,g)$$

The goal is to find the cost function $f_{Fv}(t)$ at the start dummy node F and the tree of schedules for the ship, which can be found by tracking forward from F.

Algorithm for Solving Subproblems

The most common algorithms for the shortest path problem with time windows are labelling algorithms, see [12, 13, 15]. These algorithms assign a label to each node in the network giving the cost of the currently known shortest path from the node to the final node. The algorithms repeatedly recalculate the labels for all the nodes in the network (in an order determined by some heuristic rules), until there is no improvement in the label of any node.

In this chapter we are considering a stochastic model whose objective is to minimize the expected cost. This requires the problem of finding a single shortest path which occurs in the deterministic case to be replaced by the problem of finding a tree of shortest paths. Each label associated with a node is now the lowest expected future cost known from the node to the end of the planning horizon for a specific node visit time and quantity on board. The set of all labels at a node therefore

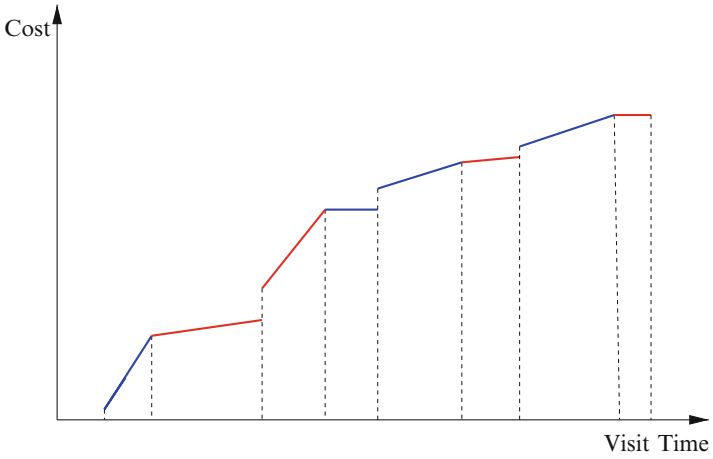

Fig. 5 Cost function

define this expected cost as a function of the visit time and quantity on board. An example of dependency of a cost function with visit time is shown in Fig. 5. The cost functions in our problem are increasing functions of time. In a deterministic DP network the shortest path calculation can be performed either from the start or from the terminal nodes. In the stochastic case where the expected costs are required the calculation starts at the terminal nodes. The iteration is started by setting the cost function to zero at this dummy final node and to infinity at all other nodes. The stochastic DP calculation then iteratively updates the cost function on each node in the network from all the nodes on its outgoing edges.

Because the graphs in our problems contain directed cycles, we may not be able to finish the updating by going through the network only once. We therefore have to update the node cost functions iteratively and be prepared to update the cost for one node several times. In an iteration of updating, we go through each node in the network, and for each node we consider all the outgoing edges from the node. If there has been any updating in the end node of an outgoing edge in last iteration, we will update the cost of the start node of the edge using the cost function of the end node. For the sum-up nodes, if one of the corresponding split nodes is updated in the previous iteration, the sum-up node will be updated in the current iteration. Therefore, we use a flag for each node to indicate whether or not the node is updated in the last DP iteration.

The number of iterations required during the updating is highly dependent on the order in which the nodes of the network are updated. Before starting to update the cost functions we order the nodes as follows. First we calculate the minimum number of directed edges from each node to the final dummy node. Then the nodes are ordered so that nodes closer to the final dummy vertex have lower index than those farther away. Then in each iteration the costs are updated in order of increasing node index. Once there has been no change in the cost of any node in a complete

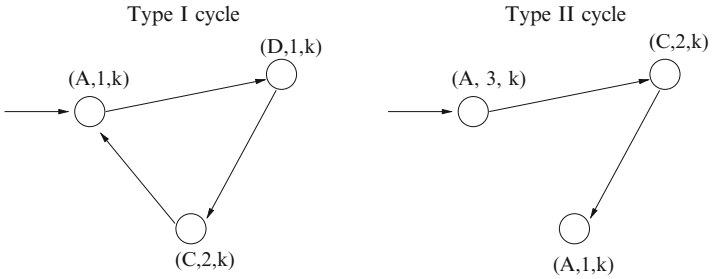

Fig. 6 Cycles of Type I and II

pass through all the nodes, then the optimal costs have been found and we choose the least cost from the cost function of the dummy start node in the network and track the shortest tree through the network from the dummy start node.

Cycle Elimination

Because there are several port visits for each port and the order of port visits to different ports is not predetermined by the structure of the network, it is possible to generate port visit paths which are not logically possible. There are two types of such impossible paths. Type I means that the ship returns to some port visit which occurred for that ship before, while Type II means that the ship route contains a port visit (i, m_1, k) and a later port visit (i, m_2, k) where $m_2 \leq m_1$. Examples of these two situations are shown in Fig. 6. We refer to both of these cases as cycles even though Type II may not include a cycle in the graph of port visits.

When there is a cycle of Type I, a port visit occurs more than once, and in this case the value of A_{svimk} will be greater than 1 and equal to the number of times the port visit occurs. Also the T_{svimk} and Q_{svimk} quantities will be the total over all the times the port visit occurs. Allowing cycles gives a relaxation of the true situation in our model, so bounds allowing them are still valid. However, a solution with cycles is not logically feasible. However allowing cycles gives a relaxation of the true situation, so we do not need to eliminate all the cycles when solving subproblems and can add the tree of routes including cycles into the master problem. Then we can eliminate cycles in the B&B algorithm by splitting a time window.

A K-cycle is defined to be a cycle of length K and elimination of K-cycles is well described in the literature, by Irnich and Villeneuve [22] and Irnich and Desaulniers [21]. However, it is not easy to avoid all the cycles with different lengths when solving the subproblems, and doing that is time consuming. We therefore only eliminate the 2-cycles. In our DP network, we divide a port visit into two nodes, a start service node and an end service node, so a 2-cycle in our problem is different from its original definition. See Fig. 7 for example. In the example, the start service and end service nodes for a port visit are regarded as a big node, and the definition of a 2-cycle is based on the corresponding port visits rather than the real nodes of the network.

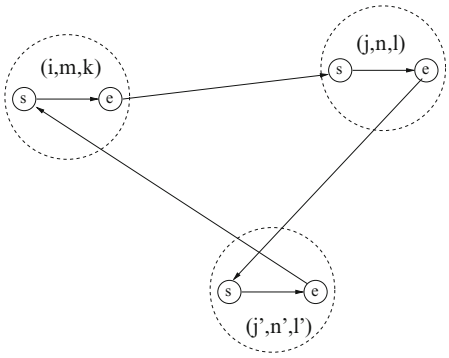

Fig. 7 A Type I cycle in our DP network

In our problem, cycles can only occur within a demand scenario part of the DP network, and there is no cycle crossing the period boundary or between different scenarios at the same period. Hence we only need to consider cycle elimination for start and end service nodes in the network. To eliminate 2-cycles using the method introduced in [22], at each Ⓢ and Ⓔ node and for every time, we store the next port visit following the current one for the best and second best path for each time point. If there is a cycle when using the best solution, we use the second best solution instead. This usually allows us to avoid 2-cycles of Type I and II and we call paths satisfying this rule *allowed paths*. Note that all legal paths are allowed paths. We need to keep updating the best and second best solution on each node during the updating. The best path is the best among all allowed paths, while the second best is the best among all allowed paths where the next port visit is different from the best path.

Branch and Bound

The optimal solution of the stochastic ship routing problem must generate feasible schedules of all ships. However the master problem is a relaxation and may yield an infeasible ship schedule, either because the schedule is a mixture of schedules with different sequences of port visits or because it contains a cycle of port visits. To avoid this we use B&B. At each node of B&B tree a master problem with the discrete requirements relaxed is solved using column generation method. If the solution of this problem is not feasible, we branch so as to eliminate one of these infeasibilities. The columns generated from subproblems are kept in the master problem for other B&B nodes, only the infeasible column is deleted by setting the upper bound of the column to zero.

There are many possible choices for the branching strategy. We branch on infeasibilities in the following order.

If there are columns with positive weight in the solution that correspond to a path with a cycle, then we first branch on a time window so as to eliminate a cycle. Assume that port visit (i,m,k) is involved in a cycle. Let $\{t_{imk}^{S1},\ldots,t_{imk}^{SK}\}$ be discrete start service times associated with the port visit (i,m,k). Let $\bar{t}_{imk} = 1/K \sum_{y=1\ldots K} t_{imk}^{Sy}$ denote the average of these start service times. We do branching by splitting the time window $[\tilde{A},\tilde{B}]$ for the start service time of port visit (i,m,k). Since the width of the port visit time window is also reduced in child nodes, there is less chance of getting other cycles later in the solution.

If there are no cycles in the solution but there are fractional port visit variables, then a branch is made so as to either force a port visit to occur or not to occur. For a port i and node k, the set of port visit variables y_{imk} satisfies $y_{i1k} \geq y_{i2k} \geq y_{i3k} \geq \cdots \geq y_{i,M-1,k} \geq y_{iMk}$ and to be feasible all values must be 0 or 1. We first calculate for each combination of (i,k) the difference between consecutive pairs of variables and choose the maximum difference:

$$Y_{i,k} = \max_{1\leq m\leq M-1} \{y_{i,m+1,k} - y_{i,m,k}\}$$

We then choose the minimum value for $Y_{i,k}$, and choose the maximum value of y_{imk} which is less than 1 and branch on that variable. If the chosen $y_{imk} \geq 0.5$, we branch first on $y_{imk} = 1$ and the other branch is $y_{imk} = 0$. If the value of chosen $y_{imk} < 0.5$, we branch first on $y_{imk} = 0$ and the other branch is $y_{imk} = 1$.

When in a branch where $y_{im'k}$ is set to 0, no port arrivals (i,m,k) can occur for $m \geq m'$. So we delete all the port arrivals (i,m,k) (where $m \geq m'$) as well as all the edges linked with these port arrivals from the network of each ship. If $y_{im'k}$ is set to 1 in a branch, no update happens for the structure of the ship networks. However, a small artificial negative cost is added to each edge from the start service node of port visit (i,m',k) to its end service node, which makes port visit (i,m',k) more attractive than port arrivals (i,m,k) for $m \geq m'$.

If there are no cycles or non-integer y_{imk}, then we calculate the flow $x_{imkjnlv}$, where $x_{imkjnlv} = \sum_{s\in R_v;(i,m,k)\to(j,n,l)\in E_s} x_{sv}$. This quantity defines whether or not ship v sails from port visit (i,m,k) to port visit (j,n,l). For each (j,n,l), we find the maximum fractional value for $x_{imkjnlv}$. Then from these maximum values we choose the minimum value over (j,n,l). The formulation for this process is shown as follows:

$$\min_{j,n,l} \max_{i,m,k,v} \{x_{imkjnlv}\}$$

If the value of the chosen variable is less than 0.5, we branch first on $x_{imkjnlv} = 0$ and $x_{imkjnlv} = 1$ on the other branch. In the branch where $x_{imkjnlv}$ is set equal to 0, the ship v does not sail from (i,m,k) to (j,n,l). Hence all corresponding edges are deleted from the network of ship v. In the branch where $x_{imkjnlv}$ set to 1, we delete all the edges for ship v coming out of (i,m,k) except those going into (j,n,l). For all other ships, the edges from (i,m,k) to (j,n,l) are deleted from the networks.

Table 3 Properties of test examples

EX	Ports	Max arrival	Scenario nodes in tree	Planning periods	Branches	Ships
a1	3	2	3	2	2	2
b1	5	3	3	2	2	2
b2	5	3	3	2	2	2
b3	5	3	3	2	2	2
c1	5	3	7	3	2	2
c2	5	3	7	3	2	2
c3	5	3	7	3	2	2
d1	6	4	7	3	2	3
d2	6	4	7	3	2	3
d3	6	4	7	3	2	3
f1	5	3	13	3	3	2
f2	5	3	13	3	3	2
g1	6	3	13	3	3	3
g2	6	3	13	3	3	3
g3	6	3	13	3	3	3
h1	8	4	40	4	3	3
h2	8	4	40	4	3	3

Stochastic ship routing problems are computationally demanding, and as a result there is the danger that the B&B search may terminate because of time or memory limits before finding an acceptable feasible solution. Depth-first search, although not the fastest B&B search strategy for proving optimality, has the advantage of finding feasible solutions early. Best-first B&B algorithm is a better strategy for proving optimality, and both strategies can be combined by first using depth-first search to find an early integer solution and then switching to best-first search to produce better bounds. This mixed strategy worked well on some examples; however, all the results reported in the next session use depth-first search only and were solved to zero gap.

Examples and Results

To test the models and solution methods developed in this chapter, a set of test problems has been built. The implementation is written in C and CPLEX10.0 is used to solve the sequence of LPs in each B&B node of the master problem. The ship subproblems are independent of each other and are solved in parallel using OpenMP on a 4-core processor. The data structures needed to represent the networks in the subproblems are generated once only before the start of the optimization.

Table 3 gives the characteristics of each test problem. Example $a1$ is very small and is used to illustrate the details of a solution, including the visit sequences, start service time, quantity on board each ship, and the storage levels. All of these

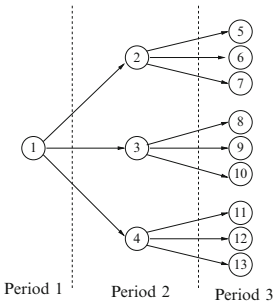

Fig. 8 Scenario tree of Example g1

Table 4 DP and master problem dimensions

EX	Nodes	Edges	(i,m,k) Combinations	Constraints
a1	56	82	18	152
b1	137	706	45	372
b2	137	706	45	372
b3	137	706	45	372
c1	347	1786	105	862
c2	347	1786	105	862
c3	347	1786	105	862
d1	416	2335	126	1033
d2	416	2335	126	1033
d3	416	2335	126	1033
f1	632	3421	195	1607
f2	632	3421	195	1607
g1	758	3421	234	1928
g2	758	4477	234	1928
g3	758	4477	234	1928
h1	3170	23,481	960	7898
h2	3170	23,481	960	7898

details are given as an example later in this section. The examples named with the same first letter are problems with the same physical ports layout and the same demand scenario tree structure, but different initial inventory levels and demand rate situations at each port. The "Max Arrival" column gives the maximum number of possible arrivals for each port in each demand scenario part, which is the parameter M in the formulation introduced in section "Master Problem". The "Scenario Nodes", "Planning Periods" and "Branches" columns give the structure of the scenario tree. For example, in example g1, there are 13 demand scenario parts, 3 time periods, and 3 branches for each period in the scenario tree, which indicates a scenario tree as shown in Fig. 8 (Table 4).

Table 5 Computational results (elapsed time in sec)

EX	B&B nodes	Columns	Total time	Subprob time	Master iters
a1	6	56	0.8	0.6	24
b1	78	1251	13	11	497
b2	177	3079	31	25	1407
b3	219	4204	47	41	1973
c1	81	2435	20	18	879
c2	87	3948	26	21	1633
c3	237	4757	57	48	1978
d1	564	6206	120	103	2649
d2	63	1353	15	14	284
d3	750	6945	138	105	2954
f1	405	9034	439	379	3799
f2	138	3623	126	118	1181
g1	342	7241	403	352	2805
g2	624	11,557	705	611	4731
g3	132	4109	181	161	1298
h1	3598	30,753	3690	3112	43,850
h2	2987	31,983	3371	2958	40,791

The computational results given in Table 5 are: the number of branch-and-bound nodes used to find the optimal discrete solution, the total number of columns generated from the subproblems, the total solving time, the elapsed time for solving the subproblems, and the total number of column generation iterations in the master problem. Examples a1–c3 are relatively small and can be solved within a minute. However, when the problem size is increased, the solving times for the later examples increase sharply. Another factor which may effect the solving time is the initial storage levels and demand situations. For instance, examples f1 and f2 have the same problem structure, but different initial storage levels and demand situations, and f2 is solved much faster than f1. This is because the initial storage levels and demand situations are related to the number of visits to each port in each demand scenario part. If there is sufficient initial storage at a port, fewer visits may be required, which reduces the length of the visiting sequences for ships and makes the problem easier to solve.

As previously discussed, because of the size of the DP networks, the major solving time in each example is taken in solving the ship subproblems, and Table 5 indicates that this takes around 75–94 % of the total time. In the tests the ship subproblems are solved in parallel using one core per ship.

Some detailed solutions are given based on two of the above examples. In example c1, there are 5 ports, and ports A, B, and C are customer ports and ports D and E are supply ports. The left-hand side of Fig. 9 shows the demand scenario tree of the example, and the demand trend changes in each demand scenario part. The tree of routes on the right-hand side of Fig. 9 shows the ship routes in the solution of c1. In the figure, ships choose different routes according to the different demand situations in each period. For instance, ship 1 visits the different ports in the upper and lower cases of period 2, since in the upper case the demands at ports A and B go up while

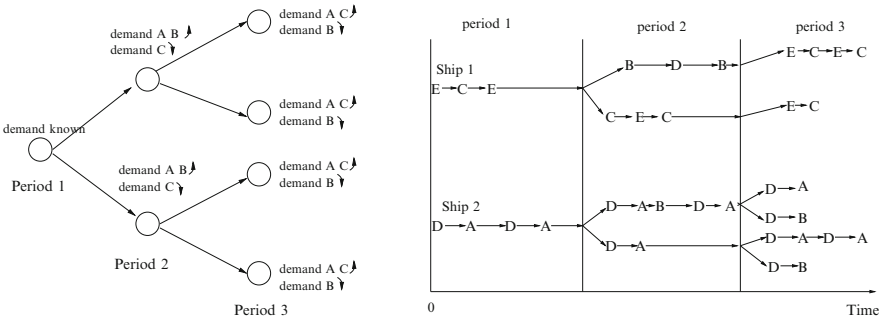

Fig. 9 Solution of Example c1

the demand at port C goes down, and in the lower case the demand situations are just the opposite. In period 3, ship 1 does nothing in the lower case, and this is because all of the demands are satisfied in the case so that there is no need to travel any further.

Figure 10 shows the optimal solution for b1. The physical routes, inventory levels, and quantities on board ships are shown. The changes in the storage of each consumer port and on ships as a function of time can be clearly seen. In period 1, ship 1 sails the route D→A→D. There is an unloading service made by the ship at port A so that there is an increase in the storage level at port A. There are also two visits made by ship 2 to port C, so the storage level of port C goes up twice during the period. There is no visit to port B for the whole period, and the stock level of port B goes down throughout the period because of the constant demand rate. A similar situation can be seen in period 2 from the same figure.

Conclusion

In this chapter, we propose a solution approach to solve the stochastic ship routing problem with inventory management at the ports. The only uncertainty considered is the demand levels at the ports. A Branch and Price algorithm is presented. A master problem is formulated as a set partitioning model including inventory constraints, while a subproblem for each ship is solved by dynamic programming to find the least reduced cost columns for the master problem. The optimal integer solution is searched along the B&B tree and column generation method is used to solve the relaxed LP iteratively in each B&B node.

The ship routing subproblems are stochastic dynamic programming problems, and they are solved by a backward labelling algorithm. The method we use is analogous to the methods that have been used in the deterministic case, but have had to be extended to deal with the scenario branching in the stochastic case. The minimum expected costs from the start node to the final dummy node is calculated.

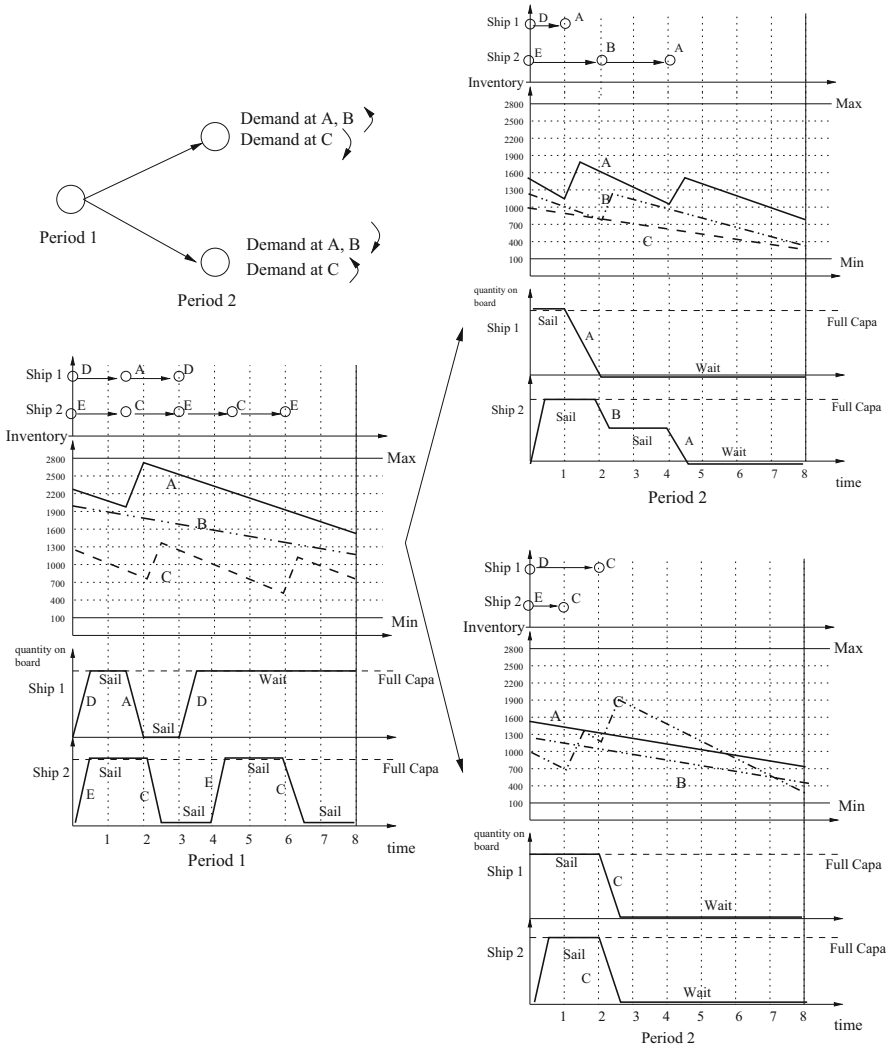

Fig. 10 Solution of Example b1

Because of the complicated DP network, there are many possible cycles (which are not feasible in a solution). Two-cycles are eliminated when solving the subproblems and other cycles with length greater than 2 are eliminated during the B&B algorithm by splitting the time windows. Because the ship subproblems are independent of each other, OpenMP is used to solve them in parallel on a multi-core computer.

From the computational experience, our decomposition method is able to solve medium sized examples. A set of test examples with different geographical port layouts, number of ships, scenario trees, and initial storage situations were built

and were solved by the decomposition method. Our computational experience shows that around 75–94 % of the elapsed time to solve the problem is used to solve the ship subproblems, even when subproblems are solved in parallel. The rest of the elapsed time is used to do B&B administration and solve the LPs. Because of the need to model on entire scenario tree, the stochastic problems become large, even for a small transport network, and this limits the size of problem that can be solved. Generating useful columns in a heuristic way a priori is a possible area for further work. The generated columns can be added into the master problem to give a warm start, which should reduce the solution times and allow larger problems to be solved.

The methods in this chapter naturally extend to cases where ships can divert during sailing (when new demand information becomes available) and cases where ships can alter their speed. These cases give rise to nonlinear subproblems (with whole problem becoming a stochastic nonlinear integer programming problem). However because the subproblems can be solved by discretization and DP the solution approach given in this chapter can still be applied.

References

1. Appelgren, L.: A column generation algorithm for a ship scheduling problem. Transp. Sci. **3**, 53–68 (1969)
2. Appelgren, L.: Integer programming methods for a vessel scheduling problem. Transp. Sci. **5**, 64–78 (1971)
3. Bendall, H., Stent, A.: A scheduling model for a high speed containership service: a hub and spoke short-sea application. J. Marit. Econ. **3**(3), 262–277 (2001)
4. Bertsimas, D.: A vehicle routing problem with stochastic demand. Oper. Res. **40**(3), 574–585 (1992)
5. Christiansen, M.: Decomposition of a combined inventory and time constrained ship routing problem. Transp. Sci. **33**(1), 3–16 (1999)
6. Christiansen, M., Fagerholt, K.: Robust ship scheduling with multiple time windows. Nav. Res. Logist. **49**, 611–625 (2002)
7. Christiansen, C., Lysgaard, J.: A branch-and-bound algorithm for the capacitated vehicle routing problem with stochastic demands. Oper. Res. Lett. **35**, 773–781 (2007)
8. Christiansen, M., Nygreen, B.: A method for solving ship routing problems with inventory constraints. Ann. Oper. Res. **81**, 357–378 (1998)
9. Christiansen, M., Nygreen, B.: Modelling path flows for a combined ship routing and inventory management problem. Ann. Oper. Res. **82**, 391–412 (1998)
10. Christiansen, M., Fagerholt, K., Ronen, D.: Ship routing and scheduling: status and perspectives. Transp. Sci. **38**(1), 1–18 (2004)
11. Crary, M., Nozick, L., Whitaker, L.: Sizing the U.S. destroyer fleet. Eur. J. Oper. Res. **136**, 680–695 (2002)
12. Desrochers, M., Soumis, F.: A generalized permanent labeling algorithm for the shortest path problem with time windows. INFOR **26**(3), 191–211 (1988)
13. Desrochers, M., Soumis, F.: A reoptimization algorithm for the shortest path problem with time windows. Eur. J. Oper. Res. **35**, 242–254 (1988)
14. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. **40**, 342–354 (1992)

15. Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F.: Time constrained routing and scheduling. In: Network Routing. Handbooks in Operations Research and Management Science, vol. 8, pp. 35–139. North-Holland, Amsterdam (1995)
16. Dror, M., Trudeau, P.: Stochastic vehicle routing with modified saving algorithm. Eur. J. Oper. Res. **23**, 228–235 (1986)
17. Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with stochastic demands: properties and solution frameworks. Transp. Sci. **23**(3), 166–176 (1989)
18. Gendreau, M., Laporte, G., Seguin, R.: An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp. Sci. **29**(2), 143–156 (1995)
19. Gunnarsson, H., Ronnqvist, M., Carlsson, D.: A combined terminal location and ship routing problem. J. Oper. Res. Soc. **57**, 928–938 (2006)
20. Hjorring, C., Holt, J.: New optimality cuts for a single-vehicle stochastic routing problem. Ann. Oper. Res. **86**, 569–584 (1999)
21. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. Les Cahiers du GERAD G-2004-11 (2004)
22. Irnich, S., Villeneuve, D.: The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. INFORMS J. Comput. **18**(3), 391–406 (2006)
23. Kleywegt, A., Nori, V., Savelsbergh, M.: Dynamic programming approximations for a stochastic inventory routing problem. Transp. Sci. **38**, 42–70 (2004)
24. Mehrez, A., Hung, M., Ahn, B.: An industrial ocean-cargo shipping problem. Decis. Sci. **26**(3), 395–423 (1995)
25. Ronen, D.: Marine inventory routing: shipments planning. J. Oper. Res. Soc. **53**, 108–114 (2002)
26. Sherali, H., Al-Yahoob, S., Hassan, M.: Fleet management models and algorithms for an oil-tanker routing and scheduling problem. IIE Trans. **31**, 395–406 (1999)
27. Shih, L.H.: Planning of fuel coal imports using a mixed integer programming method. Int. J. Prod. Econ. **51**, 243–249 (1997)

Investigation of Data Regularization and Optimization of Timetables by Lithuanian High Schools Example

Jonas Mockus and Lina Pupeikiene

Abstract In practice, we must first assign teachers and students to subject-groups for school applications. In the Lithuanian high schools, the number of subject-groups can be very large, since students are free to select just a small subset of optional subjects.

The experimental investigation of this chapter did show that in such conditions, some regularization of subject-groups is needed for prior to optimization. The regularization is a sequential elimination of the timetable-breakers. A timetable-breaker is a student or teacher the presence of which in a subject-group is most harmful for the timetable. The automatic elimination of breakers is difficult due to many subjective factors. In practice it is done by an expert trying to change the subject-group accordingly. In the case of teachers the personal communication is used, if the group changes do not help.

The application of optimization algorithms for timetabling data regularization is the new result of this work. New also is the experimental investigation applying optimization algorithms in 39 Lithuanian high schools.

Keywords School Timetabling • Optimization • Data regularization • Experimental investigation

Introduction

No polynomial time algorithms are known for exact optimization of timetables in real high schools. Thus, approximations and heuristics are applied. Efficiency of calculations is the main objective, convergence to an exact solution is a desirable property.

In this chapter, the experimental comparisons of the efficiency of timetable data regularization and optimization are performed. The algorithms are compatible with

J. Mockus (✉) • L. Pupeikiene
Institute of Mathematics and Informatics, Vilnius University,
Akademijos 4, 08663 Vilnius, Lithuania
e-mail: jmockus@gmail.com; lina.pupeikiene@gmail.com

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_9

the Lithuanian high school practice and flexible enough for adaptation to different high schools.

The necessary timetabling conditions are represented as hard constraints. Penalty points for violation of soft constraints in the general framework of Pareto optimality.

The school timetable d can be described formally as a binary four-dimensional array representing decisions of timetable developers

$$d = [d_{m,s,r,t}]_{M\times S\times R\times T}, \tag{1}$$

where M is a set of teachers, S is a set of students, R is a set of classrooms, and T is a set of weekly time-slots.

For example, the decision $d_{m,s,r,t} = 1$ defines a lesson of a teacher m attended by a student s in the room r at the time t. The decision $d_{m,s,r,t} = 0$ means no lesson. Denote by A a set of feasible timetables that satisfy the hard constraints. The theoretical aim is to define such feasible timetable $d^* \in A$ which minimizes the soft constraint violations.

A straightforward way to estimate the harmful effects is a sum of penalty points for violation of soft constraints. However, in practice a hard constraint defining the maximal daily hours should be lifted including the unfeasible hours as important penalty factor. Up to 15 daily hours were reached using the initial subject-group formations.

In practice, we must first assign teachers and students to subject-groups for school applications. The mapping $g = g(v)$ defines a subject-group (a group of students selecting an optional subject v and a teacher of this subject). Using the subject-groups g instead of students s and teachers m, we can transform the original decision array (1) into the subject-group decision array as follows:

$$d^g = d^g_{v,g,r,t},\ v \in V,\ g \in G,\ r \in R,\ t \in T. \tag{2}$$

Here G is a set of subject-groups.

In the Lithuanian high schools, the number of subject-groups can be very large, since students are free to select just a small subset of optional subjects In such conditions, some regularization of subject-groups is needed for prior to optimization. The regularization is a sequential elimination of the timetable-breakers. A timetable-breaker is a student or teacher the presence of which in a subject-group is most harmful for the timetable.

The results of experiment investigation of 39 high schools are presented. They show that an expert regularization improves the final optimization results several times (from 160,000 to 40,000 penalty points in large schools). Comparing with an expert decisions the advantage of automatic optimization is speeding up the scheduling process (from 20–60 h to 15–20 min). Unexpected result was that the main advantage of optimization algorithms was speeding-up the subject-group regularization by indicating the timetable-breakers and estimating their harmful effects (saving at least a half of 160–240 h manual work of a good expert).

Overview of Publications

A survey on educational timetabling problems [16] gives an overview of the literature up to 2010. A new survey is in [17]. Different theoretical and practical aspects of school timetabling are regarded in [1–3, 7–9, 18, 22, 26, 30–33]. New developments are described in [4, 8, 10, 11, 13–15, 19, 20, 24, 25].

The web-based software for high school timetabling is described in [6, 28]. In [28], building a master schedule involves assigning courses, teachers, and rooms to time- slots, maximizing usage of resources, and fulfilling student course requests. It is one of most complex tasks performed by administrators at a school each year. This is particularly true for secondary schools in the USA where the number and the variety of constraints (such as multi-day rotation, combined courses, room capacities, teacher teaming, and student grouping, etc.) increase the complexity of the scheduling process.

According to [29] the process of timetable construction is a common and repetitive task for high schools worldwide. In this chapter a generic approach is presented for Greek high schools organized around the idea of solving a significant number of tractable integer programming problems. Variables of the underlying mathematical model correspond to daily teacher schedules while a number of hard and soft constraints are included so as for the model to handle practical aspects that manifest themselves in Greek high schools. By selecting better teacher schedules that exist in subproblems the quality of the overall solution gradually improves. The collected results which are obtained within reasonable time are most promising. The strength of the approach is supported by the fact that it managed to find the best known results for two public instance problems included in the Benchmarking Project for High School Timetabling (XHSTT-20121).

In [27], a general model for the timetabling problem of high schools in Denmark is introduced, as seen from the perspective of the commercial system Lectio, and an adaptive large neighborhood search (ALNS) algorithm is proposed for producing solutions. Lectio is a general-purpose cloud-based system for high school administration (available only for Danish high schools), which includes an embedded application for creating a weekly timetable. Currently, 230 high schools are customers of Lectio, and 191 have bought access to the timetabling software. This constitutes the majority of high schools in Denmark. This large customer base entails a need for a model of the problem which is general enough to suit many different requirements, while still remains tractable by computer aided solution methods. This supports the recent trend of developing general models for timetabling problems.

In [20], authors present the progress on the benchmarking project for high school timetabling that was introduced at PATAT 2008. In particular, they announce the High School Timetabling Archive XHSTT-2011 with 21 instances from 8 countries and an evaluator capable of checking the syntax of instances and evaluating the solutions.

The GA approach was used to evolve timetables for a South African primary and high school [23, 24]. In this paper, one or more of the low-level construction heuristics, namely, the largest degree, split degree, and saturation degree, was/were used to construct timetables during initial population generation of the first phase.

In [25], the school timetabling problem is solved in two phases, both of which employ integer programming. The first phase focuses on day allocations and the second phase solves the rest of the problem. The approach was applied to a test problem.

In [11], a tabu search is applied, to induce school timetables for three Vietnam high schools. Firstly, a greedy search is used to create an initial timetable. This timetable is then improved using tabu search. The moves performed by the tabu search include single moves, swaps, and block moves.

In [8], the KHE general problem solver for school timetabling problems is described. KHE employs the coarse grained parallel processing model and facilitates the sharing of instances and the independent creation of multiple solutions in parallel.

In [14], an XML format is presented for both timetabling and scheduling problems.

In [10], polymorphic ejection chains are introduced, and applied to the problem of repairing time assignments in high school timetables while preserving regularity. An ejection chain is a sequence of repairs, each of which removes a defect introduced by the previous repair. Just as the elements of a polymorphic list may have different types, so in a polymorphic ejection chain the individual repairs may have different types. Methods for the efficient realization of these ideas, implemented in the author's KHE framework, are given, and some initial experiments are presented.

Most of the implementations are specifically designed for particular cases of timetabling and an objective comparison of the methods is difficult. In [7], the data formats are proposed for exchange of timetabling instances and solutions.

On Optimization Algorithm

Different algorithms were applied, mostly well known, in school timetabling publications, including ones in [12]. Here is a short description. Multi-start algorithms are a simple way to provide the convergence when the number of uniformly distributed starting points is large. A disadvantage is a slow convergence.

The next improvement is two algorithms of simple local search, considering only the closest timetables in a deterministic (LD) or random (LR) way. The local search is improved further using an algorithm similar to simulated annealing (SA) with fixed parameters. In SA, permutations are limited to the closest timetables and are performed by closing the gaps between lectures for students and teachers.

In the final improvement, in order to provide independence of the human operators, to save operators' time, and to increase the efficiency of search, the initial temperature and the annealing rate of SA are optimized using the Bayesian approach (BA). The advantage of BA is that it filters-out random deviations.

However, the best results were obtained using greedy heuristics to generate initial timetables and permutation heuristics to optimize the initial schedule. Both heuristics were improved and adapted to Lithuanian high schools by a long experimental work using data of 39 high schools of different sizes. Here is a short description of these heuristics.

Greedy Heuristic for Initial Timetables

1. Creating a random list of subject-groups.
2. Adding to the first group the first lesson and the first feasible cabinet (if not feasible, then adding one with minimum violations).
3. Repeating the step 2 using the remaining groups.
4. Repeating step 3 using the remaining lessons.
5. Calculating the penalty points of the initial timetable.

Permutation Heuristic for Optimization of Initial Timetables

1. Saving the initial timetable with the name BEST.
2. Creating a list of non-feasible groups.
3. Selecting with equal probabilities one of three following actions:
   (a) Setting randomly the new teaching time of randomly selected group from the list 2.
   (b) Interchanging the teaching times of two groups randomly selected from the list 2.
   (c) Interchanging the teaching times of three groups randomly selected from the list 2.
4. If the new timetable is better, it replaces the BEST, otherwise the step 3 is repeated for 60 s.
5. If no improvement, then two or three identical subjects are detected and interchanged.
6. If timetable is improved, the step 3 is repeated, otherwise the step 5 is repeated.
7. Optimization stops after some fixed time, depending on the school size.

The necessary timetabling conditions are represented as hard constraints. Desirable conditions are included as penalty points in the general framework of Pareto optimality. Different schools give different weights to various constraints. The software for evaluating real timetables is included to compare with the results of optimization.

On the Formal Definition of High School Timetabling Problem

This section is meant for a formal description of objectives and constraints that are implemented in the software. The description reflects the conditions of two upper classes of Lithuanian high schools. We think that the formal expressions (an updated version of those in [12]) are necessary for an exact understanding of the new results of this chapter.

Hard Constraints

The school timetable d can be described formally as a binary four-dimensional array [12, 21] representing decisions of timetable developers

$$d = [d_{m,s,r,t}]_{M\times S\times R\times T}, \tag{3}$$

where M is a set of teachers, S is a set of students, R is a set of classrooms, and T is a set of weekly time-slots.

For example, the decision $d_{m,s,r,t} = 1$ defines a lesson of a teacher m attended by a student s in the room r at the time t. The decision $d_{m,s,r,t} = 0$ means no lesson. Denote by A a set of timetables that satisfy the hard constraints. The hard constraints are mandatory therefore the set A of feasible timetables is well defined. Here is a formal definition of the hard constraints:

$$h_1(d) = \sum_{s,r} d_{m,s,r,t} \le 1, \textit{ for all } m,t, \;\; h_2(d) = \sum_{m,r} d_{m,s,r,t} \le 1, \textit{ for all } s,t, \tag{4}$$

$$h_3(d) = \sum_{m,s,r\in R_j} d_{m,s,r,t} \le R_j^{max}, \textit{ for all } t,\ j \in J, \tag{5}$$

$$h_4(d) = \sum_{s} d_{m,s,r,t} \ge S_{min}, \textit{ for all } m,r,t, \tag{6}$$

$$h_5(d) = \sum_{s} d_{m,s,r,t} \le S_{max}, \textit{ for all } m,r,t, \tag{7}$$

$$h_6(d) = \sum_{m,s,t\in T_i} d_{m,s,r,t} \le T_{max}, \textit{ for all } r,\ i \in I, \tag{8}$$

where the symbol T_i denotes a set of time-slots of the ith week-day, I is a set of week-days, J is a set of integers denoting different classrooms, T_{max} limits daily time-slots, R_j is a set of rooms of type j, and R_j^{max} is the number of available j-rooms. Condition (4) means no simultaneous lessons for teachers. Condition (4) denotes no simultaneous lessons for students. Condition (5) limits the number of classrooms of type j. Conditions (6) and (7) set the lower and upper class-size limits, and condition (8) limits the maximal number of daily time-slots.

These inequalities are written assuming "one-to-one" teacher–subject relation: *teacher* ⇔ *subject*.

The limit T_{max} of daily time-slots is an important hard constraint. However, it is convenient to regard it as a soft constraint with a large violation penalty. The reason is that schools may accept an extra hour if that improves the general timetable.

The decision array $d_{m,s,r,t}$ is for timetable developers. Students select subjects $v = v(m)$, they do not select teachers $m(v)$ of these subjects. Under the *teacher* ⇔ *subject* assumption, the decision array (1) can be directly transformed into the following decision array:

$$d^v = d^v_{v,s,r,t},\ v \in V,\ s \in S,\ r \in R,\ t \in T, \tag{9}$$

where $v = v(m)$ denotes a subject of the teacher m and V is a set of subjects.
Now we define some hard constraints specific to the upper classes of high schools. Denote by $c = c(v)$ students' reward for selecting an optional subject v. Denote by $V_1 \subset V$ a set of optional subjects, by $V_2 \subset V$ a set of mutually exclusive optional subjects, and by $V_3 \subset V$ a set of mandatory subjects. Denote a decision array of the student s as $d^s_{v,s}$, where $d^s_{v,s} = 1$, if the student s selects the subject v, otherwise $d^s_{v,s} = 0$.
Then the condition that a timetable should be based on students' decisions can be defined by the following hard constraint:

$$d^s_{v,s} = \sum_{r,t} d_{v,s,r,t},\ \textit{for all } v, s, \tag{10}$$

or

$$h_7(d) = d^s_{v,s} - \sum_{r,t} d_{v,s,r,t} = 0,\ \textit{for all } v, s. \tag{11}$$

A set of hard constraints is defined by the demand for students of upper classes to earn specific rewards:

$$h_8(d) = \sum_{v \in V_1, r, t} d_{v,s} c(v) = C_1,\ \textit{for all } s, \tag{12}$$

$$h_9(d) = \sum_{v \in V_2, r, t} d_{v,s} c(v) = C_2,\ \textit{for all } s, \tag{13}$$

$$h_{10}(d) = \sum_{v \in V_3, r, t} d_{v,s} c(v) = C_3,\ \textit{for all } s. \tag{14}$$

Here C_1 is the sum of rewards a student needs to obtain by selecting a subset of optional subjects, C_2 is the reward for selecting one of the optional mutually exclusive subjects, and C_3 is the sum of rewards for mandatory subjects. Expressions (12)–(14) define hard constraints for students to obtain the prescribed rewards. Attribution of rewards to subjects is presented in the form of a table.

The numbers of weekly working days and teaching hours are defined by the initial data and remain unchanged during optimization. We considered only individual teachers and students so far.

We must assign teachers to subjects and students to subject-groups for school applications. Assigning teachers to subjects is straightforward if a subject is assigned to no more than one teacher. We assume that at present.

The mapping $g = g(v)$ defines a subject-group (a group of students selecting an optional subject v). Using the subject-groups g and subjects v instead of teachers m and students s, we can transform the original decision array (1) into the group decision array as follows:

$$d^g = d^g_{v,g,r,t},\ v \in V,\ g \in G,\ r \in R,\ t \in T. \tag{15}$$

Here G is a set of groups.

Soft Constraints

We use penalty points [5] to balance conflicting desires represented by soft constraints.

Compactness of the timetable is a desirable property. The compactness can be improved by reducing the number of gaps between lessons by moving lectures in time while obeying the hard constraints:

$$\sum_{i \in I} \sum_{g,r,t \in T_i} |d^g_{v,g,r,t+1} - d^g_{v,g,r,t}| = B(v,d),$$

$$\textit{for a subject } v. \tag{16}$$

$$\sum_{i \in I} \sum_{v,r,t \in T_i} |d^g_{v,g,r,t+1} - d^g_{v,g,r,t}| = C(g,d),$$

$$\textit{for a group } g, \tag{17}$$

where I is a set of week-days. The numbers of gaps $B(v,d)$ and $C(g,d)$ in timetables d are defined for subject-groups g and subjects v, assuming that the names of teachers and students are irrelevant. Then the total number of gaps in the timetable d can be expressed in such a way:

$$B(d) = \sum_v B(v,d), \tag{18}$$

$$C(d) = \sum_g C(g,d). \tag{19}$$

Another indicator of timetable d compactness is the number of free week-days for teachers:

$$W(v,i,d) = \sum_{g,r,t \in T_i} d^g_{v,g,r,t},\ i \in I. \tag{20}$$

The week-day $W(v,i,d)$ is a free week-day for the subject v if the indicator $W(v,i,d) = 0$, otherwise, it is a working week-day. Denote as $I^v_{w(v,d)}$ a set of working week-days for the subject v in the timetable d. A set of working days in the timetable d for the teacher m is a union of working week-days for a set of subjects $V(m)$ delivered by the teacher m

$$I_{w(m,d)} = \cup_{v \in V(m)} I^v_{w(v,d)}. \tag{21}$$

Compactness is improved by reducing the number of working week-days in the timetable d for the teachers I_w

$$I_{w(d)} = \sum_{m \in M} I_{w(m,d)}. \tag{22}$$

In the upper classes of high schools, subject-groups are different from the traditional classes. The traditional classes can be regarded as social groups. A subject-group involves students from different classes united by a set of selected subjects. The simplest subject-group is an individual student.

In Lithuanian schools the smallest subject-group is five students, otherwise, the subject is closed. The subject-groups are split into two parallel sub-groups, if the number of students exceeds the classroom limits, 30 students, as usual. The splitting may change the composition of groups while closing the gaps between the lessons of students. For example, if there are two groups for the same subject, then a gap of some first group student can be closed by swapping this student with some student for the second group.

That is not convenient for teachers. A source of the instability is the student swaps between the parallel subject-groups. However, preventing the swaps we may increase some other undesirable factors. Thus, we consider the subject-group stability as a soft constraint that depends on the policies of a local school. At present, most of the schools regard the stability of subject-groups as a hard constraint.

Formally the desire to stabilize the groups can be expressed as the total number of swaps $\delta(d)$ in the timetable d. Denote by L a set of parallel subject-groups. Denote a group-change indicator for some parallel group $L1 \in L$ and some student $s \in L1$ by this expression

$$\begin{aligned} l(t,s,L1) &= 0, \textit{if } s \in L1 \textit{ and } \bar{s} \in \bar{L1} \textit{ at a time } t, \\ &= 1, \textit{if } s \in \bar{L1} \textit{ and } \bar{s} \in L1 \textit{at a time } t. \end{aligned} \tag{23}$$

Here $\bar{s}$ is an exchange student while swapping $s \in L1$ with some $\bar{s} \in \bar{L1}$. Denote the group $L1$ change at a time t as the difference

$$\delta(d,t,s,L1) = |l(t,s,L1) - l(t+1,s,L1)|. \tag{24}$$

Then the group-change indicator is the total number of group changes in the timetable d:

$$\delta(d) = \sum_{t \in T,\ L1 \in L} \delta(d, s, t, L1). \tag{25}$$

We represent different soft constraints as different objectives $f_j(d)$, $j = 1, \ldots, 6$, for example:

$$f_1(d) = B(d), \tag{26}$$

$$f_2(d) = C(d), \tag{27}$$

$$f_3(d) = I_{w(d)}, \tag{28}$$

$$f_4(d) = \delta(d), \tag{29}$$

$$f_5(d) = T_{max}. \tag{30}$$

Formally T_{max} represents hard constraint (8). However we include it into a set of soft constraints, because by exceeding this constraint we can improve the general timetable. Didactic constraints, such as a desirability of higher proportion of difficult subjects earlier and higher proportions of easy subjects later, are not mandatory in Lithuanian schools. Thus, they create an additional set of soft constraints. It is assumed that the "hard" subjects, such as mathematics, physics, chemistry, and information technology, are more difficult. The humanities, such as ethics and arts, are regarded as less difficult. The sports is an example of an easy subject.

Denote by $p = p(v)$ the priority of subject v and by T_t the number of lesson hours at time t. Denote by $\tau_t = T_t/5$ the length of one priority slot. Set the priority indicator $q_t(v) = 5$, if the subject v is in the morning priority slot, set $q_t(v) = 1$ if v is in the evening slot, and set $q_t(v) = 4$ *or* 3 *or* 2, if v is between the morning and evening priority slots.

Then we can write the soft priority constraint as follows:

$$f_6(d) = \rho(d) = \sum_{v \in V,\ t \in T} |p_t(v) - q_t(v)|. \tag{31}$$

By minimizing this soft constraint we may reduce the number of priority violations.

On the Experimental Results

In practice, we must first assign teachers and students to subject-groups for school applications. In the Lithuanian high schools, the number of subject-groups can be very large, since students are free to select just a small subset of optional subjects.

The experimental investigation of this chapter did show that in such conditions, some regularization of subject-groups is needed for prior to optimization. The regularization is a sequential elimination of the timetable-breakers. The automatic elimination of breakers is difficult due to many subjective factors. In practice it is done by an expert who tries to change the subject-group accordingly. In the case of teachers the personal communication is used, if the group changes do not help.

A straightforward way to estimate the harmful effects is a sum of penalty points for violation of soft constraints. However, in practice a hard constraint defining the maximal daily hours should be lifted including the unfeasible hours as important penalty factor. Up to 15 daily hours were reached using the initial subject-group formations.

The results of experiment investigation of 39 high schools are presented. They show that an expert regularization improves the final optimization results several times (from 160,000 to 40,000 penalty points in large schools). Comparing with an expert decisions the advantage of automatic optimization is speeding up the scheduling process (from 20–60 h to 15–20 min). Unexpected result was that the main advantage of optimization algorithms was speeding-up the subject-group regularization by indicating the timetable-breakers and estimating their harmful effects (saving at least a half of 160–240 h manual work of a good expert).

Figures 1 and 2 show how the average optimization results depend on school size and optimization method.

The four groups of columns in Fig. 1 correspond to schools of different sizes, the middle columns show optimization without regularization, the right columns show optimization after regularization, and the left column represents actual expert decisions of the real schools.

The three groups of columns in Fig. 2 show the same results arranged in different ways. The middle chart shows optimization without regularization, the right chart shows optimization after regularization, and the left chart represents actual expert decisions of the real schools. Different columns represent schools of different sizes: from the largest to the smallest ones.

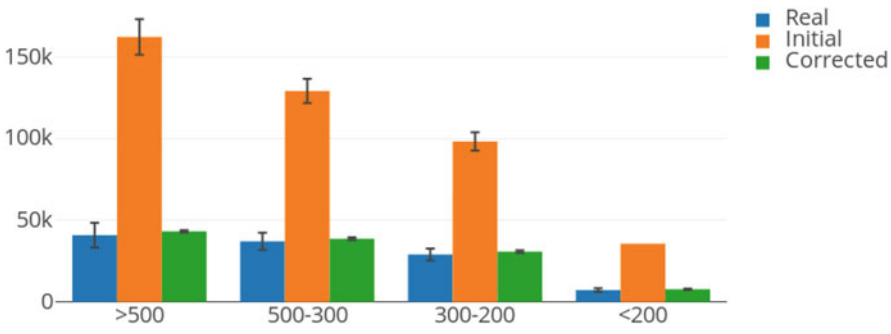

Fig. 1 Average optimization results, initial representation

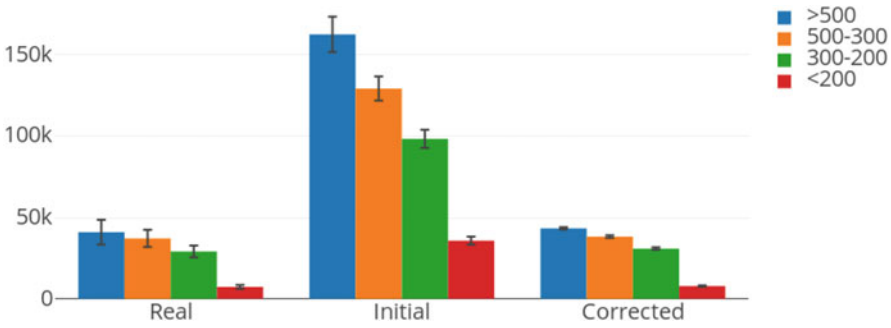

Fig. 2 Average optimization results, transformed representation

Conclusions

The results of experiment investigation of 39 high schools show that an expert regularization improves the final optimization results several times (from 160,000 to 40,000 penalty points in large schools).

Comparing with an expert decisions the advantage of automatic optimization is speeding up the scheduling process (from 20–60 h to 15–20 min).

Unexpected result was that the main advantage of optimization algorithms was speeding-up the subject-group regularization by indicating the timetable-breakers and estimating their harmful effects (saving at least a half of 160–240 h manual work of a good expert).

References

1. Beligiannis, G., Moschopoulos, C., Kaperonis, G., Likothanassis, S.: Applying evolutionary computation to the school timetabling problem: the Greek case. Comput. Oper. Res. **35**, 1265–1280 (2008)
2. Bello, G., Rangel, M., Boeres, M.: An approach for the class/teacher timetabling problem. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT2008, 18–22 August 2008. Universite de Montreal, Montreal (2008)
3. Birbas, T., Daskalaki, S., Housos, E.: School timetabling for quality student and teacher schedules. J. Scheduling **12**, 177–197 (2009)
4. Cohen-Zamir, I., Bar., D.: School time tabling ITT software. In: Practice and Theory of Automated Timetabling (PATAT 2012), 29–31 August 2012, pp. 466–477. Son, Norway (2012)
5. Fudenberg, D., Tirole, J.: Game Theory. MIT, Boston (1983)
6. Kingston, J.: The KTS high school timetabling system. http://sydney.edu.au/engineering/it/~jeff/kts.cgi (2009)
7. Kingston, J.: Data formats for exchange of real-world timetabling problem instances and solutions. In: Proceedings of PATAT10, Belfast, August 2010, pp. 513–516. Queen's University, Belfast (2010)
8. Kingston, J.: Solving the general high school timetabling problem. In: Proceedings of PATAT10, Belfast, August 2010, pp. 517–518. Queen's University, Belfast (2010)

9. Kingston, J.: Timetable construction: the algorithms and complexity perspective. In: Proceedings of PATAT10, Belfast, August 2010, pp. 26–36. Queen's University, Belfast (2010)
10. Kingston, J.H.: Repairing high school timetables with polymorphic ejection chains. In: Practice and Theory of Automated Timetabling (PATAT 2012), 29–31 August 2012, pp. 16–30. Son, Norway (2012)
11. Minh, K.N., Thanh, N.D., Trang, K.T., Hue, N.T.: Using tabu search for solving a high school timetabling problem. Stud. Comput. Intell. **283**, 305–313 (2010)
12. Mockus, J.: Bayesian heuristic approach to global optimization and examples. J. Global Optim. **22**, 191–203 (2002)
13. Moura, A.V., Scaraficci, R.A.: A GRASP strategy for a more constrained school timetabling problem. Int. J. Oper. Res. **7**, 152–170 (2010)
14. Ostler, J., Wilke, P.: The Erlangen advanced timetabling system (EATTS) unified xml file format for the specification of timetabling systems. In: Proceedings of PATAT10, Belfast, August 2010, pp. 447–464. Queen's University, Belfast (2010)
15. Pillay, N.: A study into hyper-heuristics for the school timetabling problem. In: SAICSIT 2010: fountains of computing research, pp. 258–264. ACM for Computing Machinery, New York (2010)
16. Pillay, N.: An overview for school timetabling. In: Proceedings of PATAT10, Belfast, August 2010, pp. 321–335. Queen's University, Belfast (2010)
17. Pillay, N.: A survey of school timetabling research. Ann. Oper. Res. **218**, 261–293, 261–293 (2014)
18. Post, G., Kingston, J.H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Santos, H., Rorije, B., Schaerf, A.: An XML format for benchmarks in high school. In: Proceedings of PATAT10, Belfast, August 2010, pp. 347–352. Queen's University, Belfast (2010)
19. Post, G., Ahmadi, S., Daskalaki, S., Kingston, J.H., Kyngas, J., Nurmi, C.: An XML format for benchmarks in high school timetabling. Ann. Oper. Res. **194**, 385–397 (2012)
20. Post, G., Kingston, J.H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., Schaerf, A.: XHSTT: an XML archive for high school timetabling problems in different countries. Ann. Oper. Res. **218**, 295–301 (2014)
21. Pupeikiene, L., Mockus, J.: School Scheduling Optimization, Investigation and Applications. Lambert Academic Publishing, Saarbrücken (2010)
22. Raghavjee, R., Pillay, N.: An application of genetic algorithms to the school timetabling problem. In: Proceedings of SAICSIT 6–8 October 2008, Nelson Mandela Metropolitan University Port Elizabeth South Africa, pp. 193–199. ACM, New York (2008)
23. Raghavjee, R., Pillay, N.: An informed genetic algorithm for the high school timetabling problem. In: SAICSIT 2010: Fountains of Computing Research, pp. 408–412. ACM for Computing Machinery, New York (2010)
24. Raghavjee, R., Pillay, N.: Using genetic algorithms to solve the South African school timetabling problem. In: Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC'10), pp. 286–292. IEEE, New York (2010)
25. Ribic, S., Konjicija, S.: A two phase integer programming approach to solving the school timetabling problem. In: Proceedings of the International Conference on Information Technology Interfaces (ITI), pp. 651–656 (2010)
26. Santos, H., Uchoa, E., Ochi, L., Maculan, N.: Strong bounds with cut and column generation for class-teacher timetabling. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT2008, 18–22 August 2008. Universite de Montreal, Montreal (2008)
27. Sørensen, M., Stidsen, T.R.: High school timetabling: modeling and solving a large number of cases in Denmark. In: Practice and Theory of Automated Timetabling (PATAT 2012), 29–31 August 2012, pp. 359–364. Son, Norway (2012)
28. Tao, B., Dwyer, R.: Aspen scheduler: a web-based automated master schedule builder for secondary schools. In: Practice and Theory of Automated Timetabling PATAT 2012, pp. 29–31. Sun, Norway (2004)

29. Valouxis, C., Gogos, C., Alefragis, P., Housos, E.: Decomposing the high school timetable problem. In: Practice and Theory of Automated Timetabling (PATAT 2012), 29–31 August 2012, pp. 209–221. Son, Norway (2012)
30. Wilke, P., Killer, H.: Comparison of algorithms solving school and course time tabling problems using the Erlangen Advanced Time Tabling System (EATTS). In: Proceedings of PATAT10, Belfast, August 2010, pp. 427–439. Queen's University, Belfast (2010)
31. Wilke, P., Killer, H.: Walk up jump down - a new hybrid algorithm for time tabling problems. In: Proceedings of PATAT10, Belfast, August 2010, pp. 440–447. Queen's University, Belfast (2010)
32. Wilke, P., Ostler, J.: Solving the school timetabling problem using Tabu search, simulated annealing, genetic and branch and bound algorithms. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT2008, 18–22 August 2008. Universite de Montreal, Montreal (2008)
33. Wilke, P., Ostler, J.: The Erlangen Advanced Time Tabling System (EATTS) unified XML file format for the specification of time tabling systems. In: Proceedings of PATAT10, Belfast, August 2010, pp. 447–465. Queen's University, Belfast (2010)

Dynamic Global Optimization Methods for Determining Guaranteed Solutions in Chemical Engineering

Carlos Pérez-Galván and I. David L. Bogle

Abstract Engineers seek optimal solutions in designing systems but a crucial element is to ensure bounded performance. For example, chemical reactors are often very heavy energy users so it is important to find designs that minimize energy use but the solution must be within strict safety limits.

Currently, the deterministic solution of dynamic systems to global optimality can only be addressed for small problems. The solution of the ordinary differential equation (ODE) systems in a verified way is only able to address low dimensional problems mainly because the integration has to be stopped early due to the overestimation generated by the verified method. Chemical engineering researchers have used a range of techniques to tackle this problem using ways of finding tight over/under-estimators. This chapter will review research work in chemical engineering for such problems and present results of work we are undertaking using interval methods.

In our work a verified solver that constructs upper and lower bounds on the dynamic variables of initial value problem (IVP) for ODEs is used in a dynamic global optimization method (sequential approach). Particular attention is paid to the reduction of the overestimation by means of interval contractors. The solver is used to provide guaranteed bounds on the objective function and on the first order sensitivity equations in a branch and bound framework. Uncertainty can be introduced in the dynamic constraints of the dynamic optimization problem and therefore it is possible to account for it in a guaranteed way. The chapter shows three examples from process engineering.

Keywords Dynamic optimization • Verified simulation • Interval contractors • Overestimation reduction

C. Pérez-Galván • I.D.L. Bogle (✉)
Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK
e-mail: carlos.galvan.12@ucl.ac.uk; d.bogle@ucl.ac.uk

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_10

Introduction

Dynamic optimization arises in many practical applications. Optimization problems can be formulated for dynamic processes depending on the application, e.g., the parameter estimation problem for model development can be addressed if we formulate an optimization problem in order to minimize the weighted squared errors between the observed values and those predicted by the model. Another example is the optimal control problem for which we wish to obtain the best paths of the control variables in a given trajectory that minimize a certain function (resources consumption, total time, and cost).

Direct and indirect methods are available for solving these problems. On one hand, indirect methods involve solving the necessary conditions of optimality which often results in a two-point boundary-value problem that is very difficult to solve. On the other hand, direct methods are far more widely used mainly because they take advantage of modern nonlinear programming (NLP) techniques. Regarding direct methods there are two main approaches relying on discretization: the simultaneous and sequential approach.

In the simultaneous approach (or orthogonal collocation or complete collocation or full discretization) both the state and control variables are discretized yielding a large-scale NLP problem [3] that does not require inner loops of ordinary differential equations (ODEs) solvers. This kind of approach is an infeasible path method, i.e., the problem is only feasible at the converged solution of the NLP. Large-scale problems can be solved as it saves computational cost avoiding the ODE integrations.

The sequential approach (also known as control vector parametrization or single-shooting) requires the discretization of the control variables. This approach is a feasible path method which means that the ODEs are satisfied at each iteration of the NLP algorithm hence path constraints can be incorporated in this approach. The method takes as inputs the control parameters and initial conditions to solve the ODE model while the control variables are updated using the NLP solver. They can solve relatively small scale problems because they require multiple computationally demanding numerical integrations of the ODE model and the gradients of the objective function.

This work starts by giving a review of dynamic optimization in chemical engineering (section "Global Optimization of Dynamic Process Systems"). Specifically, works where a deterministic approach is used to solve the problem in a guaranteed way by means of verified ODE solvers. In section "Enclosing the Solutions of IVPs for ODEs" various bounding techniques that have recently been contributed by several research groups are described. Section "ITS Method with Overestimation Reduction" describes the approach developed by our research group that uses an interval Taylor series with interval contractors to address the dynamic optimization problem. Section "Global Optimization Algorithm Using ITS with Overestimation Reduction" incorporates the verified method proposed in a branch and bound

method for global optimization. Results of the use of the branch and bound algorithm with the proposed verified ODE solver are presented in section "Results". Finally, conclusions and research directions are provided in section "Conclusions".

Global Optimization of Dynamic Process Systems

In chemical engineering dynamic processes arise in many applications and often an optimal trajectory is sought. For example, we need the control path for optimal resource consumption in changes of product grade on polymer plants. Some other examples of applications are: state estimation for process control as well as in model building applications [5, 37], design of distributed systems in chemical engineering, including reactors and packed column separators [2], trajectory optimization in chemical processes for transitions between operating conditions and to handle load changes [7], off-line and online problems in process control, particularly for multivariable systems that are nonlinear and output constrained [10, 28], and optimum batch process operating profiles, particularly for reactors and separators [24].

There are many problems that require guaranteed bounded performance along the whole trajectory mainly because of safety critical and environmental limits. In these applications, it is not admissible to allow a certain variable to go beyond some prescribed limits. For example, the content of a certain compound in a stream is not allowed to be present in a higher concentration than the level permitted by the environmental regulator who might otherwise oblige the plant to shut down. The engineer has to make sure that this concentration is within the admissible limits at all times, however, he/she must do it without compromising too much the cost of the plant operation and the qualities of all products. Safety critical plants require that the variables of interest, for example, temperature or pressure of plant equipment, be within prescribed limits for the whole time of operation.

Obtaining the optimal performance is not an easy task since dynamic models in chemical engineering exhibit non-convexities due to the combination of nonlinear terms, and thus, multiple local minima arise in the model. Deterministic global optimization techniques are developed and used. Stochastic methods are also used although they are not able to provide a guarantee that the global optima have been found. Hybrid approaches for global optimization have also been developed. In [47] and [46] a method that uses balanced random interval arithmetic is described and used in optimization examples, respectively. In this technique a trade-off between efficiency and robustness is proposed; however, the guarantee of finding the global optimum is sacrificed.

Moreover, to guarantee that we are within limits we are required to rigorously make sure we are including all possible solutions taking into account approximation and truncation errors in the numerical computations. Computations can be made to rigorously determine the optimal solution and are able to guarantee the best bounded performance.

This can be achieved if we obtain bounds on the dynamic variables that are mathematically verified to be within a safe operating range. A direct sequential approach with a verified method can be used to obtain such bounds. In this approach a verified ODE solver is used to provide bounds in which all the possible solutions are guaranteed to be contained. However, an important challenge in this field is the overestimation generated in the integration process that causes the bounds to be useless because they are very conservative or in the worst case they tend to $\pm\infty$ and the simulation has to be stopped. The tightness of these bounds directly affects the global optimization algorithm. Žilinskas and Žilinskas [48] proved experimentally for relatively small problems how using exact bounds results in approximately twice as efficient than using interval arithmetic inclusions.

Several chemical engineering researchers have devoted their efforts to solve the guaranteed global optimization problem for dynamic systems and most of these research works implement an extensive search NLP such as a branch and bound framework together with a guaranteed ODE solver within a sequential approach. A branch and bound framework was used with the αBB method [1] in a sequential approach and applied to four different optimal control problems including the optimal control of batch and semi-batch processes [5], and to parameter estimation problems to determine reaction kinetic constants from time series data [6]. In principle the method of Esposito and Floudas [5, 6] provides a guaranteed global optimum. The rigorous underestimators needed are hard to obtain and here only sampling approaches were proposed. Another sequential approach implementing a branch and bound framework was developed with a convex underestimating procedure [25] which they applied to parameter estimation, chemical kinetics and modelling, and optimal control problems. Some years later, again using a sequential approach and a branch and bound framework, Papamichail and Adjiman [26] used McCormick relaxations and constant and affine bounds in the solution of parameter estimation and optimal control problems. The approach used by Papamichail and Adjiman [25, 26] is computationally expensive in constructing tight affine underestimators and overestimators. Singer and Barton [36] presented a sequential approach using another branch and bound framework for problems with an integral objective function. This algorithm implements McCormick relaxations to construct the convex relaxations and the method was applied to parameter estimation and optimal control problems. The dynamic optimization problem has also been addressed using Taylor Models in a sequential approach and a branch and bound framework with focus on the tightness of the ODEs state bounds. This method uses Taylor Models method with an interval remainder term and was applied to several parameter estimation problems [14]. Later, they applied the same method but this time with a branch and reduce approach using a domain reduction technique [15] and the applications were an optimal control and a final time formulation of the oil shale pyrolysis problems. The later method was extended to account for inequality path constraints in a rigorous way [43] and was applied to three semi-batch reactor problems.

Only a few research groups have worked on this problem and they have mainly made use of a branch and bound framework in a sequential framework. The available methods for solving optimization of dynamic systems to global optimality are only

able to address low dimensional problems (of the order of 4–6 state variables). In these methods one of the main challenges to overcome is on the construction of effective and efficient bounds for the ODE system since tight and computationally cheap bounds are needed. This problem arises because of the overestimation produced while constructing the bounds due to the dependency property in interval arithmetic as well as the wrapping effect.

Enclosing the Solutions of IVPs for ODEs

The mathematical form of the problem that this section considers is as follows:

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \theta), \mathbf{y}(t_0, \theta) = \mathbf{y}_0(\theta), \mathbf{y}_0 \in \mathbf{Y}_0, \theta \in \Theta \tag{1}$$

where $t \in [t_0, t_f]$, θ represent the time-invariant parameters with $\Theta = [\underline{\theta}, \overline{\theta}]$, $\mathbf{y}$ represent the vector of state variables, and $\mathbf{y}_0$ are the initial conditions at time t_0 with $\mathbf{Y_0} = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$. Boldface characters are used to denote vector valued quantities unless otherwise noted.

In this work the functions $mid(X)$ and $w(X)$ represent the midpoint and the wideness of the interval or interval vector, respectively. In the case of the interval vector these functions are obtained componentwise. They are given by

$$w(X) = \overline{X} - \underline{X}$$

$$mid(X) = \frac{\overline{X} + \underline{X}}{2}$$

where $\underline{X}$ and $\overline{X}$ are the lower and upper endpoints, respectively.

Finding bounds that enclose the state variables of the dynamic optimization in a reliable way is a difficult task. The solution tools are often based on methods that rely on some implementations of interval analysis that are subject to overestimation specially when it is desired to propagate these bounds introducing uncertainty in the system parameters and initial conditions. The overestimation in these methods is produced mainly because of the so-called dependency problem and wrapping effect.

Dependency Problem

The dependency problem arises in interval analysis because the method does not account for the dependency of multiple repetitions of the variables in a mathematical model. In order to manage this problem ways of reducing to a minimum the number of repetitions of the same variable are sought. The next example illustrates the dependency problem.

Example 1. Consider the following function: $f(x) = xx$ and the ranges of $X = [-1, 1]$. The natural interval extension of f using X yields

$$f(X) = XX = [-1, 1][-1, 1] = [-1, 1]$$

However, if we rewrite the function as $f(x) = x^2$ and perform the natural interval extension again the result is different

$$f(X) = X^2 = [-1, 1]^2 = [0, 1]$$

The first time the function was evaluated the true range of the function was overestimated but in the second case the exact range was obtained.

Wrapping Effect

The wrapping effect is caused by the introduction of excess points, points that do not belong to the solution set, into the wrapping of a set. In most interval arithmetic algorithms wrapping operations enclose those excess points. These points collectively are called wrapping excess and contribute to the wrapping effect.

In the next subsections different existing approaches to enclose the solutions of initial value problems (IVPs) for ODEs are described and their main properties are discussed.

Interval Taylor Series

The first method for constructing bounds for ODEs using interval analysis was devised by Moore [21]. His method consists of two stages:

1. In the first stage the existence and uniqueness of the solution is validated by means of the Banach fixed-point theorem and the Picard–Lindelof operator. A Taylor series enclosure is used in the Moore algorithm to do this.
2. In the second stage a refinement of the solution is obtained using a high order Taylor series method.

The main challenge in methods based on interval arithmetic and high order Taylor series is on the reduction of the overestimation caused by the wrapping effect.

The Interval Taylor Series (ITS) method has been used in its traditional form with an interval remainder term. However, convex and concave bounds have been obtained using the McCormick relaxation rules where the underlying interval bounds are computed using an ITS method. In the next two sections "Interval Remainder Term" and "McCormick Relaxation Technique", the ITS method and the method with McCormick relaxations are presented, respectively.

Interval Remainder Term

The interval Taylor series method has been traditionally used with an interval remainder term. The algorithms of [4, 12, 21], and [17] are an example of this. It is the simplest way to evaluate the remainder term as only upper and lower bounds that include the solution are needed. The first and second stages of a general ITS method that includes parameter dependence are described by the following equations.

A Priori Enclosure

$$\tilde{\mathbf{Y}}_j = \mathbf{Y}_j + \sum_{i=1}^{k-1} [0,h_j]^i \mathbf{f}^{[i]}(\mathbf{Y}_j,\Theta) + [0,h_j]^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j^0,\Theta) \subseteq \tilde{\mathbf{Y}}_j^0 \tag{2}$$

Computing a Tighter Enclosure

$$\mathbf{u}_{j+1} = \hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j,\hat{\theta}) \tag{3}$$

$$\mathbf{S}_{j+1}^{\mathbf{y}} = \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(\mathbf{Y}_j,\Theta) \tag{4}$$

$$\mathbf{S}_{j+1}^{\theta} = \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \theta}(\mathbf{Y}_j,\Theta) \tag{5}$$

$$\mathbf{Z}_{j+1} = h_j^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j,\Theta) \tag{6}$$

$$\mathbf{Y}_{j+1} = \mathbf{u}_{j+1} + \mathbf{S}_{j+1}^{\mathbf{y}}(\mathbf{Y}_j - \hat{\mathbf{y}}_j) + \mathbf{S}_{j+1}^{\theta}(\Theta - \hat{\theta}) + \mathbf{Z}_{j+1} \tag{7}$$

The product $\mathbf{S}_{j+1}^{\mathbf{y}}(\mathbf{Y}_j - \hat{\mathbf{y}}_j)$ in (7) is rearranged in the following way to reduce the wrapping effect

$$\Gamma_{j+1} = \mathbf{A}_{j+1}^{-1}(\mathbf{Z}_{j+1} - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}_{j+1}^{-1}(\mathbf{S}_{j+1}^{\mathbf{y}}\mathbf{A}_j)\Gamma_j + (\mathbf{A}_{j+1}^{-1}\mathbf{S}_{j+1}^{\theta})(\Theta - \hat{\theta}) \tag{8}$$

$$\mathbf{Y}_{j+1} = \mathbf{u}_{j+1} + (\mathbf{S}_{j+1}^{\mathbf{y}}\mathbf{A}_j)\Gamma_j + \mathbf{S}_{j+1}^{\theta}(\Theta - \hat{\theta}) + \mathbf{Z}_{j+1} \tag{9}$$

where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{y}}_j$ is the midpoint of $\mathbf{Y}_j$. Here, the Taylor coefficients $\mathbf{f}^{[i]}$ are computed using automatic differentiation, $\Gamma_0 = \mathbf{Y}_0 - \hat{\mathbf{y}}_0$, $\mathbf{A}_0 = \mathbf{I}$, and $\mathbf{A}_{j+1}$ is chosen as the $\mathbf{Q}_{j+1}$ matrix of the QR factorization ($\mathbf{Q}_{j+1}\mathbf{R}_{j+1}$) of the midpoint of $\mathbf{S}_{j+1}^{y}\mathbf{A}_j$, the parallelepiped enclosure of Γ_{j+1}. Here, $\mathbf{R}_{j+1}$ represents the upper triangular matrix.

McCormick Relaxation Technique

The interval Taylor series method has also been considered using the McCormick relaxation rules. The idea here is to use McCormick relaxations in all the elementary

operations of a factorable function in a recursive way so as to construct convex and concave relaxations of the original ODE system, that are useful in global optimization. The method assumes that interval bounds are available. Therefore, these convex and concave bounds are guaranteed to be at least as tight as the interval bounds [32]. The use of McCormick bounds typically increases the computational time by a factor of 2 when compared to interval bounds. The first and second stages are illustrated by (10) and (12). In these equations the superindexes *cv* and *cc* are used to denote the convex underestimator and concave overestimator (convex/concave relaxations), respectively. The $\odot$ operator is used here to denote that the operation yields concave and convex bounds.

A Priori Enclosure

$$\begin{aligned}\tilde{\mathbf{y}}_j^{cv}, \tilde{\mathbf{y}}_j^{cc} t(\theta) = &\sum_{i=0}^{k-1} [0,h_j]^i \odot [\mathbf{f}^{[i],cv}, \mathbf{f}^{[i],cc}](\mathbf{y}_j^{cv}(\theta), \mathbf{y}_j^{cc}(\theta), \theta) \\ &+ [0,h_j]^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j^0, \Theta)\end{aligned} \tag{10}$$

Computing a Tighter Enclosure

$$\begin{aligned}[\mu_j^{cv}, \mu_j^{cc}](\theta) &= \tilde{\mathbf{y}}_j + [0,1] \odot ([\mathbf{y}_j^{cv}, \mathbf{y}_j^{cc}](\theta) - \tilde{\mathbf{y}}_j) \\ [\rho^{cv}, \rho^{cc}](\theta) &= \hat{\theta} + [0,1] \odot (\theta - \hat{\theta}) \\ [\mathbf{S}_{j+1}^{\mathbf{y},cv}, \mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta) &= \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],cv}}{\partial \mathbf{y}}, \frac{\partial \mathbf{f}^{[i],cc}}{\partial \mathbf{y}} \right] ([\mu_j^{cv}, \mu_j^{cc}](\theta), [\rho^{cv}, \rho^{cc}](\theta)) \\ [\mathbf{S}_{j+1}^{\theta,cv}, \mathbf{S}_{j+1}^{\theta,cc}](\theta) &= \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],cv}}{\partial \theta}, \frac{\partial \mathbf{f}^{[i],cc}}{\partial \theta} \right] ([\mu_j^{cv}, \mu_j^{cc}](\theta), [\rho^{cv}, \rho^{cc}](\theta)) \\ [\mathbf{Z}_{j+1}^{cv}, \mathbf{Z}_{j+1}^{cc}](\theta) &= h_j^k [\mathbf{f}^{[k],cv}, \mathbf{f}^{[k],cc}](\tilde{\mathbf{y}}_j^{cv}(\theta), \tilde{\mathbf{y}}_j^{cc}(\theta), \theta) \\ [\mathbf{y}_{j+1}^{cv}, \mathbf{y}_{j+1}^{cc}](\theta) &= \mathbf{u}_{j+1} + [\mathbf{S}_{j+1}^{\mathbf{y},cv}, \mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta) \odot ([\mathbf{y}_j^{cv}, \mathbf{y}_j^{cc}](\theta) - \hat{\mathbf{y}}_j) \\ &\quad + [\mathbf{S}_{j+1}^{\theta,cv}, \mathbf{S}_{j+1}^{\theta,cc}](\theta) \odot (\theta - \hat{\theta}) + [\mathbf{Z}_{j+1}^{cv}, \mathbf{Z}_{j+1}^{cc}](\theta)\end{aligned} \tag{11}$$

The term $[\mathbf{S}_{j+1}^{\mathbf{y},cv}, \mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta) \odot ([\mathbf{y}_j^{cv}, \mathbf{y}_j^{cc}](\theta) - \hat{\mathbf{y}}_j)$ is rearranged since it is an important contributor to the wrapping effect

$$\begin{aligned}[\Gamma_{j+1}^{cv}, \Gamma_{j+1}^{cc}](\theta) = &\mathbf{A}_{j+1}^{-1} \odot ([\mathbf{Z}_{j+1}^{cv}, \mathbf{Z}_{j+1}^{cc}](\theta) - mid(\mathbf{Z}_{j+1})) \\ &+ [\mathbf{A}_{j+1}^{-1} \odot ([\mathbf{S}_{j+1}^{\mathbf{y},cv}, \mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta) \odot \mathbf{A}_j)] \odot [\Gamma_j^{cv}, \Gamma_j^{cc}](\theta) \\ &+ \mathbf{A}_{j+1}^{-1} \odot ([\mathbf{S}_{j+1}^{\mathbf{y},cv}, \mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta)) \odot (\theta - \hat{\theta})\end{aligned}$$

$$\begin{aligned}\mathbf{y}_{j+1}^{cv},\mathbf{y}_{j+1}^{cc}t(\theta) = \mathbf{u}_{j+1} &+ [\mathbf{Z}_{j+1}^{cv},\mathbf{Z}_{j+1}^{cc}](\theta) \\ &+ ([\mathbf{S}_{j+1}^{\mathbf{y},cv},\mathbf{S}_{j+1}^{\mathbf{y},cc}](\theta)\odot\mathbf{A}_j)\odot[\Gamma_j^{cv},\Gamma_j^{cc}](\theta) \\ &+ [\mathbf{S}_{j+1}^{\theta,cv},\mathbf{S}_{j+1}^{\theta,cc}](\theta)\odot(\theta-\hat{\theta})\end{aligned} \tag{12}$$

The matrix $\mathbf{A}_{j+1}$ is chosen again as the $\mathbf{Q}_{j+1}$ matrix of the QR factorization of the midpoint of $\mathbf{S}_{j+1}^{y}\mathbf{A}_j$.

Interval Hermite–Obreschkoff

The software VNODE-LP [23], which is one of the best known software packages for this purpose, makes an implementation of the interval Taylor series method with the Hermite–Obreschkoff scheme (IHO). It consists of two stages as the interval Taylor series method; in the first stage, the validation of existence and uniqueness is carried out and a suitable step size and a priori enclosure are chosen and in the second stage an Hermite–Obreschkoff scheme which uses a predictor and corrector steps is used. Usually, this package represents a benchmark for other packages for the same purpose.

A Priori Enclosure
This stage is computed as in the ITS method with (2).

Computing a Tighter Enclosure

Predictor The second phase consists in a predictor and a corrector steps where a representation of the solution $\mathbf{Y}_{j+1}^{*}$ is obtained and then refined as $\mathbf{Y}_{j+1}$, respectively.

$$\hat{\mathbf{u}}_{j+1} = \mathbf{u}_j + \sum_{i=1}^{k} h_j^i \mathbf{f}^{[i]}(\mathbf{u}_j) \tag{13}$$

$$\mathbf{z}_{j+1} = h_j^{k+1}\mathbf{f}^{[k+1]}(\tilde{\mathbf{Y}}_j) \tag{14}$$

$$\mathbf{U}_{j+1} = \mathbf{I} + \sum_{i=1}^{k} \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(\mathbf{Y}_j) \tag{15}$$

$$\mathbf{X}_{j+1} = (\mathbf{U}_{j+1}\hat{\mathbf{S}}_j)\alpha + \{(\mathbf{U}_{j+1}\mathbf{A}_j)\Omega_j \cap (\mathbf{U}_{j+1}\mathbf{Q}_j)\Omega_{QR,j}\} \tag{16}$$

with $\alpha = \mathbf{Y}_0 - \mathbf{u}_0$. $\Omega_j, \hat{\mathbf{A}}_j$ and $\Omega_{QR,j}, \hat{\mathbf{Q}}_j$ correspond to the modifying terms of the parallelepiped and QR factorization methods [17], respectively.

A predicting enclosure $\mathbf{Y}_{j+1}^{*}$ is obtained with (17)

$$\mathbf{Y}_{j+1}^{*} = (\hat{u}_{j+1} + z_{j+1} + \mathbf{X}_{j+1}) \cap \tilde{\mathbf{Y}}_j \tag{17}$$

Corrector The corrector step uses the solution obtained in the predictor step $\mathbf{Y}^*_{j+1}$ and refines it.

$$\hat{\mathbf{y}}^*_{j+1} = mid(\mathbf{Y}^*_{j+1}) \tag{18}$$

$$c_i^{k,p} = \frac{k!(k+p-i)!}{(p+k)!(k-i)!} \quad k,p,i \geq 0 \tag{19}$$

$$\mathbf{B}_{j+1} = \sum_{i=0}^{k} (-1)^i c_i^{k,p} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(\mathbf{Y}^*_{j+1}) \tag{20}$$

$$\mathbf{F}_j = \sum_{i=0}^{p} c_i^{k,p} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(\mathbf{Y}_j) \tag{21}$$

$$\hat{\mathbf{b}}_{j+1} = mid(\mathbf{B}_{j+1}) \tag{22}$$

$$\mathbf{S}_{j+1} = (\mathbf{b}_{j+1}^{-1}\mathbf{F}_j)\hat{\mathbf{S}}_j \tag{23}$$

$$\mathbf{A}_{j+1} = (\hat{\mathbf{b}}_{j+1}^{-1}\mathbf{F}_j)\hat{\mathbf{A}}_j \tag{24}$$

$$\mathbf{Q}_{j+1} = (\hat{\mathbf{b}}_{j+1}^{-1}\mathbf{F}_j)\hat{\mathbf{Q}}_j \tag{25}$$

$$\gamma_{p,k} = \frac{k!p!}{(p+k)!} \tag{26}$$

$$\mathbf{E}_{j+1} = (-1)^k \gamma_{p,k} h_j^{p+k+1} \mathbf{f}^{[p+k+1]}(\tilde{\mathbf{Y}}_j) \tag{27}$$

$$\hat{\mathbf{g}}_{j+1} = \sum_{i=0}^{k} c_i^{p,k} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{u}}_j) - \sum_{i=0}^{p} (-1)^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}^*_{j+1}) \tag{28}$$

$$\mathbf{D}_{j+1} = \hat{\mathbf{g}}_{j+1} + \mathbf{E}_{j+1} \tag{29}$$

$$\mathbf{W}_{j+1} = \hat{\mathbf{b}}_{j+1}^{-1}\mathbf{D}_{j+1} + (\mathbf{I} - \hat{\mathbf{b}}_{j+1}^{-1}\mathbf{B}_{j+1})(\mathbf{Y}^*_{j+1} - \hat{\mathbf{y}}^*_{j+1}) \tag{30}$$

$$\mathbf{L}_{j+1} = (\mathbf{A}_{j+1}\Omega_j) \cap (\mathbf{Q}_{j+1}\Omega_{QR,j}) \tag{31}$$

The tight enclosure obtained with the corrector step is represented by (32).

$$\mathbf{Y}_{j+1} = (\hat{\mathbf{y}}^*_{j+1} + \mathbf{S}_{j+1}\alpha + \mathbf{L}_{j+1} + \mathbf{W}_{j+1}) \cap \mathbf{Y}^*_{j+1} \tag{32}$$

For the next time step a series of new values are needed for

$$\hat{\mathbf{u}}_{j+1} = mid(\mathbf{Y}_{j+1})$$

$$\hat{\mathbf{S}}_{j+1} = mid(\mathbf{S}_{j+1})$$

$$\mathbf{V}_{j+1} = \hat{\mathbf{y}}^*_{j+1} - \hat{\mathbf{u}} + (\mathbf{S}_{j+1} - \hat{\mathbf{S}}_{j+1})\alpha + \mathbf{W}_{j+1}$$

$$\hat{\mathbf{A}}_{j+1} = mid(\mathbf{A}_{j+1})$$

$$\Omega_{j+1} = (\hat{\mathbf{A}}_{j+1}^{-1}\mathbf{A}_{j+1})\Omega_j + \hat{\mathbf{A}}_{j+1}^{-1}\mathbf{V}_{j+1}$$

$\hat{\mathbf{Q}}_{j+1}$ is chosen as the orthogonal matrix in the QR factorization as in the previous methods.

$$\Omega_{QR,j+1} = (\hat{\mathbf{Q}}_{j+1}^{-1}\mathbf{Q}_{j+1})\Omega_{QR,j} + \hat{\mathbf{Q}}_{j+1}^{-1}\mathbf{V}_{j+1}$$

Taylor Models

The Taylor Models (TM) method is based on Remainder Differential Algebra (RDA) operations [19] which are used in the Taylor Model arithmetic. The Taylor Models method is currently one of the best ways to deal with the dependency problem in verified solution of ODEs mainly because Taylor Models of functions represent Taylor polynomials in symbolic form with an interval remainder term; however, their computation results in high computational cost. Taylor Models have been studied using different techniques such as the traditional interval remainder (section "Interval Remainder Term"), McCormick relaxations (section "McCormick Relaxation Technique"), and ellipsoidal calculus (section "Ellipsoidal Remainder Term"). A Taylor Model of a function is represented by a qth order truncated Taylor series polynomial $\mathscr{P}_f$ and a remainder term $\mathscr{R}_f$.

$$\mathscr{T}_f = (\mathscr{P}_f, \mathscr{R}_f) \tag{33}$$

and $B(\mathscr{P}_f)$ is the interval bound of the polynomial. Bounds on the Taylor Model can be computed by $B(\mathscr{T}_f) = B(\mathscr{P}_f) + \mathscr{R}_f$.

Interval Remainder Term

Similar to the ITS method with an interval remainder term the easiest way to construct bounds in Taylor Models is by considering its remainder term in the form of an interval. This approach is used in VSPODE [16], and COSY [20].

A Priori Enclosure
An a priori enclosure can be computed following a high order enclosure approach either by (2) [16] or by Taylor Models (34) [33].

$$\mathscr{T}_{\tilde{\mathbf{y}}_j}(\theta) = \sum_{i=1}^{k-1} [0,h_j]^i \mathscr{T}_{\mathbf{f}^{[i]}}(\mathscr{T}_{\mathbf{y}_j}(\theta),\theta) + [0,h_j]^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j^0, \Theta) \tag{34}$$

However, the use of (2) over (34) does not usually have a big impact in the two-stage method.

Computing a Tighter Enclosure According to [16]

$$\hat{\mathscr{P}}_{\mathbf{U}_{j+1}} = \hat{\mathscr{P}}_{\mathbf{y}_j} + \sum_{i=1}^{k-1} h_j^i \mathscr{T}_{\hat{\mathbf{f}}^{[i]}} + h_j^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j^0, \Theta) \tag{35}$$

$$\hat{\mathscr{R}}_{\mathbf{y}_j} = \mathscr{T}_{\mathbf{y}_j} - \hat{\mathscr{P}}_{\mathbf{y}_j} \tag{36}$$

$$\mathscr{T}_{\mathbf{y}_{j+1}} = \hat{\mathscr{P}}_{\mathbf{U}_{j+1}} + \hat{\mathscr{R}}_{\mathbf{U}_{j+1}} + \mathbf{S}_{j+1}^{y} \hat{\mathscr{R}}_{\mathbf{y}_j} \tag{37}$$

where $mid(\hat{\mathscr{R}}_{\mathbf{U}_{j+1}}) = 0$ and $\hat{\mathscr{P}}_{\mathbf{U}_{j+1}}$ is the polynomial part with the constant term conveniently modified, the centered polynomial.
Rearranging the product $\mathbf{S}_{j+1}^{\mathbf{y}} \hat{\mathscr{R}}_{\mathbf{y}_j}$.

$$\mathbf{V}_{j+1} = (\mathbf{A}_{j+1}^{-1} \mathbf{S}_{j+1}^{\mathbf{y}} \mathbf{A}_j)\mathbf{V}_j + \mathbf{A}_{j+1}^{-1} \hat{\mathscr{R}}_{\mathbf{U}_{j+1}}$$

$$\mathscr{T}_{\mathbf{y}_{j+1}} = \hat{\mathscr{P}}_{\mathbf{U}_{j+1}} + \{\mathbf{A}_{j+1}\mathbf{v}_{j+1} | \mathbf{v}_{j+1} \in \mathbf{V}_{j+1}\} \tag{38}$$

$$\mathbf{Y}_{j+1} = B(\hat{\mathscr{P}}_{\mathbf{U}_{j+1}}) + \mathbf{A}_{j+1}\mathbf{V}_{j+1} \tag{39}$$

where $\mathbf{V}_0 = 0$, $\mathbf{A}_0 = \mathbf{I}$, $\mathbf{A}_{j+1} = \mathbf{Q}_j$, and $(\mathbf{Q}_j, \mathbf{R}_j) = mid(\mathbf{S}_{j+1}^{\mathbf{y}} \mathbf{A}_j)$, the QR factorization.

Computing a Tighter Enclosure According to [33]

$$\mathscr{T}_{\mathbf{V}_{j+1}}(\theta) = \sum_{i=0}^{k-1} h_j^i \mathscr{T}_{\mathbf{f}^{[i]}}(\hat{\mathscr{P}}_{\mathbf{y}_j}(\theta), \theta) \tag{40}$$

$$\mathscr{T}_{\mathbf{R}_{j+1}}(\theta) = h_j^k \mathscr{T}_{\mathbf{f}^{[k]}}(\mathscr{T}_{\tilde{\mathbf{y}}_j}(\theta), \theta) \tag{41}$$

$$\mathscr{T}_{\mathbf{J}_{j+1}}(\theta) = \sum_{i=0}^{k-1} h_j^i \mathscr{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}}(\mathscr{T}_{\mathbf{y}_j}(\theta), \theta) \tag{42}$$

$$\mathscr{T}_{\mathbf{y}_{j+1}}(\theta) = \mathscr{T}_{\mathbf{V}_{j+1}}(\theta) + \mathscr{T}_{\mathbf{J}_{j+1}}(\theta)\hat{\mathscr{R}}_{\mathbf{y}_j} + \mathscr{T}_{\mathbf{R}_{j+1}}(\theta) \tag{43}$$

As in the previous cases the rearrangement of the term $\mathscr{T}_{\mathbf{J}_{j+1}}(\theta)\hat{\mathscr{R}}_{\mathbf{y}_j}$ is carried out in order to avoid the wrapping effect.

$$\begin{aligned} \mathscr{T}_{\Delta_{j+1}}(\theta) = &\left\{ \mathbf{A}_{j+1}^{-1} \left[\mathscr{T}_{\mathbf{J}_{j+1}}(\theta)\mathbf{A}_j \right] \right\} \mathscr{T}_{\Delta_j}(\theta) \\ &+ \mathbf{A}_{j+1}^{-1} \left[\mathscr{T}_{\mathbf{R}_{j+1}}(\theta) + \mathscr{T}_{\mathbf{V}_{j+1}}(\theta) - \hat{\mathscr{P}}_{\mathbf{x}_{j+1}}(\theta) \right] \end{aligned} \tag{44}$$

$$\mathscr{T}_{\mathbf{y}_{j+1}}(\theta) = \mathscr{T}_{\mathbf{V}_{j+1}}(\theta) + \mathscr{T}_{\mathbf{R}_{j+1}}(\theta) + \left[\mathscr{T}_{\mathbf{J}_{j+1}}(\theta)\mathbf{A}_j \right] \mathscr{T}_{\Delta_j}(\theta) \tag{45}$$

The matrix $\mathbf{A}_{j+1}$ is chosen once more as the midpoint of the product $\mathbf{S}_{j+1}^{\mathbf{y}} \mathbf{A}_j$.

McCormick Relaxation Technique

McCormick relaxations have been used with Taylor Models in an approach called McCormick–Taylor Model. In this approach the convex/concave bounds are computed alongside the interval bounds [33].
A McCormick–Taylor Model is defined as

$$\mathscr{M}\mathscr{T}_{\mathbf{f}}(\theta) = \mathscr{P}_{\mathbf{f}}(\theta) + \mathscr{M}_{\mathbf{R_f}}(\theta) \tag{46}$$

$$\mathscr{M}_{\mathbf{R_f}}(\theta) = \{[r_{\mathbf{f}}^{L}, r_{\mathbf{f}}^{U}], [r_{\mathbf{f}}^{cv}, r_{\mathbf{f}}^{cc}](\theta)\} \tag{47}$$

A Priori Enclosure

$$\mathscr{M}_{\tilde{\mathbf{y}}_j}(\theta) = \sum_{i=1}^{k-1} [0,h_j]^i \mathscr{M}_{\mathbf{f}^{[i]}}(\mathscr{M}_{\mathbf{y}_j}(\theta),\theta) + [0,h_j]^k \mathbf{f}^{[k]}(\tilde{\mathbf{Y}}_j^0, \Theta) \tag{48}$$

Computing a Tighter Enclosure

$$\mathscr{M}\mathscr{T}_{\mathbf{V}_{j+1}}(\theta) = \sum_{i=0}^{k-1} h_j^i \mathscr{M}\mathscr{T}_{\mathbf{f}^{[i]}}(\hat{\mathscr{P}}_{\mathbf{y}_j}(\theta),\theta)$$

$$\mathscr{M}_{\mathbf{J}_{j+1}}(\theta) = \sum_{i=0}^{k-1} h_j^i \mathscr{M}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}}(\mathscr{M}_{\mathbf{y}_j}(\theta),\theta)$$

$$\mathscr{M}_{\mathbf{R}_{j+1}}(\theta) = h_j^k \mathscr{M}_{\mathbf{f}^{[k]}}(\mathscr{M}_{\tilde{\mathbf{y}}_j}(\theta),\theta)$$

$$\mathscr{M}\mathscr{T}_{\mathbf{y}_{j+1}}(\theta) = \mathscr{M}\mathscr{T}_{\mathbf{V}_{j+1}}(\theta) + \mathscr{M}_{\mathbf{J}_{j+1}}(\theta) \times \hat{\mathscr{M}}_{\mathbf{R}_{\mathbf{y}_j}} + \mathscr{M}_{\mathbf{R}_{j+1}}(\theta) \tag{49}$$

The wrapping effect needs to be mitigated as in the previous cases by modifying the term $\mathscr{M}_{\mathbf{J}_{j+1}}(\theta) \times \hat{\mathscr{M}}_{R_{\mathbf{y}_j}}$

$$\begin{aligned}\mathscr{M}_{\Delta_{j+1}}(\theta) = &\left\{\mathbf{A}_{j+1}^{-1}\left[\mathscr{M}_{\mathbf{J}_{j+1}}(\theta)\mathbf{A}_j\right]\right\}\mathscr{M}_{\Delta_j}(\theta) \\ &+ \mathbf{A}_{j+1}^{-1}\left[\mathscr{M}_{\mathbf{R}_{j+1}}(\theta) + \mathscr{M}_{\mathbf{V}_{j+1}}(\theta) - \hat{\mathscr{P}}_{\mathbf{x}_{j+1}}(\theta)\right]\end{aligned} \tag{50}$$

$$\mathscr{M}\mathscr{T}_{\mathbf{y}_{j+1}}(\theta) = \mathscr{M}\mathscr{T}_{\mathbf{V}_{j+1}}(\theta) + \left[\mathscr{M}_{\mathbf{J}_{j+1}}(\theta)\mathbf{A}_j\right]\mathscr{M}_{\Delta_j}(\theta) + \mathscr{M}_{\mathbf{R}_{j+1}}(\theta) \tag{51}$$

where $\mathscr{M}_{\Delta_0}(\theta) = \hat{\mathscr{M}}_{\mathbf{R}_{\mathbf{y}_0}}(\theta)$. The matrix $\mathbf{A}_{j+1}^{-1}$ is obtained by computing the QR factorization of the midpoint of $\mathbf{S}_{j+1}^{\mathbf{y}}\mathbf{A}_j$.

Ellipsoidal Remainder Term

On top of that ellipsoidal remainder bounds have been used in Taylor Models. This new approach can guarantee the propagation of stable bounds for asymptotically stable systems and with a given level of uncertainty [40].

$$\mathscr{E}\mathscr{T}_f = \mathscr{P}_f \oplus \mathscr{E}(Q_f) \tag{52}$$

Here, $\oplus$ stands for the Minkowski sum of two sets, $\mathscr{E}(Q_f)$ is an ellipsoid centered at the origin and is defined as $\{Q_f^{\frac{1}{2}} v | v \in \mathbb{R}^n : v^T v \leq 1\}$ where Q_f is an n-dimensional positive semi-definite symmetric shape matrix.

The way in which this method proceeds is different to the previous ones in that a predicting enclosure is computed first and then a step size for which the predictor gives a guaranteed enclosure is computed by solving an optimization problem [8].

In order to compute a Taylor Model with ellipsoidal remainder for the function: $\mathbf{f}^{[i]}$ (Taylor coefficients), the following steps are needed.

Obtain the Taylor Model with ellipsoidal remainder of $\mathbf{y}_j$ as

$$\mathscr{E}\mathscr{T}_{\mathbf{y}_j}(\theta) = \mathscr{P}_{\mathbf{y}_j}(\theta) \oplus \mathscr{E}(Q_{\mathbf{y}_j}) \tag{53}$$

The Taylor Model of $\mathbf{f}^{[i]}$ can be obtained with the polynomial part of (53)

$$\mathscr{T}_{\mathbf{f}^{[i]}}(\mathscr{P}_{\mathbf{y}_j}(\theta), \theta) = \mathscr{P}_{\mathbf{f}^{[i]}}(\theta) \oplus \mathbf{R}_{\mathbf{f}^{[i]}} \tag{54}$$

Obtain the interval hull of $\mathscr{E}(Q_{\mathbf{y}_j})$

$$\mathbf{R}_{\mathbf{y}_j} = B(\mathscr{E}(Q_{\mathbf{y}_j}))$$

Compute the derivative functions $\mathbf{G}_1$ and $\mathbf{G}_2$

$$\mathbf{G}_1 = \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(B(\mathscr{P}_{\mathbf{y}_j}(\theta))) \tag{55}$$

$$\mathbf{G}_2 = \frac{1}{2}\frac{\partial^2 \mathbf{f}^{[i]}}{\partial \mathbf{y}^2}(B(\mathscr{E}\mathscr{T}_{\mathbf{y}_j}(\theta)) + B(\mathscr{E}(Q_{\mathbf{y}_j})))B(\mathscr{E}(Q_{\mathbf{y}_j}))B(\mathscr{E}(Q_{\mathbf{y}_j})) \tag{56}$$

where

$$B(\mathscr{E}\mathscr{T}_{\mathbf{y}_j}(\theta)) = B(\mathscr{P}_{\mathbf{y}_j}(\theta)) + B(Q_{\mathbf{y}_j})$$

and

$$B(Q_{\mathbf{y}_j}) = [-1, 1]\left(\sqrt{Q_{1,1}} \ldots \sqrt{Q_{n,n}}\right)^T$$

$$A_{\mathbf{f}^{[i]}} = mid(G_1) \tag{57}$$

$$N_{\mathbf{f}^{[i]}} = \mathbf{R}_{\mathbf{f}^{[i]}} \oplus (G_1 - A_{\mathbf{f}^{[i]}})\mathbf{R}_{\mathbf{y}_j} \oplus G_2 \tag{58}$$

Find $Q_{\mathbf{f}^{[i]}}$ such that $\mathscr{E}(A_{\mathbf{f}^{[i]}} Q_{\mathbf{y}_j} A_{\mathbf{f}^{[i]}}^T) \oplus N_{\mathbf{f}^{[i]}} \subseteq \mathscr{E}(Q_{\mathbf{f}^{[i]}})$

$$Q_{\mathbf{f}^{[i]}} = \frac{1}{\lambda_0} A_{\mathbf{f}^{[i]}} Q_{\mathbf{y}_j} A_{\mathbf{f}^{[i]}}^T + \sum_{ic=1}^{n} \frac{1}{\lambda_{ic}} rad(N_{\mathbf{f}^{[i]},ic})^2 e_{ic} e_{ic}^T,\ ic = 1,\ldots,n \tag{59}$$

Here, the unitary vector is represented by $e = (1,\ldots,1)^T = \sum_{ic} e_{ic}$. Also, $\sum_{i=0}^{n} \lambda_i \leq 1$. The scalars $\lambda_0, \lambda_1 \ldots \lambda_n$ can be chosen as

$$\lambda_0 = \frac{\sqrt{Tr(A_{\mathbf{f}^{[i]}} Q_{\mathbf{y}_j} A_{\mathbf{f}^{[i]}}^T)}}{\sqrt{Tr(A_{\mathbf{f}^{[i]}} Q_{\mathbf{y}_j} A_{\mathbf{f}^{[i]}}^T)} + \sum_{k=1}^{n} \|rad(N_{\mathbf{f}^{[i]},k})\|_2} \tag{60}$$

$$\lambda_{ic} = \frac{\|rad(N_{\mathbf{f}^{[i]},ic})\|_2}{\sqrt{Tr(A_{\mathbf{f}^{[i]}} Q_{\mathbf{y}_j} A_{\mathbf{f}^{[i]}}^T)} + \sum_{k=1}^{n} \|rad(N_{\mathbf{f}^{[i]},k})\|_2} \tag{61}$$

The Taylor Model with ellipsoidal remainder of the i^{th} Taylor coefficient $\mathbf{f}^{[i]}$ is

$$\mathscr{ET}_{\mathbf{f}^{[i]}}(\mathscr{ET}_{\mathbf{y}_j}(\theta),\theta) = \mathscr{P}_{\mathbf{f}^{[i]}}(\theta) \oplus \mathscr{E}(Q_{\mathbf{f}^{[i]}}) \tag{62}$$

First Stage

The first stage consists in determining a predicting enclosure of the solution for all $h \in (0, t_f - t]$

$$\mathscr{ET}_{\mathbf{y}_{j+1}}(\theta) = \bigoplus_{i=0}^{k-1} h_j^i \mathscr{ET}_{\mathbf{f}^{[i]}}(\mathscr{ET}_{\mathbf{y}_j}(\theta),\theta) \oplus h\delta[-e,e] \tag{63}$$

with a tolerance $\delta > 0$.

Second Stage

In the second stage a time step $\bar{h} > 0$ is computed such that the predicting enclosure (63) is guaranteed to provide a valid enclosure of the solution.

The iteration formula is initialized with

$$h_0 = \rho \left(\frac{\delta}{\|r(0)\|} \right)^{\frac{1}{k-1}} \tag{64}$$

where $r(h) = \mathscr{T}_{\mathbf{f}^{[k]}}(B(\mathscr{ET}_{\mathbf{y}_{j+1}}(\theta)),\theta)$.

When the inclusion $h^{k-1}r(h) \subseteq \delta[-e,e]$ is met h_0 is chosen otherwise the time step is modified with the adjusting factor ρ. If successful, the algorithm yields a valid enclosure of the solution.

Differential Inequality Bounds

As its name implies this method is based on the theory of differential inequalities [42]. Here the system of ODEs with interval values (initial conditions or parameters) is decomposed into an upper and lower bounding system. The resulting system of ODEs is twice as large as the original system and has to be solved by a conventional ODE solver; this is why this method is regarded as a non-verified method.

The upper and lower bounding ODEs are as follows. For $i = 1 \ldots n_x$,
with initial conditions: $\mathbf{y}_0 \subseteq [\mathbf{y}^L(0), \mathbf{y}^U(0)]$

$$\begin{aligned} \dot{y}_i^L(t) &\leq \min_{\substack{\mathbf{z}\in[\mathbf{y}^L(t),\mathbf{y}^U(t)],\theta\in\Theta \\ z_i=y_i^L(t)}} f_i(t,\mathbf{z},\theta) \\ \dot{y}_i^U(t) &\geq \max_{\substack{\mathbf{z}\in[\mathbf{y}^L(t),\mathbf{y}^U(t)],\theta\in\Theta \\ z_i=y_i^U(t)}} f_i(t,\mathbf{z},\theta) \end{aligned} \tag{65}$$

$\mathbf{y}(t) \subseteq [\mathbf{y}^L(t), \mathbf{y}^U(t)]$. The differential inequalities approach has been used with McCormick relaxations as well [34, 35]. This approach yields lower and upper relaxations of the ODE system which are convex and concave, respectively. Moreover, a generalized differential inequality has been proposed. This method allows the use of interval bounds as well as ellipsoidal set-propagation techniques [41].

Approximate Solution with Verified Enclosure

The ValEncIA-IVP [31] software package makes an implementation of an approximate solution with guaranteed exponential enclosure method. The software has recently been extended to solve differential-algebraic equations (DAEs) [29, 30]. In this method a nonguaranteed solution for the ODE system is obtained and taken as a reference solution and then an iterative computation of an interval enclosure is carried out, the enclosure obtained is then integrated in a verified way to obtain a guaranteed error bound. Finally, the Picard iteration is used to generate a solution where the part of the overestimation produced is avoided using a guaranteed exponential enclosure. This method does not make use of Taylor coefficients; it only uses derivatives to compute the Jacobian matrix of the

function. However, the method heavily relies on the quality of the nonguaranteed initial approximation since smaller deviations between the unknown exact solution and its initial approximation lead to tighter enclosures of the solution with less computational effort.

The enclosure is defined by

$$\mathbf{Y}(t) = \mathbf{y}_{app}(t) + \mathbf{R}(t) \tag{66}$$

Initialization of the algorithm with $\mathbf{R}_l(0) = \mathbf{R}_{l+1}(0) = \mathbf{Y}_0 - \mathbf{y}_{app}$

$$\dot{\mathbf{R}}_{l+1}(t) = -\dot{\mathbf{y}}_{app}(t) + \mathbf{f}(\mathbf{y}_{app}(t) + \mathbf{R}_l(t)) = \mathbf{r}(\mathbf{R}_l(t)) \tag{67}$$

Equation (67) takes as inputs values for $\mathbf{R}(t)$ and $\dot{\mathbf{R}}(t)$ and if the inclusion $\dot{\mathbf{R}}_1 \subseteq \dot{\mathbf{R}}_0$ holds, then the iteration can be continued. Otherwise, the initial guesses of $\mathbf{R}(t)$ and $\dot{\mathbf{R}}(t)$ have to be modified.

A non-verified solver is used to obtain $\mathbf{y}_{app}$ as well as the step sizes; however, in (67) an analytic expression is required for $\mathbf{y}_{app}$ and its time derivative $\dot{\mathbf{y}}_{app}$ so linear interpolation is used between grid points.

$$\mathbf{R}_{l+1}(t) = \mathbf{R}_{l+1}(0) + \int_0^t \mathbf{r}\left(\mathbf{R}_l([0;T])\right) ds \tag{68}$$

$$\mathbf{R}_{l+1}(t) \subseteq \mathbf{R}_{l+1}(0) + \mathbf{r}\left(\mathbf{R}_l([0;T])\right) \cdot t \tag{69}$$

$$\mathbf{R}_{l+1}(t) = \mathbf{R}_{l+1}([0;T]) \tag{70}$$

$$\mathbf{Y}_{l+1}(t) = \mathbf{Y}_{app}(t) + \mathbf{R}_{l+1}(t) \tag{71}$$

For the reduction of the overestimation a mean-value evaluation and a monotonicity test are performed.

Exponential State Enclosures

In order to prevent the growth of the overestimation an exponential state enclosure was proposed [30].

$$\mathbf{Y}(t) = e^{\Lambda \cdot t} \cdot \mathbf{Y}(0) \tag{72}$$

where $\Lambda = diag\{[\lambda_i]\}$ $i = 1, \ldots n_x$ is the diagonal matrix.

Here $\mathbf{Y}(t)$ is obtained by evaluating the iteration formula

$$[\lambda_{i,l+1}] = \frac{f_i(e^{(\Lambda_l \cdot [0;T])} \cdot \mathbf{Y}(0))}{e^{[\lambda_{i,l}] \cdot [0;T]} \cdot \mathbf{Y}_i(0)} \; i = 1, \ldots, n_x \tag{73}$$

This iteration formula is only applicable if $0 \notin \mathbf{Y}_i(t)$ for all $i = 1, \ldots, n_x$.

ITS Method with Overestimation Reduction

Reduction of the overestimation is an important issue in verified simulation. Several ways to tackle this problem are addressed by the techniques discussed in section "Enclosing the Solutions of IVPs for ODEs". In [45] an interesting way to reduce the dependency problem by using inner interval arithmetic in a balanced interval arithmetic approach is used to calculate ranges of functions. The ranges obtained are tighter but the computations used are no longer verified. In this chapter we make use of a verified approach. The method for which we present numerical experiments uses interval contractors [9] for the reduction of the overestimation. In this method an interval Taylor series as in section "Interval Remainder Term" is used.

Let us consider an implicit form of (7) for the second stage of the ITS method. If we make this reformulation, it is possible to consider the new implicit equation as a constraint satisfaction problem (CSP) in the form of $\mathbf{f}(\mathbf{y}) = 0$, $\mathbf{y} \in \mathbf{Y}_j$. The formulation is the following

$$\mathbf{g}(\mathbf{y}) = \mathbf{u}_{j+1} + [\mathbf{S}^y_{j+1}\mathbf{A}_j]\Gamma_j + \mathbf{S}^\theta_{j+1}[\Theta - \hat{\theta}] + \mathbf{Z}_{j+1} - \mathbf{Y}_{j+1} = 0, \mathbf{y} \in \mathbf{Y}_j \tag{74}$$

however this reformulation does not yield the form as in $\mathbf{f}(\mathbf{y}) = 0$ as the subtraction of identical vectors in interval arithmetic is not equal to 0 but an interval vector. So a midpoint evaluation is performed and we get

$$\begin{aligned}\mathbf{g}(\hat{\mathbf{y}}_j) =& \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\theta}) + \{[\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, \Theta)]\mathbf{A}_j\}\Gamma_j + [\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, \Theta)][\Theta - \hat{\theta}] \\ &+ \mathbf{Z}_{j+1}(\tilde{\mathbf{y}}_j, \Theta) - \hat{\mathbf{Y}}_{j+1}(\mathbf{Y}_j, \Theta) = 0\end{aligned} \tag{75}$$

which has the form of $\mathbf{f}(\mathbf{y}) = 0$. Now since

$$\hat{\mathbf{Y}}_{j+1}(\mathbf{Y}_j, \Theta) = \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\theta}) + \hat{\mathbf{z}}_{j+1}(\tilde{\mathbf{Y}}_j, \Theta)$$

we have

$$\begin{aligned}\mathbf{g}(\hat{\mathbf{y}}_j) =& \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\theta}) + \{[\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, \Theta)]\mathbf{A}_j\}\Gamma_j + [\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, \Theta)][\Theta - \hat{\theta}] \\ &+ \mathbf{Z}_{j+1}(\tilde{\mathbf{y}}_j, \Theta) - [\mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\theta}) + \hat{\mathbf{z}}_{j+1}(\tilde{\mathbf{Y}}_j, \Theta)] = 0\end{aligned}$$

$$\begin{aligned}=& \{[\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, \Theta)]\mathbf{A}_j\}\Gamma_j + [\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, \Theta)][\Theta - \hat{\theta}] \\ &+ \mathbf{Z}_{j+1}(\tilde{\mathbf{y}}_j, \Theta) - \hat{\mathbf{z}}_{j+1}(\tilde{\mathbf{Y}}_j, \Theta) = 0\end{aligned}$$

Also recalling (8)

$$\Gamma_{j+1} = \mathbf{A}^{-1}_{j+1}(\mathbf{S}^y_{j+1}\mathbf{A}_j)\Gamma_j + (\mathbf{A}^{-1}_{j+1}\mathbf{S}^\theta_{j+1})(\Theta - \hat{\theta}) + \mathbf{A}^{-1}_{j+1}(\mathbf{Z}_{j+1} - \hat{\mathbf{z}}_{j+1})$$

We get that

$$\mathbf{g}(\hat{\mathbf{y}}_j) = \mathbf{A}_{j+1}\Gamma_{j+1}(\hat{\mathbf{y}}_j, \Theta) \tag{76}$$

which corresponds to the global error in the verified simulation [17].

Now when there is no uncertainty then $\mathbf{g}(\hat{\mathbf{y}}_j) = \mathbf{A}_{j+1}\Gamma_{j+1}(\hat{\mathbf{y}}_j, \Theta) = 0$. In the present work uncertainty has been considered in the initial conditions and parameters of the ODEs and it will be shown in the next section that when a number of iterations of the Newton step are used in (76) reduction of the overestimation is achieved.

In summary, since (76) is defined at each time step it is possible to implement the contractors as in an equation of the form of $\mathbf{f}(\mathbf{y}) = 0$. In this way at each time step the interval Taylor series method obtains an interval $(\mathbf{Y}_{j+1})$ that encloses the solution of the problem. The midpoint $\hat{\mathbf{Y}}_{j+1}$ is then used to define (75). After a number of iterations if sufficient reduction is achieved, the verified method obtains a new $\mathbf{Y}_{j+1}$ and the contraction step is repeated; otherwise, the algorithm returns the best $\mathbf{Y}_{j+1}$ found so far. According to our previous work comparing the Krawczyk and Newton/Gauss–Seidel contractors, the Newton/Gauss–Seidel contractor proved to be superior in several models from chemical and biochemical engineering. Thus, only the Newton/Gauss–Seidel contractor is considered in this work [27].

Newton/Gauss–Seidel Contractor

We consider a CSP as in $\mathbf{f}(\mathbf{y}) = 0$ and apply the mean-value theorem to obtain

$$(\mathbf{f}(\hat{\mathbf{y}}_j) + \mathbf{J}_f(\boldsymbol{\xi})(\mathbf{y}_j - \hat{\mathbf{y}}_j) = 0, \mathbf{y}_j \in \mathbf{Y}_j, \boldsymbol{\xi} \in \mathbf{Y}_j) \tag{77}$$

The CSP in (77) can be arranged as

$$\begin{pmatrix} \mathbf{A}\mathbf{p} + \mathbf{f}(\hat{\mathbf{y}}_j) = 0 \\ \mathbf{p} = (\mathbf{y}_j - \hat{\mathbf{y}}_j) \\ \mathbf{A} = \mathbf{J}_f(\boldsymbol{\xi}) \\ \mathbf{b} = -\mathbf{f}(\hat{\mathbf{y}}_j) \\ \mathbf{y}_j \in \mathbf{Y}_j, \boldsymbol{\xi} \in \mathbf{Y}_j \end{pmatrix} \tag{78}$$

In this way, a linear contractor can be used such as the Gauss–Seidel contractor. The Gauss–Seidel contractor is able to contract domains of linear systems of the form

$$\mathbf{A}\mathbf{p} - \mathbf{b} = 0 \tag{79}$$

If A is square, it can be decomposed as the sum of a diagonal matrix and a matrix with zeroes on its diagonal (extdiag):

$$diag(\mathbf{A})\mathbf{p} + extdiag(\mathbf{A})\mathbf{p} = \mathbf{b} \tag{80}$$

Also if $\mathbf{A}$ is invertible, then (80) can be rewritten as

$$\mathbf{p} = (diag(\mathbf{A}))^{-1}(\mathbf{b} - extdiag(\mathbf{A})\mathbf{p}) \tag{81}$$

Hence, the solution of the Gauss–Seidel contractor is defined as the intersection of the original domain $\mathbf{p}$ and the new $\mathbf{p}$ calculated with (81). This results in

$$\mathbf{p} \leftarrow \mathbf{p} \cap (diag(\mathbf{A}))^{-1}(\mathbf{b} - extdiag(\mathbf{A})\mathbf{p}) \tag{82}$$

Finally, the Gauss–Seidel contractor solution in (82) is used to update $\mathbf{Y}_j$ and the intersection $\mathbf{Y}_j \leftarrow \mathbf{Y}_j \cap (\mathbf{p} + \hat{\mathbf{y}}_j)$ is obtained to finish with the Newton procedure.

Global Optimization Algorithm Using ITS with Overestimation Reduction

A sequential approach has been implemented to address the dynamic optimization problem. The formulation of the dynamic optimization problem can be stated as

$$\begin{aligned} \min_{\theta} \ & \phi(\mathbf{z}(t_i, \theta), \theta; i = 1, \ldots, ns) \\ s.t. \ \ & \dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \theta) \\ & \mathbf{z}(t_0, \theta) = \mathbf{z}_0(\theta) \\ & t \in [t_0, t_f] \\ & \theta \in \Theta \end{aligned} \tag{83}$$

where ϕ is the objective function, Θ is an interval vector, and $\mathbf{f}$ is assumed to be continuously differentiable with respect to $\mathbf{z}$ and θ. When a sequential approach is used a verified ODE method is applied to the dynamic part of the optimization problem leaving a problem only constrained by the system parameters θ.

Branch and Bound Algorithm

The spatial search procedure used was similar to a standard branch and bound algorithm by Moore et al. [22]. The global optimization method considers a problem of the form:

$$\begin{aligned} \min_x \phi &= f(x) \\ s.t. x &\in X \end{aligned} \tag{84}$$

1. The problem is first solved by a non-verified solver and the solution obtained (local minimum) is used as an initial upper bound, if no local solution can be obtained, then a first approximation of the upper bound can be obtained by $\overline{F(mid(X_0))}$.
2. The initial interval vector is taken as the next box to be processed.
   Set $X \leftarrow X_0$
3. Create two lists L and C; initially, these lists are empty.
4. Bisect X in the appropriate coordinate direction i such that $w(X_i) = w(X) = max_{1\leq i\leq n} w(X_i) : X = X^{(1)} \cup X^{(2)}$
5. Update the upper bound on the global optimum.
   $f_{ub} = min\{f_{ub}, \overline{F(mid(X^{(1)}))}, \overline{F(mid(X^{(2)}))}\}$
6. IF $max\{\overline{F(X^{(1)})}, \overline{F(X^{(2)})}\} - min\{\underline{F(X^{(1)})}, \underline{F(X^{(2)})}\} < \epsilon$, then
   (a) Test if 0 belongs to the gradient of the objective function and discard any box that does not satisfy this condition
   (b) Place $X^{(1)}$ and $X^{(2)}$ into C in order
   (c) IF L is not empty, then
      (i) remove the first item from L and place its box into X;
      (ii) IF $\underline{F(X)} > f_{ub}$, then
         RETURN with LB equal to lower bound on the first box in C and with the lists C and L
         END IF
      ELSE
      RETURN with LB equal to the lower bound on the first list in C and with list C
      END IF
   ELSE
   (a) Test if 0 belongs to the gradient of the objective function and discard any box that does not satisfy this condition
   (b) enter the items $(X^{(1)}, F(X^{(1)}))$ and $(X^{(2)}, F(X^{(2)}))$ in proper order in the list L;
   (c) set $X \leftarrow$ the argument (first member of the pair) of the first item in the list L (with the lowest $\underline{F(X)}$) and remove the item $(X, \underline{F(X)})$ from the list;
   END IF
7. IF L is not empty, then return to step 4.

This is the global optimization method that has been used in the numerical experiments. In this algorithm each time a box is branched new bounds are needed and so the bounding routine is called for each box. This call represents the most

expensive part in the branch and bound framework. Therefore it is needed to reduce the number of calls by discarding as many boxes as possible prior to the bounding step. A condition that tests whether or not zero is contained in the gradient of the objective function is being used in this algorithm (in order to compute the gradient the first order sensitivity equations are solved). Another way to discard boxes is by means of the second order sensitivities. With this information one can test for convexity of the function.

Results

The dynamic optimization method features a method that enhances the overestimation capabilities of a traditional method for verified simulation (ITS) and makes it suitable for its application in the solution of dynamic optimization problems.

The next three examples were solved in a sequential approach and using the interval Taylor series with the Newton/Gauss–Seidel contractor to bound the dynamic variables. The CPU times are for an Intel^{TM} Core^{TM} i5 with 8 GB RAM, running Ubuntu 14 and gcc 4.8.4. The programs were written in C++ and the third party libraries FADBAD++ [38] and Profil/Bias [11] were used for the automatic differentiation and the interval arithmetic operations, respectively. The Profil/Bias library was used since it is faster when only interval arithmetic operations and the square function are needed [44] as in the examples considered.

First Order Irreversible Series Reaction

A first order irreversible chain reaction taken from [39] considers the following reaction

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

The procedure developed in section "ITS Method with Overestimation Reduction" is applied to the parameter estimation problem with two parameters and two dynamic variables. The experimental data has been taken from [6]. The problem has been solved to global optimality using an absolute tolerance $\varepsilon_{abs} = 10^{-4}$ in 5.39 s. [14] solved the problem with a relative tolerance $\varepsilon_{rel} = 10^{-3}$ and exactly ($\varepsilon = 0$) in 0.023 and 0.059 s, respectively. The machine used was an Intel Pentium 4 3.2 GHz. Singer and Barton [36] solved the problem with an absolute tolerance $\varepsilon_{abs} = 10^{-4}$ in 0.036 s in an AMD Athlon XP2000+ 1667 MHz. However, the differential inequalities approach is used in this work in which the auxiliary system is solved by a conventional solver and hence the solution is not computationally verified. Papamichail and Adjiman [26] solved the problem in 9 and 11 s using constant, and constant and affine underestimation schemes, respectively. The tolerance they used

was $\varepsilon_{rel} = 10^{-7}$ in an UltraSPARC-II 2×360 MHz.

$$\begin{aligned} \min_{\mathbf{k}} \phi &= \sum_{j=1}^{10} \sum_{i=1}^{2} (x_i(t_j) - x_i^{exp}(t_j)) \\ s.t.\ \dot{x}_1 &= -k_1 x_1 \\ \dot{x}_2 &= k_1 x_1 - k_2 x_2 \\ x_1(0) &= 1,\ x_2(0) = 0 \\ t &\in [0,1] \\ \mathbf{k} &\in [0,10] \times [0,10] \end{aligned} \tag{85}$$

Oil Shale Pyrolysis

The optimal temperature profile in a plug flow reactor is considered. The reactions involved and the model of the problem are shown in (86). In the model only components A_1 and A_2 are included and the objective is to maximize the production of A_2. Here u is the adjustable parameter and is taken as a piecewise constant profile.

$$\begin{aligned} & & \min_{u}\ \phi &= -x_2(t_f) \\ A_1 \xrightarrow{k_1} A_2 & & s.t.\ \dot{x}_1 &= -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \\ A_2 \xrightarrow{k_2} A_3 & & \dot{x}_2 &= k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \\ A_1 + A_2 \xrightarrow{k_3} 2A_2 & & k_i &= a_i e^{\left(\frac{-b_i/R}{698.15+50u}\right)},\ i = 1,\ldots,5 \\ A_1 + A_2 \xrightarrow{k_4} A_3 + A_2 & & x_1(0) &= 1,\ x_2(0) = 0 \\ A_1 + A_2 \xrightarrow{k_5} A_4 + A_2 & & t &\in [0,10] \\ & & u &\in [0,1] \end{aligned} \tag{86}$$

The dynamic optimization problem has been solved to global optimality using an absolute tolerance of $\varepsilon_{abs} = 10^{-3}$ in 116.62 s. The same problem was solved by [13] in 3.2 s using $\varepsilon_{abs} = 10^{-3}$ and an Intel Pentium 4 3.2 GHz. Singer and Barton [36] solved the problem in a non-verified manner in 27.30 and 26.20 s without and with heuristics, respectively. The machine used was an AMD Athlon XP2000+ 1667 MHz.

Singular Control Problem

The procedure from section "ITS Method with Overestimation Reduction" is applied to a nonlinear singular control problem taken from [18].

$$\begin{aligned}
\min_u \phi &= \int_{t_0}^{t_f} [x_1^2 + x_2^2 + 0.0005(x_2 + 16t - 8 - 0.1x_3u^2)^2]dt \\
s.t.\ \dot{x}_1 &= x_2 \\
\dot{x}_2 &= -x_3u + 16t - 8 \\
\dot{x}_3 &= u \\
x_1(0) &= 0,\ x_2(0) = -1,\ x_3(0) = -\sqrt{5} \\
t &\in [0,1] \\
u &\in [-4,10]
\end{aligned} \tag{87}$$

The sequential approach was used to provide a solution for this problem. The computational time the method took in solving the problem was 18.95 s with an absolute tolerance of $\varepsilon_{abs} = 10^{-3}$. The problem was also solved by Lin and Stadtherr [15] in 0.02 s with the same absolute tolerance. It is worth mentioning that their global optimization algorithm implements a branch and reduce approach and is able to reduce the search space. The machine used was an Intel Pentium 4 3.2 GHz. Also [36] provided a (non-verified) solution for the problem with the same tolerance in 2 and 1.8 s without and with heuristics. They used an AMD Athlon XP2000+ 1667 MHz to solve the problem.

While the method presented is not faster than the method that features Taylor Models it provides the ideas of an interval contracting technique that can be extended to other verified simulation methods (Taylor Models and differential inequalities) and in turn other dynamic optimization methods.

Conclusions

The reduction of the overestimation by means of the interval Newton/Gauss–Seidel contractor allowed the use of an ITS method in the solution of dynamic optimization problems. This suggests that the application of these kinds of contracting techniques to other verified simulation methods could improve the performance when addressing dynamic optimization problems.

The method proposed is able to solve dynamic optimization problems to global optimality in a verified way. More sophisticated global optimization strategies such as the branch and reduce approach can be implemented as the comparisons show that the times are an order of magnitude bigger or more with respect to other works.

So far there is no universal technique that is able to provide acceptable bounds for all kinds of problems and thus more research is being carried out. The main challenges are the scalability of these kinds of methods to higher dimensional problems and significant uncertain amounts. For the method presented we would expect it to be easier to scale since it does not need the symbolic computations which Taylor Models do.

Finally, for future research directions the use of interval contractors in the Taylor Models method seems to be promising for the reduction of the overestimation. Some other techniques that take advantage of the constraint satisfaction problem could be implemented; for example, constraint propagation, linear programming, and other types of contractors.

Acknowledgements Financial support from CONACYT (Mexico), UCL, and SEP DGRI (Mexico) is gratefully acknowledged.

References

1. Adjiman, C.S., Androulakis, I., Maranas, C., Floudas, C.A.: A global optimization method, aBB, for process design. Eur. Symp. Comput. Aided Process Eng. **20**(96), 419–424 (1996)
2. Bhatia, T., Biegler, L.T.: Dynamic optimization in the design and scheduling of multiproduct batch Plants. Ind. Eng. Chem. Res. **35**(7), 2234–2246 (1996)
3. Biegler, L.T.: Nonlinear Programming Concepts, Algorithms, and Applications to Chemical Processes, 1 edn. SIAM, Philadelphia (2010)
4. Eijgenraam, P.: The Solution of Initial Value Problems Using Interval Arithmetic: Formulation and Analysis of an Algorithm. Mathematisch Centrum, Amsterdam (1981)
5. Esposito, W.R., Floudas, C.A.: Deterministic global optimization in nonlinear optimal control problems. J. Glob. Optim. **17**(1–4), 97–126 (2000)
6. Esposito, W.R., Floudas, C.A.: Global optimization for the parameter estimation of differential-algebraic systems. Ind. Eng. Chem. Res. **39**(5), 1291–1310 (2000)
7. Flores-Tlacuahuac, A., Biegler, L.T., Saldívar-Guerra, E.: Dynamic optimization of HIPS open-loop unstable polymerization reactors. Ind. Eng. Chem. Res. **44**(8), 2659–2674 (2005)
8. Houska, B., Villanueva, M.E., Chachuat, B.: A validated integration algorithm for nonlinear ODEs using Taylor models and ellipsoidal calculus. In: 52nd IEEE Conference on Decision and Control, pp. 484–489, Florence (2013)
9. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, London (2001)
10. Kloppenburg, E., Gilles, E.D.: A new concept for operating simulated moving-bed processes. Chem. Eng. Technol. **22**(10), 813–817 (1999)
11. Knuppel, O.: PROFIL/BIAS-A fast interval library. Computing **53**, 277–287 (1994)
12. Kruckeberg, F.: Ordinary differential equations. In: Hansen, E. (ed.) Topics in Interval Analysis, pp. 91–97. Clarendon Press, Oxford (1969)
13. Lin, Y., Stadtherr, M.A.: Deterministic global optimization for dynamic systems using interval analysis. In: 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006), pp. 38–38 (2006)
14. Lin, Y., Stadtherr, M.A.: Deterministic global optimization for parameter estimation of dynamic systems. Ind. Eng. Chem. Res. **45**(25), 8438–8448 (2006)
15. Lin, Y., Stadtherr, M.A.: Deterministic global optimization of nonlinear dynamic systems. AIChE J. **53**(4), 866–875 (2007)

16. Lin, Y., Stadtherr, M.A.: Validated solutions of initial value problems for parametric ODEs. Appl. Numer. Math. **57**(10), 1145–1162 (2007)
17. Lohner, R.J.: Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In: Cash, J.R., Gladwell, I. (eds.) Computational Ordinary Differential Equations, pp. 425–435. Clarendon Press, Oxford (1992)
18. Luus, R.: Optimal control by dynamic programming using systematic reduction in grid size. Int. J. Control **51**(5), 995–1013 (1990)
19. Makino, K., Berz, M.: Remainder differential algebras and their applications. In: Berz, M., Bischof, C., Corliss, G., Griewank, A. (eds.) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia (1996)
20. Makino, K., Berz, M.: COSY INFINITY version 9. Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip. **558**(1), 346–350 (2006). Proceedings of the 8th International Computational Accelerator Physics Conference (ICAP) (2004)
21. Moore, R.E.: Interval arithmetic and automatic error analysis in digital computing. Ph.D. thesis, Stanford University (1962)
22. Moore, R.E., Kearfort, R.B., Cloud, M.J.: Introduction to Interval Analysis, vol. 22. SIAM, Philadelphia (2009)
23. Nedialkov, N.S.: Implementing a rigorous ode solver through literate programming. In: Rauh, A., Auer, E. (eds.) Modeling, Design, and Simulation of Systems with Uncertainties. Mathematical Engineering, vol. 3, pp. 3–19. Springer, Berlin/Heidelberg (2011)
24. Oldenburg, J., Marquardt, W., Heinz, D., Leineweber, D.B.: Mixed-logic dynamic optimization applied to batch distillation process design. AIChE J. **49**(11), 2900–2917 (2003)
25. Papamichail, I., Adjiman, C.S.: A rigorous global optimization algorithm for problems with ordinary differential equations. J. Glob. Optim. **24**, 1–33 (2002)
26. Papamichail, I., Adjiman, C.S.: Global optimization of dynamic systems. Comput. Chem. Eng. **28**(3), 403–415 (2004)
27. Perez-Galvan, C., Bogle, I.D.L.: Comparison between interval methods to solve initial value problems in chemical process design. In: Klemeš, J.J., Varbanov, P.S., Liew, P.Y. (eds.) 24th European Symposium on Computer Aided Process Engineering, vol. 33, pp. 1405–1410. Elsevier, Budapest (2014)
28. Raghunathan, A.U., Soledad Diaz, M., Biegler, L.T.: An MPEC formulation for dynamic optimization of distillation operations. Comput. Chem. Eng. **28**(10), 2037–2052 (2004)
29. Rauh, A., Auer, E.: Verified simulation of ODEs and DAEs in ValEncIA-IVP. Reliab. Comput. **15**, 370–381 (2011)
30. Rauh, A., Brill, M., Günther, C.: A novel interval arithmetic approach for solving differential-algebraic equations with ValEncIA-IVP. Int. J. Appl. Math. Comput. Sci. **19**(3), 381–397 (2009)
31. Rauh, A., Hofer, E., Auer, E.: Valencia-ivp: a comparison with other initial value problem solvers. In: Klemeš, J.J., Varbanov, P.S., Liew, P.Y. (eds.) 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006, p. 36. IEEE, Duisburg (2006)
32. Sahlodin, A.M., Chachuat, B.: Convex/concave relaxations of parametric ODEs using Taylor models. Comput. Chem. Eng. **35**, 844–857 (2011)
33. Sahlodin, A.M., Chachuat, B.: Discretize-then-relax approach for convex/concave relaxations of the solutions of parametric ODEs. Appl. Numer. Math. **61**(7), 803–820 (2011)
34. Scott, J.K., Barton, P.I.: Improved relaxations for the parametric solutions of ODEs using differential inequalities. J. Glob. Optim. **57**(1), 143–176 (2013)
35. Scott, J.K., Chachuat, B., Barton, P.I.: Nonlinear convex and concave relaxations for the solutions of parametric ODEs. Optim. Control Appl. Methods **34**, 145–163 (2013)
36. Singer, A.B., Barton, P.I.: Global optimization with nonlinear ordinary differential equations. J. Glob. Optim. **34**(2), 159–190 (2006)
37. Singer, A.B., Taylor, J.W., Barton, P.I., Green, W.H.: Global dynamic optimization for parameter estimation in chemical kinetics. J. Phys. Chem. **110**, 971–976 (2006)

38. Stauning, O., Bendtsen, C.: FADBAD++ web page. http://www.fadbad.com/fadbad.html (2003)
39. Tjoa, I.B., Biegler, L.T.: Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. Ind. Eng. Chem. Res. **30**(2), 376–385 (1991)
40. Villanueva, M.E., Houska, B., Chachuat, B.: On the stability of set-valued integration for parametric nonlinear odes. 24th European Symposium on Computer Aided Process Engineering, Pts A and B, vol. 33, pp. 595–600 (2014)
41. Villanueva, M.E., Houska, B., Chachuat, B.: Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs. J. Glob. Optim. **62**(3), 575–613 (2014)
42. Walter, W.: Differential and Integral Inequalities. Translated by Rosenblatt, L., Shampine, L. Springer, Heidelberg (1970)
43. Zhao, Y., Stadtherr, M.A.: Rigorous global optimization for dynamic systems subject to inequality path constraints. Ind. Eng. Chem. Res. **50**(22), 12678–12693 (2011)
44. Žilinskas, J.: Comparison of packages for interval arithmetic. Informatica **16**(1), 145–154 (2005)
45. Žilinskas, J., Bogle, I.D.L.: Evaluation ranges of functions using balanced random interval arithmetic. Informatica **14**(3), 403–416 (2003)
46. Žilinskas, J., Bogle, I.D.L.: Balanced random interval arithmetic. Comput. Chem. Eng. **28**(5), 839–851 (2004). 10.1016/j.compchemeng.2004.02.020
47. Žilinskas, J., Bogle, I.D.L.: Global optimization: interval analysis and balanced interval arithmetic. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 1346–1350. Springer, New York (2009). doi:10.1016/10.1007/978-0-387-74759-0_237
48. Žilinskas, A., Žilinskas, J.: On efficiency of tightening bounds in interval global optimization. In: Applied Parallel Computing. State of the Art in Scientific Computing. Lecture Notes in Computer Science, vol. 3732, pp. 197–205. Springer, Heidelberg (2006)

On the Least-Squares Fitting of Data by Sinusoids

Yaroslav D. Sergeyev, Dmitri E. Kvasov, and Marat S. Mukhametzhanov

Abstract The sinusoidal parameter estimation problem is considered to fit a sum of damped sinusoids to a series of noisy observations. It is formulated as a nonlinear least-squares global optimization problem. A one-parametric case study is examined to determine an unknown frequency of a signal. Univariate Lipschitz-based deterministic methods are used for solving such problems within a limited computational budget. It is shown that the usage of local information in these methods (such as local tuning on the objective function behavior and/or evaluating the function first derivatives) can significantly accelerate the search for the problem solution with a required guarantee. Results of a numerical comparison with metaheuristic techniques frequently used in engineering design are also reported and commented on.

Keywords Nonlinear regression • Least-squares fitting • Lipschitz-based deterministic methods • Metaheuristics • Numerical comparison

Introduction

A general nonlinear regression model can be often considered in the form of fitting a sum of damped sinusoids to a series of observations corrupted by noise (see, e.g., [2, 5, 10, 16, 31]). The sinusoidal functions are frequently used in many real-life applications such as signal processing (see, e.g., [6, 7, 9, 14, 19, 33, 39]). Parameters **x** of these functions (consisting of amplitudes, frequencies, and phases) can be estimated by solving the following minimization problem:

Y.D. Sergeyev (✉) • D.E. Kvasov • M.S. Mukhametzhanov
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, Via P. Bucci, Cubo 42C – 87036 Rende (CS), Italy

Department of Software and Supercomputing Technologies, Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia
e-mail: yaro@dimes.unical.it; kvadim@dimes.unical.it; marat@dimes.unical.it

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_11

$$f(\mathbf{x}^*) = f^* = \min_{\mathbf{x} \in D} f(\mathbf{x}), \quad f(\mathbf{x}) = \sum_{i=1}^{T} (y_{t_i} - \phi(\mathbf{x}, t_i))^2, \quad \mathbf{x} \in D \subset \mathbb{R}^n, \tag{1}$$

where

$$\phi(\mathbf{x}, t) = \sum_{l=1}^{s} a_l e^{d_l t} \sin(2\pi\omega_l t + \theta_l), \quad t = t_1, \ldots, t_T, \tag{2}$$

and $s \geq 1$ is a fixed integer, $\mathbf{x} = (\mathbf{a}, \mathbf{d}, \omega, \theta)$ with $\mathbf{a} = (a_1, \ldots, a_s)$, $\mathbf{d} = (d_1, \ldots, d_s)$, $\omega = (\omega_1, \ldots, \omega_s)$, and $\theta = (\theta_1, \ldots, \theta_s)$.

It is supposed that real-valued observations y_{t_i} are affected by noise:

$$y_{t_i} = \phi(\bar{\mathbf{x}}, t_i) + \xi_{t_i}, \quad i = 1, \ldots, T, \tag{3}$$

where $\bar{\mathbf{x}}$ is the true vector of parameters (it coincides with the estimator $\mathbf{x}^*$ from (1) in the case of noise-free observations) and ξ_{t_i}, $i = 1, \ldots, T$, are independently and identically distributed random variables with zero mean and a given variance σ^2.

In spite of the practical importance of problem (1)–(3), not so many publications that discuss the behavior of the objective function (1)–(2) are available (see, e.g., [15, 17, 18, 22, 29, 54, 60]). For example, in [16, 17], its multiextremal character is observed and it is experimentally demonstrated that the noise-free part of the objective function dominates its total shape. Noise-corrupted data in (1)–(3) increase furthermore the complexity of the objective function (see, e.g., [1, 3, 4, 57]). As observed, e.g., in [15, 17], the optimization problem (1)–(3) is difficult even in the case $s = 1$ in (2) since the objective function possesses many local minima and a guarantee on the found solution to the problem is required in practice. Even though the function to be minimized in (1) is Lipschitz-continuous and differentiable (see, e.g., [18, 28, 36, 47, 55, 59]), its Lipschitz constant is very high (as well as that of the function derivatives) and rapidly increases with the number of observations T in (3). Moreover, a higher number of observations lead to a more oscillating objective function with a higher number of local minima. Therefore, efficient global optimization approaches should be adopted to solve problem (1)–(3).

In [15], it is shown that Lipschitz-based deterministic algorithms can be successfully used for studying the stated global optimization problem, since these methods can often provide a solution to the problem together with its global optimality certificate (see, e.g., [34, 35, 38, 46, 49, 52, 53, 55, 56]). In this chapter, we will further analyze the usage of Lipschitz global optimization methods for solving problem (1)–(3), illustrating their numerical performance on specific instances of the problem. A particular attention will be dedicated to a number of local tuning techniques improving the performance of the methods and allowing one to accelerate the search of the optimal parameters $\mathbf{x}^*$ in terms of the number of function evaluations (which can be often computationally expensive).

The rest of this chapter is structured as follows. In the next section "Case Study", some benchmark one-parametric instances of the general regression problem (1)–(3)

are described. In section "Univariate Lipschitz Global Optimization Methods", a general algorithmic scheme of Lipschitz-based deterministic methods is given and some approaches for the usage of local information within this scheme are indicated. Results of numerical experiments with the Lipschitz-based methods on the problems from section "Case Study" and their comparison with some widely used metaheuristic methods are reported and discussed in section "Univariate Lipschitz Global Optimization Methods". Section "Conclusions" concludes the chapter.

Case Study

As a case study to illustrate the performance of various techniques for solving the global optimization problem (1)–(3), let us consider the sine function with unknown frequency only in (2) [i.e., $s = 1$ and $a_1 = 1$, $d_1 = 0$, $\theta_1 = 0$ in (2)] over uniformly sampled observations y_i, $i = 1, \ldots, T$ in (3). In this case, the vector of parameters $\mathbf{x}$ consists of only one component $x := \omega = \omega_1$. In spite of its apparent simplicity, such a problem is however representative from the practical point of view (see, e.g., [2, 10, 39]) and useful to obtain conclusions on the problem behavior that can be then generalized to a general multiparametric model (1)–(3).

Problem (1)–(3) is thus reduced to the following one-dimensional global minimization problem:

$$f(x^*) = f^* = \min_{x \in [a,b]} f(x), \quad f(x) = \sum_{i=1}^{T} (y_i - \sin(2\pi x\, i))^2, \quad x \in [a,b] \subset \mathbb{R}, \tag{4}$$

where for any fixed number of observations T the function $f(x)$ is Lipschitzian with the Lipschitz constant L, $0 < L < \infty$, and continuously differentiable (see [15, 18]) with the Lipschitz constant K, $0 < K < \infty$, for its Lipschitzian first derivative $f'(x)$ over the search interval $[a,b]$ [taken here as $[0,1]$ due to the periodicity of the sine function in (4)].

By changing the number of observations T in (3), the objective functions (4) of different shapes can be obtained, with the number of local minima and the average function value proportional to T. In our case study, the values $T = 10$, 50, and 100 were considered. The noise-free observations y_i, $i = 1, \ldots, T$, were obtained numerically to keep the estimator x^* in (4) equal to the true parameter $\bar{x} = 0.7$ with $f(x^*) = 0$. The noisy observations were then produced by adding random variables ξ_i, $i = 1, \ldots, T$, taken from normal distribution ($\mathbf{N}(0, \sigma^2)$) with $\sigma^2 = 9$ (see, e.g., [16, 17] for other noise parameters), to the corresponding noise-free values. Consequently, the global minimizer x^* in (4) was shifted from the true parameter value $\bar{x}$ (see the second column in Table 1) and the (non-normalized) objective function (4) became more erratic (with an unknown positive minimum value f^*). The search for the global minimizer x^* from (4), rather than for the true parameter value $\bar{x}$, was performed in this case too, in order to estimate the behavior of the numerical methods from the global optimization viewpoint.

Table 1 The Lipschitz constants L and K for the objective functions $f(x)$ from (4) and their derivatives $f'(x)$, respectively, considered in our case study with different numbers of observations T, noise terms from normal distribution $\mathbf{N}(0,\sigma^2)$, and the true frequency value $\bar{x} = 0.7$

Function (T,σ^2)	x^*	Lipschitz constant L	Lipschitz constant K
$(10,0)$	$\bar{x}$	354.1	30,567.2
$(50,0)$	$\bar{x}$	7216.4	3,390,330.5
$(100,0)$	$\bar{x}$	28,126.7	26,717,323.0
$(10,9)$	0.6126936	978.2	47,952.8
$(50,9)$	0.7650428	19,486.4	6,602,640.8
$(100,9)$	0.3055982	56,272.4	26,724,566.7

Hence, six particular functions (4) were considered in our study: three of them were based on noise-free terms in (3) and the other three—on the corresponding noisy observations. In what follows, these functions are labelled as pairs (T,σ^2), with $T = 10, 50$, and 100 and $\sigma^2 = 9$. The values (determined over 10^{-7} grid) of the Lipschitz constants L and K for $f(x)$ and $f'(x)$, respectively, are reported in Table 1. It can be seen from this table how the complexity of the functions increases both with the increasing of T and with adding the noise. Moreover, high values of the Lipschitz constants (especially, for K) can be observed from Table 1: they make the usage of Lipschitz global optimization methods challenging to solve the stated problem either in its one-parametric variant (4) or in its general form (1)–(3).

The complexity of the considered instances of problem (1)–(3) from the global optimization point of view can be also seen from the graphs of the objective functions $f(x)$ from Table 1 and their first derivatives reported in Figs. 1 and 2 for the cases of noise-free ($\sigma^2 = 0$) and noisy ($\sigma^2 = 9$) observations, respectively. As one can note, the objective functions are highly multiextremal and irregular, with the global minimizers having narrow attraction regions, although well separated with respect to the global minimum values for higher numbers of observations T. Hence, already for our case study which is a relatively simple case with respect to the general problem (1)–(3), a particular attention should be paid to the choice of numerical methods able to tackle efficiently the stated global optimization problem.

Univariate Lipschitz Global Optimization Methods

To obtain guaranteed estimates of the global solution to problem (4) within a finite number of function evaluations, as required in many engineering design problems (including the stated one), the framework of Lipschitz global optimization can be used since the Lipschitz-continuity property often appears naturally in practical optimization (see, e.g., [8, 26, 27, 38, 46, 51, 52, 55, 56]). Lipschitz global optimization methods can have an intuitive geometric interpretation and many of them can be successfully described and studied in a unique theoretical framework

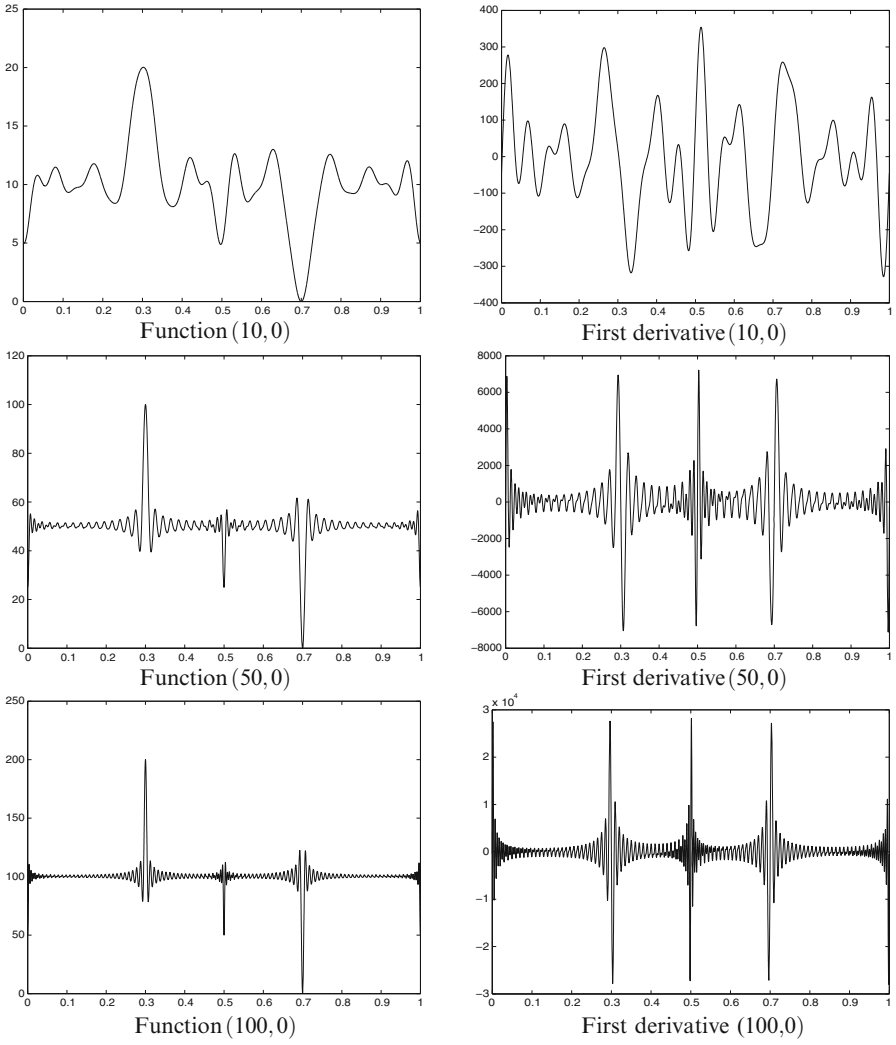

Fig. 1 Graphs of the objective functions (*left column*) and their first derivatives (*right column*) for the considered noise-free problems from Table 1

as, e.g., in the framework of divide-the-best algorithms (see, e.g., [32, 38, 43, 46]). Therefore, we will apply them to solving problem (4) (both in the case of derivative-free methods when the information on the function derivatives is not available for some reason and in the case when this information is taken into consideration during the work of an algorithm). A particular attention will be paid to the usage of local properties of the objective function (4) in the scheme of a global optimization

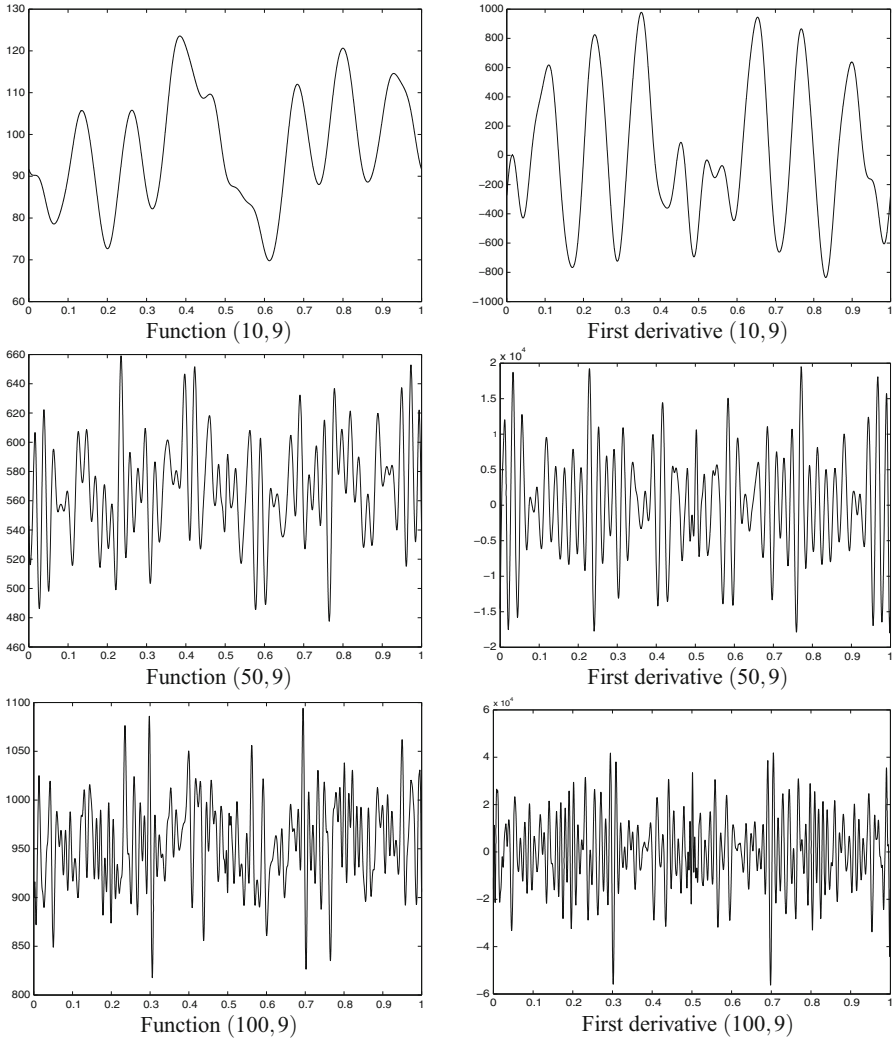

Fig. 2 Graphs of the objective functions (*left column*) and their first derivatives (*right column*) for the considered noisy problems from Table 1

method. This kind of local approach is able to tune the algorithm's execution on the objective function behavior and to determine the global solution with both a smaller number of function evaluations and a higher precision.

The Lipschitz global optimization methods proposed for solving one-dimensional problem (4) can be described by the following general scheme [in what follows, the term trial will indicate the operation of evaluating the objective function, and also its first derivative if required by a particular algorithm, at a point of the search interval $[a, b]$ from (4)].

Step 0. (*Initialization.*) Perform the first two trials at points a and b: $x^1 := a$, $z^1 := f(x^1)$ (and, in case the algorithm uses derivatives in its work, $\mathrm{d}z^1 := f'(x^1)$); $x^2 := b$, $z^2 := f(x^2)$ (and $\mathrm{d}z^2 := f'(x^2)$). Set the iteration counter $k := 2$.

The next $(k+1)$th iteration is realized as follows.

Step 1. (*Reordering trial points.*) Renumber the trial points $x^1, \ldots, x^k$ and the corresponding functions values by subscripts so that $a = x_1 < \ldots < x_k = b$.

Step 2. (*Estimating the Lipschitz constants.*) Calculate the current estimates L_j or K_j of the Lipschitz constants L or K for $f(x)$ or $f'(x)$ over each sub-interval $[x_{j-1}, x_j]$, $j = 2, \ldots, k$, in one of the following ways.

Step 2.1. (*A priori estimate.*) Take some a priori given value L or K (as, for example, those from Table 1 in section "Case Study") as an estimate of the Lipschitz constant for $f(x)$ or $f'(x)$ over the whole search interval $[a,b]$ from (4), i.e., set $L_j := L$ or $K_j := K$.

Step 2.2. (*Global estimate.*) Set $L_j := r \cdot \max\{H^k, \eta\}$ or $K_j := r \cdot \max\{G^j, \eta\}$, where $r > 1$ is the reliability parameter of the method under consideration and η is a small technical parameter (set here as $\eta = 10^{-8}$), and, for every sub-interval j, $2 \le j \le k$:

$$H_j = \frac{|z_j - z_{j-1}|}{x_j - x_{j-1}}, \tag{5}$$

with

$$H^k = \max\{H_j : \ j = 2, \ldots, k\} \tag{6}$$

(for the case of a derivative-free algorithm); or

$$G_j = \frac{|2(z_{j-1} - z_j) + (\mathrm{d}z_i + \mathrm{d}z_{i-1})(x_j - x_{j-1})| + \delta_j}{(x_j - x_{j-1})^2}, \tag{7}$$

with

$$\delta_j = \{[2(z_{j-1} - z_j) + (\mathrm{d}z_j + \mathrm{d}z_{j-1})(x_j - x_{j-1})]^2 + \\ + (\mathrm{d}z_j - \mathrm{d}z_{j-1})^2 (x_j - x_{j-1})^2\}^{1/2}$$

(for the case of an algorithm using the first derivative information).

Step 2.3. ("*Maximum*" *local tuning.*) In the case of a derivative-free algorithm, set $L_j := r \cdot \max\{\lambda_j, \gamma_j, \eta\}$, where

$$\lambda_j = \max\{H_{j-1}, H_j, H_{j+1}\},$$

with H_j from (5), when $j = 2$ or $j = k$ then just two respective values of H_j are used in this formula, and

$$\gamma_j = H^k \frac{x_j - x_{j-1}}{X^{\max}},$$

where H^k is from (6), $X^{\max} = \max\{x_j - x_{j-1} : j = 2,\ldots,k\}$, and r and η have the same sense as for global estimates (see Step 2.2).
In the case of an algorithm using the first derivative information, set the estimate $K_j := r \cdot \max\{\Lambda_j, \Gamma_j, \eta\}$, where

$$\Lambda_j = \max\{G_{\tilde{j}} : 2 \le \tilde{j} \le k, j-1 \le \tilde{j} \le j+1\},$$
$$\Gamma_j = \max\{G_j : j = 2,\ldots,k\} \cdot (x_j - x_{j-1})/X^{\max},$$

with G_j from (7), $X^{\max} = \max\{x_j - x_{j-1} : j = 2,\ldots,k\}$, and r and η have the same sense as for global estimates (see Step 2.2).

Step 3. (*Calculating the characteristics.*) For each sub-interval $[x_{j-1}, x_j]$, $j = 2,\ldots,k$, compute its characteristic R_j (see, e.g., [21, 46, 52] for more details on the characteristic global optimization methods) by using one of the following rules (see Steps 3.1 and 3.2 for a derivative-free algorithm and Step 3.3 for an algorithm using the first derivative information).

Step 3.1. (*Geometric characteristics.*)

$$R_j = \frac{z_j + z_{j-1}}{2} - L_j \frac{x_j - x_{j-1}}{2}.$$

Step 3.2. (*Information characteristics.*)

$$R_j = -L_j(x_j - x_{j-1}) - \frac{(z_j - z_{j-1})^2}{L_j(x_j - x_{j-1})} + 2(z_j + z_{j-1}).$$

Step 3.3. (*Smooth characteristics.*) The lower bound of a smooth auxiliary function over each sub-interval j, $2 \le j \le k$, can be calculated by using an estimate K_j of the Lipschitz constant K for $f'(x)$, as reported, e.g., in [25, 42, 46].

Step 4. (*Sub-interval selection.*) Select the sub-interval $[x_{t-1}, x_t]$ such that $t = t(k) = \arg\min\{R_j : j = 2,\ldots,k\}$ (the most "promising" sub-interval for a subsequent subdivision).

Step 5. (*Internal stopping criterion.*) If

$$x_t - x_{t-1} \le \varepsilon, \tag{8}$$

where $\varepsilon > 0$ is a given search accuracy, then **Stop** the algorithm and take the value $f_k^* = \min\{z_j : j = 1,\ldots,k\}$ as an estimate of the global minimum f^* from (4) and the value $x_k^* = \arg\min\{z_j : j = 1,\ldots,k\}$ as an estimate of the global minimizer x^* from (4). Criterion (8) can be considered as a global optimality certificate of

the solution x_k^*, f_k^* (when the global convergence conditions are satisfied: see the related discussion in [15, 27, 28, 45, 46, 52]).
Otherwise, go to Step 6.

Step 5. (*New trial.*) Perform the next trial either at the point

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{z_t - z_{t-1}}{2L_t},$$

for a derivative-free algorithm or at the point providing the minimum value of the current smooth auxiliary function (see Step 3.2) which can be calculated by an explicit formula (see [25, 42, 46] for details).
Update the iteration counter $k := k + 1$ and go to Step 1.

The following Lipschitz global optimization methods belonging to this general scheme were used in our study:

Geom-A*L*: **Geom**etric derivative-free method with **A** priori given Lipschitz constant ***L*** (see, e.g., [8, 46, 52, 53]): see Step 2.1 and Step 3.1 in the general scheme.

Geom-G*L*: **Geom**etric derivative-free method with the **G**lobal estimate of the Lipschitz constant ***L*** (see, e.g., [38, 46, 49, 52]): see Step 2.2 and Step 3.1 in the general scheme.

Inf-G*L*: **Inf**ormation-statistical derivative-free method with the **G**lobal estimate of the Lipschitz constant ***L*** (see, e.g., [49, 52]): see Step 2.2 and Step 3.2 in the general scheme.

Geom-LTM: **Geom**etric derivative-free **L**ocal **T**uning method with the **M**aximum convolution product of local and global estimates of the Lipschitz constant (see, e.g., [40, 41, 46, 49, 50]): see Step 2.3 and Step 3.1 in the general scheme.

Smooth-A*K*: Geometric method constructing **Smooth** auxiliary functions based on the usage of the first derivative information $f'(x)$ and **A** priori estimate of the Lipschitz constant ***K*** for $f'(x)$ (see, e.g., [11, 42, 46, 52]): see Step 2.1 and Step 3.3 in the general scheme.

Smooth-LTM: Geometric derivative-based method constructing **Smooth** auxiliary functions and using the **L**ocal **T**uning technique with the **M**aximum convolution product of local and global estimates of the Lipschitz constant for $f'(x)$ (see, e.g., [40, 41, 46, 49]): see Step 2.3 and Step 3.3 in the general scheme.

As established by a thoroughly developed convergence theory of the reported general scheme (see, e.g., [21, 43, 46, 49, 52]), a suitable value of the reliability parameter r of the considered methods can be always found which guarantees the convergence of trial points only to the global minimizers x^* from (4). General non-convex constraints can be also treated within their general scheme (see, e.g., [20, 48, 52]). These methods can be generalized to the multidimensional case by different approaches (see, e.g., [12, 23, 37, 38, 44, 46, 52, 55, 58, 60]) that allow their usage in the study of the general multiparametric identification problem (1)–(3).

Numerical Experiments

In order to estimate the behavior of the Lipschitz global optimization methods described in the previous section "Univariate Lipschitz Global Optimization Methods" on the problem instances from section "Case Study", some widely used metaheuristic algorithms were taken for the comparison, namely: **D**ifferential **E**volution (DE), **P**article **S**warm **O**ptimization (PSO), **A**rtificial **B**ee **C**olony (ABC), and **F**ire**F**ly (FF) algorithms (see, e.g., [24, 28] for their detailed description).

Parameters of these methods were chosen as follows (see, e.g., [24, 28]; all the parameters were thoroughly studied to respect the multiextremal character of the objective functions, as suggested in the literature). For the DE algorithm, the differential weight F was set equal to 0.7; the crossover rate CR was set equal to 0.5; and the mutation strategy DE/rand/1/exp was used as one of the most powerful strategies. For the PSO algorithm, the static inertia weight ω was set equal to 0.6; the cognitive ϕ_l and social ϕ_g parameters were set both equal to 2.0; and the velocity $v_{\max}$ was set equal to 15 % of the search interval. For the ABC method, the number of the employed bees was set equal to the number of the onlooker bees and was equal to half population; the number of scout bees was set equal to 1; and the *limit* parameter was set equal to the number of food sources (candidate solutions) as half population. Finally, for the FF algorithm, the randomization parameter α was set equal to $0.005(b-a)$; the absorption coefficient γ was set equal to $0.01/\sqrt{b-a}$; and the attractiveness parameter β_0 was set equal to 1.

The reliability parameter r of the considered Lipschitz global optimization methods (except the methods with a priori given Lipschitz constants) was set close to 1 (namely, 1.1) for the geometric methods Geom-GL, Geom-LTM, Smooth-AK, and Smooth-LTM and to 2.0 for the Inf-GL method, as recommended by the convergence study of these algorithms.

For all metaheuristic algorithms the population size was set equal to 10, the number of runs was set equal to 100 (i.e., each metaheuristic algorithm was launched 100 times for a given problem), and for all the methods the maximal number of trials was set equal to 10,000. Since the metaheuristic methods do not have any internal stopping criterion, each of their run was arrested when a trial point in an ε-neighborhood of the global minimizer x^* (known for the considered benchmark instances) was generated. In what follows, such a stopping criterion will be termed as the first-successful-point stopping criterion.

The results of the experiments are presented in Tables 2, 3, 4, and 5. Particularly, the numbers of trials generated by the considered Lipschitz global optimization methods when using their internal stopping criterion (aiming at the global optimality certification) with different accuracy coefficients ε are given in Tables 2 and 3. It can be seen from Tables 2 and 3 that the usage of the local information (as in the methods Geom-LTM, Smooth-AK, and Smooth-LTM) allowed us to solve the problems by executing less trials with respect to the methods that used in their work only global information (as the Geom-AL, Geom-GL, and Inf-GL methods). Moreover, the local tuning methods were less sensitive (in terms of the performed trials) to increasing

Table 2 Number of trials for the considered Lipschitz methods stopped by their internal stopping criterion (8) with $\varepsilon = 10^{-4}$

Func	Geom-A*L*	Geom-G*L*	Inf-G*L*	Geom-LTM	Smooth-A*K*	Smooth-LTM
$(10,0)$	109	83	72	45	26	21
$(50,0)$	149	130	116	80	128	65
$(100,0)$	273	173	185	150	250	128
$(10,9)$	113	87	111	50	23	17
$(50,9)$	199	191	188	145	116	84
$(100,9)$	329	315	323	290	149	138
Avg	195.3	163.2	165.8	126.7	115.3	77.5

Table 3 Number of trials for the considered Lipschitz methods stopped by their internal stopping criterion (8) with $\varepsilon = 10^{-6}$

Func	Geom-A*L*	Geom-G*L*	Inf-G*L*	Geom-LTM	Smooth-A*K*	Smooth-LTM
$(10,0)$	677	535	668	66	32	23
$(50,0)$	433	395	353	101	135	67
$(100,0)$	452	352	385	174	256	129
$(10,9)$	845	763	730	73	28	21
$(50,9)$	513	501	424	165	120	85
$(100,9)$	545	511	516	314	153	141
Avg	577.5	509.5	512.7	148.8	120.7	77.7

the accuracy (compare Tables 2 and 3). In this context, it would be interesting to notice that the functions based on $T = 50$ and $T = 100$ observations were solved (in the case of a higher precision $\varepsilon = 10^{-6}$; see Table 3) faster by the three methods Geom-A*L*, Geom-G*L*, and Inf-G*L* than the functions with the smaller observations number $T = 10$. This happened because the functions $(50, \sigma^2)$ and $(100, \sigma^2)$ had the global minimum values much smaller than the mean function values (proportional to T) and, thus, the functions $(10, \sigma^2)$ were more difficult for the methods using only global information during the search. It should be also noted that due to extremely high values of the Lipschitz constants K for $f'(x)$ in our case study (see Table 1 in section "Case Study"), the first derivative information gave less advantage with respect to the situations when more regular objective functions are minimized.

Results of the numerical comparison of all the methods when using the first-successful-point criterion with different accuracy coefficients ε are shown in Tables 4 and 5, where all the average values for the metaheuristic algorithms were calculated without taking into consideration the failed runs. Even in this condition advantageous for the metaheuristics, their performance was worse than that of the Lipschitz-based methods in our case study.

The distribution of trial points generated by all the methods when minimizing the functions $(50,0)$ ($T = 50$ noise-free observations) and $(50,9)$ ($T = 50$ noisy

Table 4 Number of trials for the considered Lipschitz and metaheuristic methods stopped by the first-successful-point stopping criteria with $\varepsilon = 10^{-4}$

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM	DE		PSO		ABC		FF	
							Avg	Fails (%)	Avg	Fails (%)	Avg	Fails (%)	Avg	Fails (%)
(10,0)	55	27	35	34	22	16	254.1	0	187.0	0	548.3	0	188.2	0
(50,0)	130	109	87	44	71	63	472.0	0	433.4	0	1493.2	0	277.2	0
(100,0)	130	134	136	135	220	100	635.7	0	669.6	7	1456.4	1	410.3	0
(10,9)	60	58	54	40	19	15	258.4	4	232.1	0	338.8	0	183.0	7
(50,9)	153	156	159	129	112	73	658.0	20	541.5	52	2668.7	8	853.7	0
(100,9)	251	240	195	262	146	130	769.2	16	737.1	37	2722.2	1	1133.7	0
Avg	129.8	120.7	111.0	107.3	98.3	66.2	507.9	6.7	466.8	16.0	1537.9	1.7	507.7	1.2

Table 5 Number of trials for the considered Lipschitz and metaheuristic methods stopped by the first-successful-point stopping criteria with $\varepsilon = 10^{-6}$

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM	DE		PSO		ABC		FF	
							Avg	Fails (%)	Avg	Fails (%)	Avg	Fails (%)	Avg	Fails (%)
$(10,0)$	145	199	131	59	29	20	384.9	0	479.9	0	3620.6	20	4088.6	23
$(50,0)$	277	228	151	97	132	65	615.7	0	607.1	0	5850.5	84	4045.1	20
$(100,0)$	342	162	221	169	253	127	798.2	0	717.3	8	5035.1	90	4239.3	25
$(10,9)$	386	281	85	60	25	19	399.4	4	528.8	0	1878.7	0	3701.1	27
$(50,9)$	293	221	202	157	118	83	859.1	20	860.6	52	4330.7	97	3866.7	63
$(100,9)$	386	298	406	309	150	138	1011.4	16	1053.4	38	4956.0	89	5538.3	74
Avg	304.8	231.5	199.3	141.8	117.8	75.3	678.1	6.7	707.9	16.3	4278.6	63.3	4246.5	38.7

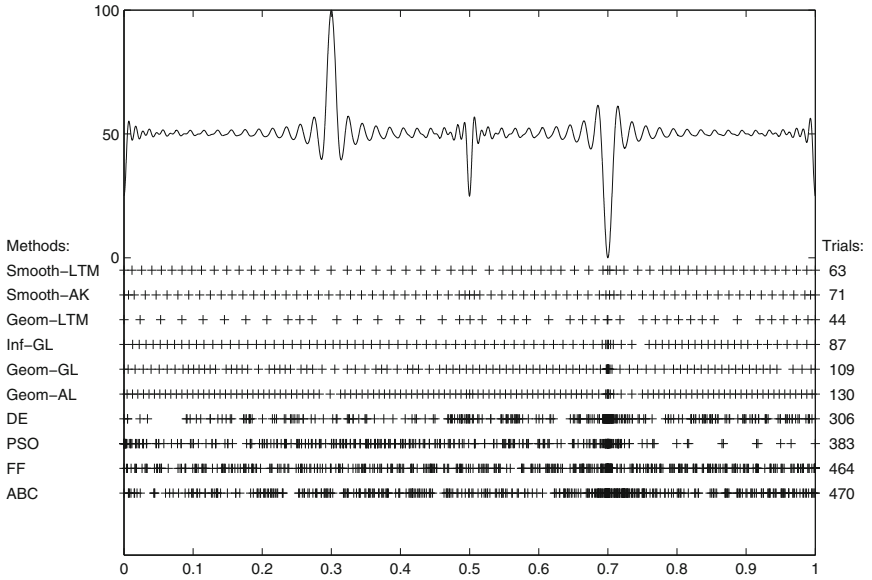

Fig. 3 Distribution of trial points generated by the tested methods when minimizing the function $(50,0)$ with the accuracy $\varepsilon = 10^{-4}$ in the first-successful-point stopping criterion

observations) with the first-successful-point stopping criterion and two different accuracy coefficients $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$ is illustrated in Figs. 3 and 4 (for $\varepsilon = 10^{-4}$) and Figs. 5 and 6 (for $\varepsilon = 10^{-6}$), respectively.

Conclusions

As it follows from the results of numerical experiments, the performance of Lipschitz global optimization methods seems to be very promising with respect to the considered metaheuristic algorithms in our case study. Moreover, the Lipschitz-based algorithms give the possibility to obtain the solution to the studied problem with some guaranteed gap, which is important from the practical point of view. The usage of local information in these methods (as in the local tuning Geom-LTM algorithm and in the Smooth-AK and Smooth-LTM algorithms with the usage of the first derivative information) can be a subsequent step in the increasing efficiency of the search for the optimal parameters. As it has been recently shown, there exist local improvement techniques (see, e.g., [13, 30, 49, 50]) that can produce even more acceleration effect on the considered methods.

Acknowledgements This work was supported by the Russian Science Foundation, project number 15-11-30022 "Global optimization, supercomputing computations, and applications."

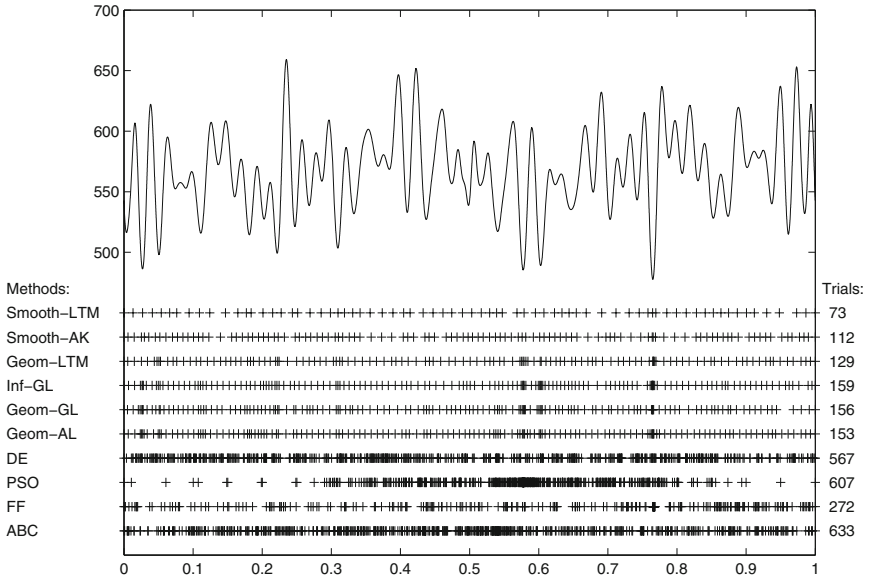

Fig. 4 Distribution of trial points generated by the tested methods when minimizing the function $(50,9)$ with the accuracy $\varepsilon = 10^{-4}$ in the first-successful-point stopping criterion

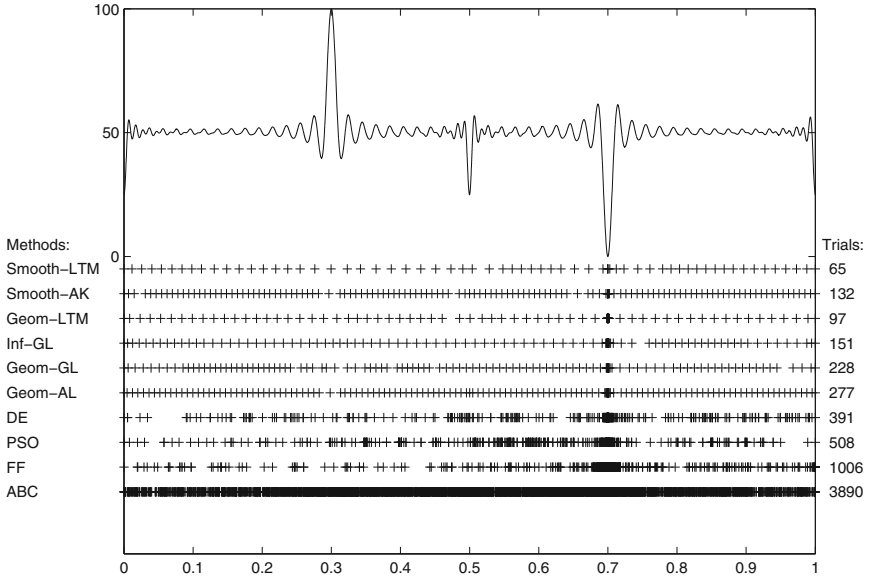

Fig. 5 Distribution of trial points generated by the tested methods when minimizing the function $(T = 50, \sigma^2 = 0)$ with the accuracy $\varepsilon = 10^{-6}$

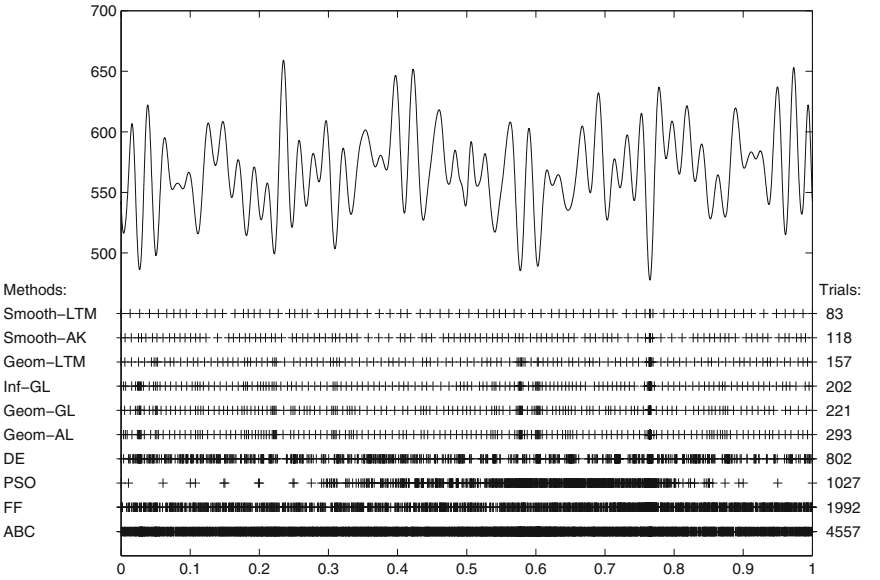

Fig. 6 Distribution of trial points generated by the tested methods when minimizing the function $(T = 50, \sigma^2 = 9)$ with the accuracy $\varepsilon = 10^{-6}$

References

1. Barkalov, K., Polovinkin, A., Meyerov, I., Sidorov, S., Zolotykh, N.: SVM regression parameters optimization using parallel global search algorithm. In: Parallel Computing Technologies. LNCS, vol. 7979, pp. 154–166. Springer, Heidelberg (2013)
2. Bloomfield, P.: Fourier Analysis of Time Series: An Introduction. Wiley, New York (2000)
3. Broomhead, D.S., King, G.P.: Extracting qualitative dynamics from experimental data. Phys. D Nonlinear Phenom. **20**(2–3), 217–236 (1986)
4. Calvin, J.M., Žilinskas, A.: One-dimensional global optimization for observations with noise. Comput. Math. Appl. **50**(1–2), 157–169 (2005)
5. Carnì, D.L., Fedele, G.: Multi-sine fitting algorithm enhancement for sinusoidal signal characterization. Comput. Stand. Interfaces **34**(6), 535–540 (2012)
6. Costanzo, S.: Synthesis of multi-step coplanar waveguide-to-microstrip transition. Prog. Electromagn. Res. **113**, 111–126 (2011)
7. Elsner, J.B., Tsonis, A.A.: Singular Spectrum Analysis: A New Tool in Time Series Analysis. Springer, New York (1996)
8. Evtushenko, Y.G.: Numerical Optimization Techniques. Translations Series in Mathematics and Engineering. Springer, Berlin (1985)
9. Fedele, G., Ferrise, A.: A frequency-locked-loop filter for biased multi-sinusoidal estimation. IEEE Trans. Signal Process. **62**(5), 1125–1134 (2014)
10. Garnier, H., Wang, L. (eds.): Identification of Continuous-Time Models from Sampled Data. Springer, London (2008)
11. Gergel, V.P., Sergeyev, Y.D.: Sequential and parallel algorithms for global minimizing functions with Lipschitzian derivatives. Comput. Math. Appl. **37**(4–5), 163–179 (1999)

12. Gergel, V.P., Grishagin, V.A., Gergel, A.V.: Adaptive nested optimization scheme for multidimensional global search. J. Glob. Optim. (2015, to appear). doi 10.1007/s10898-015-0355-7
13. Gergel, V.P., Grishagin, V.A., Israfilov, R.A.: Local tuning in nested scheme of global optimization. Proc. Comput. Sci. **51**, 865–874 (2015). (International Conference on Computational Science ICCS 2015 – Computational Science at the Gates of Nature)
14. Gillard, J.W.: Cadzow's basic algorithm, alternating projections and singular spectrum analysis. Stat. Interface **3**(3), 335–343 (2010)
15. Gillard, J.W., Kvasov, D.E.: Lipschitz optimization methods for fitting a sum of damped sinusoids to a series of observations. Stat. Interface (2016, to appear)
16. Gillard, J.W., Zhigljavsky, A.: Analysis of structured low rank approximation as an optimisation problem. Informatica **22**(4), 489–505 (2011)
17. Gillard, J.W., Zhigljavsky, A.: Optimization challenges in the structured low rank approximation problem. J. Glob. Optim. **57**(3), 733–751 (2013)
18. Gillard, J.W., Zhigljavsky, A.: Stochastic algorithms for solving structured low-rank matrix approximation problems. Commun. Nonlinear Sci. Numer. Simul. **21**, 70–88 (2015)
19. Golyandina, N., Nekrutkin, V., Zhigljavsky, A.: Analysis of Time Series Structure: SSA and Related Techniques. Chapman & Hall/CRC, Boca Raton (2001)
20. Grishagin, V.A., Strongin, R.G.: Optimization of multi-extremal functions subject to monotonically unimodal constraints. Eng. Cybern. **22**(5), 117–122 (1984)
21. Grishagin, V.A., Sergeyev, Y.D., Strongin, R.G.: Parallel characteristic algorithms for solving problems of global optimization. J. Glob. Optim. **10**(2), 185–206 (1997)
22. Holmström, K., Petersson, J.: A review of the parameter estimation problem of fitting positive exponential sums to empirical data. Appl. Math. Comput. **126**(1), 31–61 (2002)
23. Kvasov, D.E.: Diagonal numerical methods for solving Lipschitz global optimization problems. Boll. Unione Mat. Ital. **I (Serie IX)**(3), 857–871 (2008)
24. Kvasov, D.E., Mukhametzhanov, M.S.: One-dimensional global search: nature-inspired vs. Lipschitz methods. In: Proceedings of the ICNAAM2015 Conference, AIP Conference Proceedings. AIP Publishing LLC, New York (2015).
25. Kvasov, D.E., Sergeyev, Y.D.: Univariate geometric Lipschitz global optimization algorithms. Numer. Algebra Control Optim. **2**(1), 69–90 (2012)
26. Kvasov, D.E., Sergeyev, Y.D.: Lipschitz global optimization methods in control problems. Autom. Remote Control **74**(9), 1435–1448 (2013)
27. Kvasov, D.E., Sergeyev, Y.D.: Deterministic approaches for solving practical black-box global optimization problems. Adv. Eng. Softw. **80**, 58–66 (2015)
28. Kvasov, D.E., Mukhametzhanov, M.S., Sergeyev, Y.D.: Solving univariate global optimization problems by nature-inspired and deterministic algorithms. Adv. Eng. Softw. (2015, submitted)
29. Lemmerling, P., Van Huffel, S.: Analysis of the structured total least squares problem for Hankel/Toeplitz matrices. Numer. Algorithms **27**(1), 89–114 (2001)
30. Lera, D., Sergeyev, Y.D.: Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives. SIAM J. Optim. **23**(1), 508–529 (2013)
31. Li, Y., Liu, K., Razavilar, J.: A parameter estimation scheme for damped sinusoidal signals based on low-rank Hankel approximation. IEEE Trans. Signal Process. **45**(2), 481–486 (1997)
32. Liuzzi, G., Lucidi, S., Piccialli, V.: A partition-based global optimization algorithm. J. Glob. Optim. **48**(1), 113–128 (2010)
33. Markovsky, I.: Bibliography on total least squares and related methods. Stat. Interface **3**(3), 329–334 (2010)
34. Mockus, J.: Bayesian Approach to Global Optimization. Kluwer Academic, Dordrecht (1989)
35. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. SpringerBriefs in Optimization. Springer, New York (2014)
36. Paulavičius, R., Žilinskas, J.: Simplicial Lipschitz optimization without the Lipschitz constant. J. Glob. Optim. **59**(1), 23–40 (2014)
37. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Glob. Optim. **59**(2–3), 545–567 (2014)

38. Pintér, J.D.: Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. Kluwer Academic, Dordrecht (1996)
39. Pollock, D.: A Handbook of Time Series Analysis, Signal Processing, and Dynamics. Academic, London (1999)
40. Sergeyev, Y.D.: An information global optimization algorithm with local tuning. SIAM J. Optim. **5**(4), 858–870 (1995)
41. Sergeyev, Y.D.: A one-dimensional deterministic global minimization algorithm. Comput. Math. Math. Phys. **35**(5), 705–717 (1995)
42. Sergeyev, Y.D.: Global one-dimensional optimization using smooth auxiliary functions. Math. Program. **81**(1), 127–146 (1998)
43. Sergeyev, Y.D.: On convergence of "Divide the Best" global optimization algorithms. Optimization **44**(3), 303–325 (1998)
44. Sergeyev, Y.D.: Multidimensional global optimization using the first derivatives. Comput. Math. Math. Phys. **39**(5), 711–720 (1999)
45. Sergeyev, Y.D., Grishagin, V.A.: A parallel method for finding the global minimum of univariate functions. J. Optim. Theory Appl. **80**(3), 513–536 (1994)
46. Sergeyev, Y.D., Kvasov, D.E.: Diagonal Global Optimization Methods. FizMatLit, Moscow (2008) [in Russian]
47. Sergeyev, Y.D., Kvasov, D.E.: Lipschitz global optimization. In: Cochran, J.J. (ed.) Wiley Encyclopedia of Operations Research and Management Science, vol. 4, pp. 2812–2828. Wiley, New York (2011)
48. Sergeyev, Y.D., Khalaf, F.M.H., Kvasov, D.E.: Univariate algorithms for solving global optimization problems with multiextremal non-differentiable constraints. In: A. Törn, J. Žilinskas (eds.) Models and Algorithms for Global Optimization, pp. 123–140. Springer, Heidelberg (2007)
49. Sergeyev, Y.D., Strongin, R.G., Lera, D.: Introduction to Global Optimization Exploiting Space-Filling Curves. SpringerBriefs in Optimization. Springer, New York (2013)
50. Sergeyev, Y.D., Mukhametzhanov, M.S., Kvasov, D.E., Lera, D.: Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization. J. Optim. Theory Appl. (2016, to appear)
51. Strekalovsky, A.S.: Elements of Nonconvex Optimization. Nauka, Novosibirsk (2003) [in Russian]
52. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic, Dordrecht (2000). 3rd edn. by Springer, Berlin (2014)
53. Törn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989)
54. Van Huffel, S., Vandewalle, J.: The Total Least Squares Problem: Computational Aspects and Analysis. SIAM, Philadelphia (1991)
55. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer, New York (2008)
56. Žilinskas, A.: Global Optimization. Axiomatics of Statistical Models, Algorithms, and Applications. Mokslas, Vilnius (1986) [in Russian]
57. Žilinskas, A.: On similarities between two models of global optimization: statistical models and radial basis functions. J. Glob. Optim. **48**(1), 173–182 (2010)
58. Žilinskas, A., Žilinskas, J.: Global optimization based on a statistical model and simplicial partitioning. Comput. Math. Appl. **44**(7), 957–967 (2002)
59. Žilinskas, A., Žilinskas, J.: Interval arithmetic based optimization in nonlinear regression. Informatica **21**(1), 149–158 (2010)
60. Žilinskas, A., Žilinskas, J.: A hybrid global optimization algorithm for non-linear least squares regression. J. Glob. Optim. **56**(2), 265–277 (2013)

Part III
Multi-Objective Global Optimization

A Multicriteria Generalization of Bayesian Global Optimization

Michael Emmerich, Kaifeng Yang, André Deutz, Hao Wang, and Carlos M. Fonseca

Abstract This chapter discusses a generalization of the expected improvement used in Bayesian global optimization to the multicriteria optimization domain, where the goal is to find an approximation to the Pareto front. The expected hypervolume improvement (EHVI) measures improvement as the gain in dominated hypervolume relative to a given approximation to the Pareto front. We will review known properties of the EHVI, applications in practice and propose a new exact algorithm for computing EHVI. The new algorithm has asymptotically optimal time complexity $O(n \log n)$. This improves existing computation schemes by a factor of $n/\log n$. It shows that this measure, at least for a small number of objective functions, is as fast as other simpler measures of multicriteria expected improvement that were considered in recent years.

Keywords Bayesian Global Optimization • Expected Hypervolume Improvement • Computation Complexity

Introduction

In the 1970s several new ideas for global optimization were proposed. Among these the idea of *Bayesian Global Optimization* (BGO) was proposed by the Lithuanian research group Jonas Mockus and Antanas Žilinskas [14, 15, 21, 25]. It had a lasting impact on the development of both deterministic and stochastic global optimization techniques. Today variations of this idea are known under various names, such as

M. Emmerich • K. Yang • A. Deutz (✉) • H. Wang
Multiobjective Optimization and Decision Analysis (MODA) Research Group, LIACS, Faculty of Science, Leiden University, Leiden, The Netherlands
e-mail: m.t.m.emmerich@liacs.leidenuniv.nl; k.yang@liacs.leidenuniv.nl; a.h.deutz@liacs.leidenuniv.nl; h.wang@liacs.leidenuniv.nl
http://moda.liacs.nl

C.M. Fonseca
CISUC, Department of Informatics Engineering, University of Coimbra, Polo II, Pinhal de Marrocos, P-3030 290 Coimbra, Portugal
e-mail: cmfonsec@dei.uc.pt

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_12

Efficient Global Optimization [9] or *Expected Improvement Algorithm* [22]. In these techniques the goal is to find the extremum of a function $f : \mathscr{X} \to \mathbb{R}$ where $\mathscr{X}$ is a compact subspace of $\mathbb{R}^d$. BGO assumes that the objective function is the realization of a Gaussian random field. This random field can be conditioned by the knowledge of $f(\mathbf{x}^{(i)})$ at some points $\mathbf{x}^{(i)} \in \mathscr{X}, i = 1, \ldots, n$. Under this assumption, measures such as the expected improvement of a new design point are well defined, and can be used to guide search towards the global optimum.

In this chapter we describe a generalization of this approach to multicriteria optimization. It iteratively evaluates points from $\mathscr{X}$ and finds a well distributed subset of the Pareto front of a multicriteria optimization problem. The algorithm is based on a generalization of the expected improvement, which is based on the hypervolume indicator, the so-called *Expected Hypervolume Improvement* (EHVI) [3]. It has attractive theoretical properties [23], but so far its computation time was considered to be expensive. In this chapter it is shown that, for bicriteria optimization, a fast algorithm exists for computing EHVI that has only linear time complexity in the size of the intermediate approximation to the Pareto front, given that the Pareto front is given as a sorted set. It is shown that this algorithm has asymptotically optimal time complexity.

This chapter is organized as follows: section "Bayesian Global Optimization" introduces the framework of BGO. Section "Multicriteria Optimization" shows how this framework can be generalized to multicriteria optimization. Section "Expected Hypervolume Improvement" defines the EHVI, discusses some of its theoretical properties, and reviews recent applications of it. Section "Efficient Exact Computation" outlines the new, asymptotically efficient algorithm for its exact computation and proves that it has an asymptotically optimal time complexity for bicriteria problems. A numerical example is discussed in section "Numerical Example". Section "Application Notes and Further Reading" points to some recent applications and related work. Finally, section "Summary and Outlook", concludes with a summary and discusses open questions.

Bayesian Global Optimization

In BGO the goal is to solve d-dimensional global optimization problems of the type: Find $\mathbf{x}^*$ with

$$\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathscr{X}} f(\mathbf{x}), \mathscr{X} = [\mathbf{x}_{min}, \mathbf{x}_{max}] \subset \mathbb{R}^d \tag{1}$$

(Without loss of generality we consider minimization only.)

In order to do so, a sequence $\{\mathbf{x}^{(t)}\}_{t=1,2,\ldots}$ of points is computed such that

$$\mathbf{x}^{(t)} \in \arg\max_{\mathbf{x} \in S} (\mathrm{E}(I(\mathbf{x}) | (\mathbf{x}^{(1)}, f(\mathbf{x}^{(1)}), \ldots, (\mathbf{x}^{(t-1)}, f(\mathbf{x}^{(t-1)})) \tag{2}$$

Here $\mathrm{E}(I(\mathbf{x}) | (\mathbf{x}^{(1)}, f(\mathbf{x}^{(1)}), \ldots, (\mathbf{x}^{(t-1)}, f(\mathbf{x}^{(t-1)})$ denotes the expected improvement measure that measures how promising the new point $\mathbf{x}$ is, given $t-1$ previous

evaluations of f at $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(t-1)}$. This expected improvement is an expected value of a random variable, here called $I(\mathbf{x})$ that requires further explanation.

In BGO one makes the assumption that the function f is the realization of a Gaussian random field $\mathbf{F}$. A Gaussian random field is an infinite set of random variables. Each random variable in F is identified by its spatial index $\mathbf{x} \in \mathbb{R}^d$. We will denote it with $\mathbf{F}_\mathbf{x}$. It is assumed that the random variables share the same global mean value β and global variance s^2. Moreover, a correlation $\rho(\mathbf{F}_\mathbf{u},\mathbf{F}_\mathbf{v})$ is defined for every pair of indices $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$. This correlation depends on the relation between $\mathbf{u}$ and $\mathbf{v}$. A typical family of correlation functions is

$$\rho(\mathbf{F}_\mathbf{u},\mathbf{F}_\mathbf{v}) = \exp\left(-\sum_{i=1}^{d}\theta_i|u_i - v_i|^{q_i}\right)$$

It is important that this correlation function is positive definite. It obtains the value of 1, if $\mathbf{v} = \mathbf{u}$ and gets smaller with increasing distance between $\mathbf{v}$ and $\mathbf{u}$. The parameters q_i and θ_i are either set by the user or obtained from data fitting. The parameters θ_i are positive.

The Gaussian random field can be viewed as a multivariate Gaussian distribution of infinite dimension. We can use well-known expressions for the marginal distributions of the multivariate distribution to find the conditional distribution, given that some of the realizations of one dimensional random variables are known. That is, given the prior information $\mathbf{F}_{\mathbf{x}^{(1)}} = f(\mathbf{x}^{(1)}),\ldots,\mathbf{F}_{\mathbf{x}^{(t-1)}} = f(\mathbf{x}^{(t-1)})$ we can compute the parameters μ (conditional mean) and σ^2 (conditional variance) of the conditioned random variable:

$$\mathbf{F}_\mathbf{x} \mid \mathbf{F}_{\mathbf{x}^{(1)}} = f(\mathbf{x}^{(1)}),\ldots,\mathbf{F}_{\mathbf{x}^{(t-1)}} = f(\mathbf{x}^{(t-1)}) \tag{3}$$

As a shortcut we will denote this random variable with $\mathbf{F}_\mathbf{x} \mid \mathbf{X}, f(\mathbf{X})$, where $\mathbf{X} = (\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(t-1)})$ denote the indices for which we know realizations and the values of the corresponding realizations are abbreviated with

$$f(\mathbf{X}) = \left(f(\mathbf{x}^{(1)}),\ldots,f(\mathbf{x}^{(t-1)})\right)$$

The estimation of hyperparameters θ_i and q_i, $i = 1,\ldots,d$, of the correlation function, as well as the global variance and mean can be accomplished by maximum likelihood methods. For details on the computations of the parameters of the conditional distribution we refer to the specialized literature [19].

Now, the expected improvement can be defined: The improvement of a function value $y \in \mathbb{R}$ is defined as

$$\mathrm{I}(y) = \max\{0, y_{min} - y)\} \tag{4}$$

where $y_{min} = \min\{f(\mathbf{x}^{(1)}),\ldots,f(\mathbf{x}^{(t-1)})\}$. Then the *expected improvement* is defined as

$$\mathrm{E}(\mathrm{I}(\mathbf{F}_{\mathbf{x}}|\mathbf{X},f(\mathbf{X})) = \int_{y=-\infty}^{y_{min}} \mathrm{I}(y)\mathrm{PDF}_{\mathbf{x}|\mathbf{X},f(\mathbf{X})}(y)\mathrm{d}y \tag{5}$$

Here $\mathrm{PDF}_{\mathbf{x}|\mathbf{X},f(\mathbf{X})}$ is the probability density function of $\mathbf{F}_{\mathbf{x}}\,|\mathbf{X},f(\mathbf{X})$.

Multicriteria Optimization

A continuous m-dimensional multicriteria optimization problem is a problem where multiple objective functions, say $f_1 : \mathscr{X} \to \mathbb{R}, \ldots, f_m : \mathscr{X}^m \to \mathbb{R}$, are to be minimized simultaneously, $\mathscr{X} \subseteq \mathbb{R}^m$.

In the a posteriori approach to multicriteria optimization an approximation to the Pareto front of the problem is computed first. Based on this, the trade-off is analyzed and a solution is selected by the decision maker. To define a Pareto front, we introduce the Pareto dominance order $\prec$ on $\mathbb{R}^m$, with $\forall \mathbf{y},\mathbf{z} \in \mathbb{R}^m : \mathbf{y} \prec \mathbf{z} \Leftrightarrow (\forall i \in \{1,\ldots,m\} : y_i \leq z_i)$ and $\mathbf{y} \neq \mathbf{z}$. The non-dominated subset of a multiset of vectors $\mathrm{Y} = \{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(m)}\}$ is defined as $\mathrm{nd}(\mathrm{Y}) = \{\mathbf{y} \in \mathrm{Y} \mid \nexists \mathbf{z} \in \mathrm{Y} : \mathbf{z} \prec \mathbf{y}\}$. Given a multicriteria optimization problem, the image of $\mathscr{X}$ is defined as $\mathscr{Y} = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathscr{X}\}$. The Pareto front of a multicriteria optimization problem is defined as $\mathscr{Y}_{\mathrm{nd}} := \mathrm{nd}(\mathscr{Y})$. An important special case is bicriteria optimization, where $m = 2$.

One way to generalize the BGO algorithm is to compute the expected improvement of the hypervolume indicator. The hypervolume indicator is the m-dimensional Lebesgue measure λ_m of the dominated subspace limited from above by some reference point $\mathbf{r}^m$. More precisely the hypervolume indicator is defined as

$$\mathrm{hv}(\mathrm{Y}) = \lambda_m\left(\{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{z} \in \mathrm{Y} : \mathbf{z} \prec \mathbf{y} \wedge \mathbf{y} \prec \mathbf{r}\}\right) = \lambda_m\left(\bigcup_{\mathbf{y}\in\mathrm{Y}} [\mathbf{y},\mathbf{r}]\right) \tag{6}$$

In Fig. 1 the hypervolume indicator is illustrated for a Pareto front approximation with nine points and two objective functions ($m = 2$). Given a problem with a Pareto front bounded above by the reference point, sets that maximize the hypervolume indicator are well distributed subsets of the Pareto front [1]. This is why finding the Pareto front is sometimes recast as the problem of maximizing the hypervolume indicator over the set of all subsets of $\mathscr{X}$. We will call this problem *hypervolume maximization*.

Expected Hypervolume Improvement

For hypervolume maximization problems the generalization of the improvement function is straightforward. We generalize the best solution found up to iteration $t-1$, namely $y_{min,t-1}$, to

$$\mathrm{Y}_{\mathrm{nd},t-1} = \mathrm{nd}\left(\{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(t-1)}\}\right) \tag{7}$$

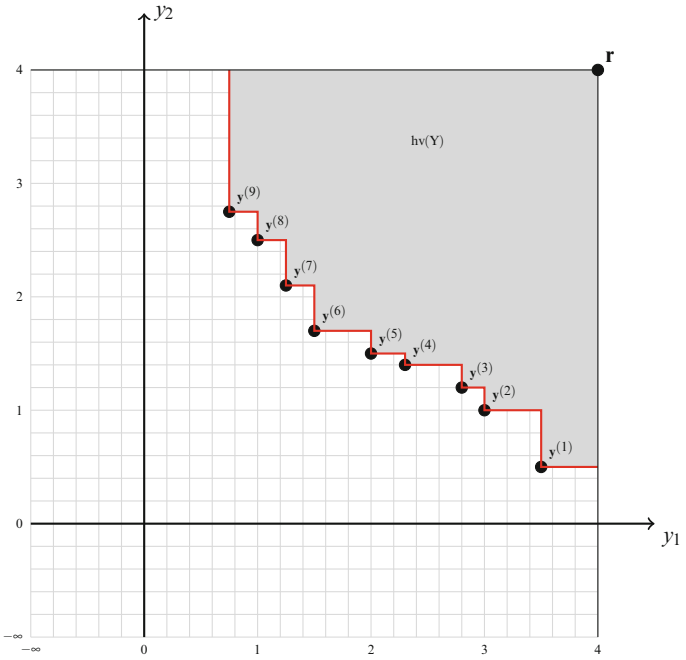

Fig. 1 Hypervolume indicator of Pareto front approximation

The improvement function is generalized by the following definition of an m-dimensional improvement:

$$\mathrm{I}_m(\mathbf{y}, \mathrm{Y}_{\mathrm{nd},t-1}, \mathbf{r}) := \mathrm{hv}\left(\mathrm{Y}_{\mathrm{nd},t-1} \cup \{\mathbf{f}(\mathbf{x})\}\right) - \mathrm{hv}(\mathrm{Y}_{\mathrm{nd},t-1}) \tag{8}$$

It is easy to show that this I_m specializes to the improvement function in one dimension, if we chose r_1 to be sufficiently large.

In order to compute the *expected* improvement in the multicriteria case, we need also to generalize the assumption on the Gaussian random field. For this, we consider one Gaussian random field per objective function and assume that there is no correlation between random variables from different random fields. For every point $\mathbf{x} \in \mathscr{X}$ we obtain an m-dimensional random variable conditioned on previous information, that is given by $\mathbf{X}$ and $\mathbf{f}(\mathbf{X}) = (\mathbf{f}(\mathbf{x}^{(1)}, \ldots, \mathbf{f}(\mathbf{x}^{(t-1)})$.

The resulting EHVI can be denoted with

$$\mathrm{E}\left(\mathrm{I}_m\left((F_1(\mathbf{x}), \ldots, F_m(\mathbf{x}))\right) \mid \mathbf{X}, \mathbf{f}(\mathbf{X}), \mathrm{Y}_{\mathrm{nd},t-1}, \mathbf{r}\right) =$$

$$\int_{\mathbf{y} \in \mathbb{R}^{\mathbf{m}}} \mathrm{I}_m(\mathbf{y}, \mathrm{Y}_{\mathrm{nd},t-1}, \mathbf{r}) \mathrm{PDF}_{\mathbf{x}|\mathbf{X},\mathbf{f}(\mathbf{X})}(\mathbf{y}) \mathrm{d}\mathbf{y} \tag{9}$$

and it is a generalization of the single objective expected improvement, if we consider $y_{\min,0} = r_1$.

Efficient Exact Computation

In this section the problem of computing the EHVI is studied and a new, efficient algorithm for bicriteria optimization will be derived. Fast algorithms for computing the EHVI are important, because in BGO a large number evaluations of the EHVI are performed in each iteration when searching for its maximizer. Although the BGO algorithm is typically used in the context of expensive function evaluations, the optimization of the EHVI can significantly contribute to the total running time of the algorithm. For instance, this was recently reported as a major drawback of using EHVI in [12], even when considering only the two dimensional case.

A simplified notation will be used in the following. It focuses only on the elements that are relevant for the EHVI computation.

Symbol	Type	Description
μ	$\mathbb{R}^m$	Mean values of predictive distribution
σ	$(\mathbb{R}_0^+)^m$	Standard deviations of predictive distribution
Y	$(\mathbb{R}^m)^n$	Sequence of mutually non-dominated points (Pareto front approximation in $t-1$)
$\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(n)}$	$\mathbb{R}^m$	The vectors in Y
$\mathbf{r}$	$\mathbb{R}^m$	Reference point

For computing integrals of the expected improvement it is useful to define the function Δ. For a given vector of objective function values $\mathbf{y} \in \mathbb{R}^m$, $\Delta(\mathbf{y}, \mathrm{Y}, \mathrm{r})$ is the subset of the vectors in $\mathbb{R}^m$ which are exclusively dominated by a vector $\mathbf{y}$ and not by elements in Y and that dominate the reference point, in symbols

$$\Delta(\mathbf{y}, \mathrm{Y}, \mathbf{r}) = \lambda_m\{\mathbf{z} \in \mathbb{R} \mid \mathbf{y} \prec \mathbf{z} \text{ and } \mathbf{z} \prec \mathbf{r} \text{ and } \nexists \mathbf{q} \in \mathrm{Y} : \mathbf{q} \prec \mathbf{z}\} \tag{10}$$

In order to simplify notation, we will write $\Delta(\mathbf{y})$ whenever $\mathrm{Y}, \mathbf{r}$ are given by the context.

Based on this, we can now concisely (re-)define the EHVI function as

$$\mathrm{EHVI}(\mu, \sigma, \mathrm{Y}, \mathbf{r}) = \int_{y_1=-\infty}^{\infty} \cdots \int_{y_m=-\infty}^{\infty} \lambda_m(\Delta(\mathbf{y}))\mathrm{PDF}_{\mu,\sigma}(\mathbf{y})\mathrm{d}y_1 \ldots \mathrm{d}y_m \tag{11}$$

Example 1. An illustration of the EHVI is displayed in Fig. 2. The light gray area is the dominated subspace of $\mathrm{Y} = \{\mathbf{y}^{(1)} = (3,1)^\top, \mathbf{y}^{(2)} = (2,1.5)^\top, \mathbf{y}^{(3)} = (1,2.5)^\top\}$ cut by the reference point $r = (4,4)^\top$. The bivariate Gaussian distribution has the parameters $\mu_1 = 2$, $\mu_2 = 1.5$, $\sigma_1 = 0.7$, and $\sigma_2 = 0.6$. The PDF of the bivariate Gaussian distribution is indicated as a 3-D plot. Here $\mathbf{y}$ is a sample from this distribution, and the area of improvement relative to Y is indicated by the dark shaded area. The variable y_1 stands for the f_1 value and y_2 for the f_2 value.

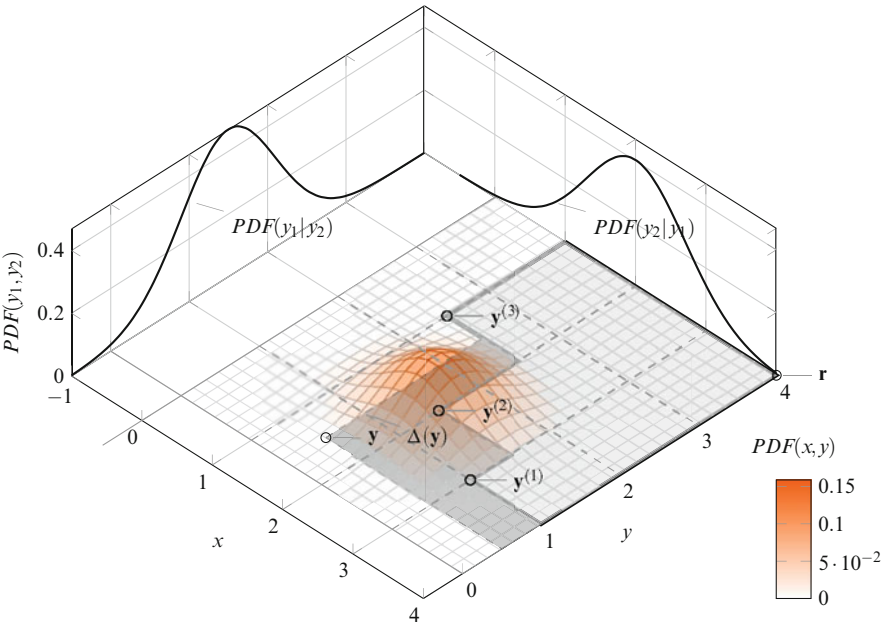

Fig. 2 Expected hypervolume improvement in 2-D (cf. Example 1)

State of the Art

To compute the EHVI (9) Monte Carlo integration is suggested in [3, 4]. Exact algorithms for computing EHVI for $m = 2$ are derived in [5] and for $m > 2$ in [2]. A different algorithm is described in [7].

Fast algorithms have been proposed in [2] and even faster algorithms for $m = 2,3$ in [8]. So far the best known bounds for the time complexity of exact computations are $O(n^2)$ for $m = 2$, and $O(n^3)$ for $m = 3$. It is notable that the number of transcendental function evaluations scales only linearly in n in the algorithm presented in [8]. A lower bound of $\Omega(n \log n)$ is provided for unsorted Y. However, it makes sense to assume that Y is sorted in the first coordinate. In that case, as will be shown, a lower bound of $\Omega(n)$ still holds. None of the algorithms found so far for EHVI reach these lower bounds. In this paper we will present an algorithm for $m = 2$ that does so.

Next an algorithm is outlined that reaches the lower bound time complexity of $\Omega(n \log n)$. We thereby prove that the time complexity of EHVI is $\Theta(n \log n)$. However, this complexity stems from the complexity that is inherent to sorting Y by the first coordinate.

To keep Y sorted in the first coordinate requires an effort of amortized time complexity $O(\log n)$ per iteration. It makes therefore sense to assume a sorted Y. For this case we can show that the time complexity is $\Theta(n)$. To do so, we will first establish a lower bound of $\Omega(n)$ for this case:

Lemma 1. *The computational time complexity of computing the EHVI for a set Y that is sorted by the first coordinate is bounded from below by* $\Omega(n)$.

Proof. An adversary argument can be used to prove this statement. The algorithm has to "look at" all n points. If one point is not used, it could be moved by an adversary and this move will not be noticed by the algorithm; a move of any single point can, in general, change the EHVI. □

Efficient Algorithm

For $m = 2$ the expected improvement can be computed in linear time, given that Y is already sorted by the first coordinate. Next, a formula will be derived that consists of $n+1$ integrals, each of which can be solved in constant time.

The starting point of the derivation is to partition the objective space into $n+1$ disjoint rectangular stripes $S_1, \ldots, S_{n+1}$, as indicated in Fig. 3 (left). In order to define the stripes formally, augment Y with two sentinels: $\mathbf{y}^{(0)} = (r_1, -\infty)$ and $\mathbf{y}^{(n+1)} = (-\infty, r_2)$. The stripes are now defined by

$$S_i = \left(\begin{pmatrix} y_1^{(i)} \\ -\infty \end{pmatrix}, \begin{pmatrix} y_1^{(i-1)} \\ y_2^{(i)} \end{pmatrix} \right), i = 1, \ldots, n+1$$

We can now express the improvement of a point $\mathbf{y} \in \mathbb{R}^2$ by

$$\mathrm{I}_2(\mathbf{y}, \mathrm{Y}, \mathbf{r}) = \sum_{i=1}^{n+1} \lambda_2[S_i \cap \Delta(\mathbf{y})] \tag{12}$$

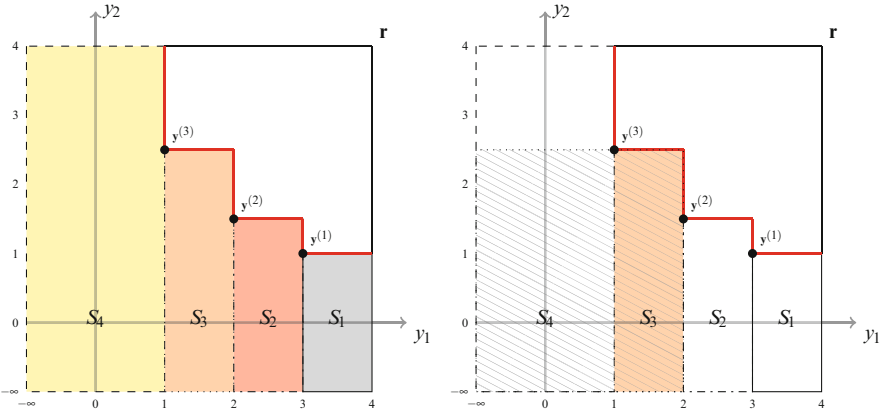

Fig. 3 *Left*: partitioning of the integration region into stripes. *Right*: new partitioning of the reduced integration region after first iteration of the algorithm

This gives rise to the compact integral for the original EHVI, $\mathbf{y} = (y_1, y_2)$:

$$\mathrm{EHVI}(\mu, \sigma, \mathrm{Y}, \mathbf{r}) = \int_{y_1=-\infty}^{\infty} \int_{y_2=-\infty}^{\infty} \sum_{i=1}^{n+1} \lambda_2[S_i \cap \Delta(y_1, y_2)] \cdot PDF_{\mu,\sigma}(\mathbf{y}) d\mathbf{y} \tag{13}$$

It is observed that the intersection of S_i with $\Delta(y_1, y_2)$ is non-empty if and only if $\mathbf{y} = (y_1, y_2)$ dominates the upper right corner of S_i. In other words, if and only if $\mathbf{y}$ is located in the rectangle with lower left corner $(-\infty, -\infty)$ and upper right corner $(y_1^{(i-1)}, y_2^{(i)})$. See Fig. 3 (right) for an illustration. Therefore

$$\mathrm{EHVI}(\mu, \sigma, \mathrm{Y}, \mathbf{r}) = \sum_{i=1}^{n+1} \int_{y_1=-\infty}^{y_1^{(i-1)}} \int_{y_2=-\infty}^{y_2^{(i)}} \lambda_2[S_i \cap \Delta(y_1, y_2)] \cdot PDF_{\mu,\sigma}(\mathbf{y}) d\mathbf{y} \tag{14}$$

In (14) also the summation is done after integration. This is allowed, because integration is a linear mapping.

Details of the Constant Time Integration

$$\mathrm{EHVI}(\mu, \sigma, \mathrm{Y}, \mathbf{r}) = \sum_{i=1}^{n+1} \int_{y_1=-\infty}^{y_1^{(i-1)}} \int_{y_2=-\infty}^{y_2^{(i)}} \lambda_2[S_i \cap \Delta(y_1, y_2)] \cdot PDF_{\mu,\sigma}(\mathbf{y}) d\mathbf{y} \tag{15}$$

$$\begin{aligned} = & \sum_{i=1}^{n+1} \int_{y_1=-\infty}^{y_1^{(i)}} \int_{y_2=-\infty}^{y_2^{(i)}} \lambda_2[S_i \cap \Delta(y_1, y_2)] \cdot PDF_{\mu,\sigma}(\mathbf{y}) d\mathbf{y} + \\ & \sum_{i=1}^{n+1} \int_{y_1=y^{(i)}}^{y_1^{(i-1)}} \int_{y_2=-\infty}^{y_2^{(i)}} \lambda_2[S_i \cap \Delta(y_1, y_2)] \cdot PDF_{\mu,\sigma}(\mathbf{y}) d\mathbf{y}. \end{aligned} \tag{16}$$

Recall the definition of the standard Gaussian PDF and CDF: $\phi(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$, $\Phi(x) = \frac{1}{2}(1 + \mathrm{erf}(\sqrt{2}))$, and a function Ψ that was defined in [8] as follows:

$$\Psi(a, b, \mu, \sigma) = \int_{-\infty}^{b} (a - z) \frac{1}{\sigma} \phi\left(\frac{z - \mu}{\sigma}\right) dz.$$

Moreover it can be shown that

$$\Psi(a,b,\mu,\sigma) = \int_{-\infty}^{b}(a-z)\frac{1}{\sigma}\phi\left(\frac{z-\mu}{\sigma}\right)dz \quad = \sigma\phi\left(\frac{b-\mu}{\sigma}\right) + (a-\mu)\Phi\left(\frac{b-\mu}{\sigma}\right).$$

Then the first summand of (16) can be written as follows:

$$\begin{aligned}
&= \sum_{i=1}^{n+1}\int_{y_1=-\infty}^{y_1^{(i)}}\int_{y_2=-\infty}^{y_2^{(i)}}\lambda_2[S_i \cap \Delta(y_1,y_2)]\cdot PDF_{\mu,\sigma}(\mathbf{y})d\mathbf{y},\\
&= \sum_{i=1}^{n+1}\int_{y_1=-\infty}^{y_1^{(i)}}(y_1^{(i-1)}-y_1^{(i)})\cdot PDF_{\mu_1,\sigma_1}(y_1)dy_1\int_{y_2=-\infty}^{y_2^{(i)}}(y_2^{(i)}-y_2)\cdot PDF_{\mu_2,\sigma_2}(y_2)dy_2,\\
&= \sum_{i=1}^{n+1}(y_1^{(i-1)}-y_1^{(i)})\int_{y_1=-\infty}^{y_1^{(i)}}PDF_{\mu_1,\sigma_1}(y_1)dy_1\int_{y_2=-\infty}^{y_2^{(i)}}(y_2^{(i)}-y_2)\cdot PDF_{\mu_2,\sigma_2}(y_2)dy_2,\\
&= \sum_{i=1}^{n+1}(y_1^{(i-1)}-y_1^{(i)})\cdot\Phi\left(\frac{y_1^{(i)}-\mu_1}{\sigma_1}\right)\cdot\Psi(y_2^{(i)},y_2^{(i)},\mu_2,\sigma_2). \qquad (17)
\end{aligned}$$

And the second summand of (16) can be written as follows:

$$\begin{aligned}
&= \sum_{i=1}^{n+1}\int_{y_1=y^{(i)}}^{y_1^{(i-1)}}\int_{y_2=-\infty}^{y_2^{(i)}}\lambda_2[S_i \cap \Delta(y_1,y_2)]\cdot PDF_{\mu,\sigma}(\mathbf{y})d\mathbf{y},\\
&= \sum_{i=1}^{n+1}\int_{y_1=y^{(i)}}^{y_1^{(i-1)}}(y_1^{(i-1)}-y_1)\cdot PDF_{\mu_1,\sigma_1}(y_1)dy_1\cdot\int_{y_2=-\infty}^{y_2^{(i)}}(y^{(i)}-y_2)\cdot PDF_{\mu_2,\sigma_2}(y_2)dy_2,\\
&= \sum_{i=1}^{n+1}\left(\Psi(y_1^{(i-1)},y_1^{(i-1)},\mu_1,\sigma_1)-\Psi(y_1^{(i-1)},y_1^{(i)},\mu_1,\sigma_1)\right)\cdot\Psi(y_2^{(i)},y_2^{(i)},\mu_2,\sigma_2).
\end{aligned} \qquad (18)$$

The C++ and MATLAB source-code for computing the EHVI is made available under http://moda.liacs.nl or on request by the authors. The code has been compared to results of Monte Carlo integration and earlier implementations of the exact EHVI.

Numerical Example

The behavior of the BGO based on the EHVI will be illustrated by a single numerical experiment.

The numerical example is visualized in the plots of Fig. 4.The bicriteria optimization problem is: $f_1(\mathbf{x}) = ||\mathbf{x}-\mathbf{1}|| \rightarrow \min$, $f_2(\mathbf{x}) = ||\mathbf{x}+\mathbf{1}|| \rightarrow \min$, and $\mathbf{x} \in [-2,2]\times[-2,2] \subset \mathbb{R}^2$. The Pareto front is the line segment from $(0, 2\cdot\sqrt{2})$ to

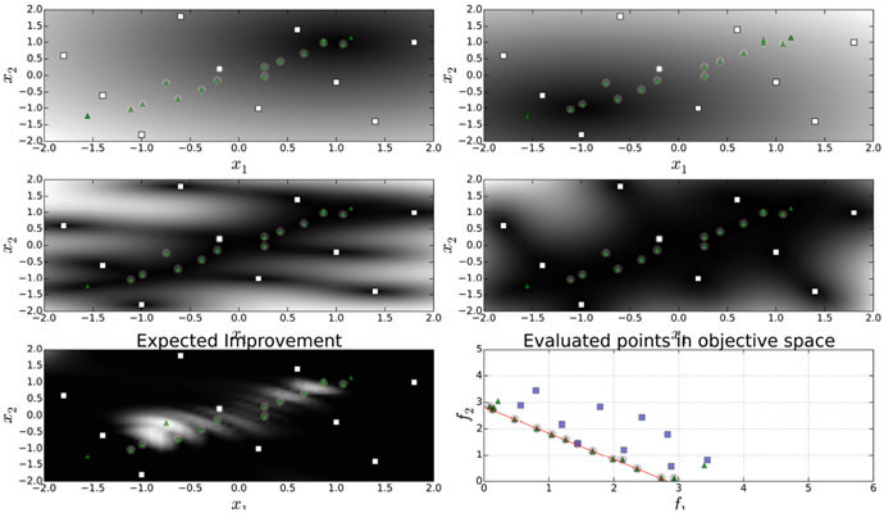

Fig. 4 Example run of multicriteria Bayesian global optimization

$(2 \cdot \sqrt{2}, 0)$, the efficient set is the line segment that connects $(-1,-1)$ and $(1,1)$. The metamodel used is a Gaussian random field model with Gaussian correlation function $\exp(-\theta||\mathbf{x}^{(1)} - \mathbf{x}^{(2)}||^2)$, for $\mathbf{x}^{(1)} \in \mathbb{R}^m$ and $\mathbf{x}^{(2)} \in \mathbb{R}^m$. We set $\theta = 0.0001$, which was estimated by maximum likelihood method for initial sample. An initial set of 10 points was evaluated, indicated by the dark blue squares. From this starting set 15 new points were generated using the EHVI. The maximizer of the expected improvement was found using a uniform grid. In total each objective function was evaluated 25 times.

The results of the experiment are depicted in plots. In all pictures, points that have been evaluated are indicated by triangles. The points from the initial set are additionally marked by squares. Efficient points are surrounded by circles. The top row depicts the mean value of the Gaussian random field model at $\mathbf{x} \in [-\mathbf{2}, \mathbf{2}] \times [-\mathbf{2}, \mathbf{2}]$ for f_1 and f_2, resp. Likewise, the middle row depicts the variance of the Gaussian random field model at $\mathbf{x} \in [-\mathbf{2}, \mathbf{2}] \times [-\mathbf{2}, \mathbf{2}]$ for f_1 and f_2, resp. On the left-hand side of the bottom row the hypervolume-based expected improvement values after 25 iterations are shown. The final set of points in the objective space and the Pareto front approximation is seen in the plot in the lower right corner. Using only *25 evaluations of the original objective functions*, the algorithm finds a good approximation to the Pareto front.

Application Notes and Further Reading

In addition to this experiment, other applications of the EHVI have been recently reported. It was first used as selection criterion in evolutionary optimization [3] and in the context of airfoil design [4] and quantum control [18]. To our knowledge, it was used for the first time in BGO in the context of airfoil optimization in [13] and conceptually compared other multicriteria infill criteria, including proposal made in [11], [10], and in [23]. Other applications are robotics [20], biogas plant controllers [6], event detection in water quality management [24], structural design optimization [17], and tuning of machine learning tools [12]. An empirical comparison with other infill criteria is found in [16].

Summary and Outlook

This chapter described the EHVI as a multicriteria generalization of the expected improvement used in BGO. This generalization is based on the hypervolume indicator, which is a quality indicator for Pareto front approximations. It has recently served as an infill criterion in a number of BGO case studies, but was criticized for its high computational complexity. In this chapter, the time complexity of the 2-D EHVI was shown to be only $\Theta(n)$. The linear time algorithm presented in this paper improves upon previously proposed algorithms which required quadratic time complexity. It assumes a sorted Pareto front (otherwise its complexity is $O(n \log n)$), which is typically given in BGO. During a single iteration of BGO a large number of evaluations need to be performed, in order to find a minimizer based on the Gaussian random field model. Therefore the fast algorithm will be of great benefit for reducing the running time of multicriteria BGO based on EHVI.

Future research will investigate in more depth the theoretical properties of the EHVI. For the first results in this direction refer to [5], where it was shown that the 2-D EHVI is monotonic in the mean values and variance. Also it will be interesting to analyze the time complexity of EHVI for more than two objective functions.

Acknowledgements Hao Wang gratefully acknowledges support by the Netherlands Organisation for Scientific Research, NWO ICT PPP Project Grant "Process mining for multi-objective online control (PROMIMOOC)". Kaifeng Yang acknowledges financial support from China Scholarship Council (CSC), CSC No. 201306370037.

References

1. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, pp. 87–102. ACM, Chicago (2009)
2. Couckuyt, I., Deschrijver, D., Dhaene, T.: Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. J. Global Optim. **60**(3), 575–594 (2014)

3. Emmerich, M.: Single-and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. Ph.D. thesis, Fachbereich Informatik, Chair of Systems Analysis, University of Dortmund (2005)
4. Emmerich, M., Giannakoglou, K.C., Naujoks, B.: Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans. Evol. Comput. **10**(4), 421–439 (2006)
5. Emmerich, M., Deutz, A.H., Klinkenberg, J.W.: Hypervolume-based expected improvement: monotonicity properties and exact computation. In: 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 2147–2154. IEEE, New Jersey (2011)
6. Gaida, D.: Dynamic real-time substrate feed optimization of anaerobic co-digestion plants. Ph.D. thesis, Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University (2014)
7. Hupkens, I., Emmerich, M., Deutz, A.: Faster computation of expected hypervolume improvement. arXiv preprint arXiv:1408.7114 (2014)
8. Hupkens, I., Deutz, A., Yang, K., Emmerich, M.: Faster exact algorithms for computing expected hypervolume improvement. In: Evolutionary Multi-Criterion Optimization, pp. 65–79. Springer, Berlin, Heidelberg (2015)
9. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455–492 (1998)
10. Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA J. **44**(4), 879–891 (2006)
11. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. **10**(1), 50–66 (2006)
12. Koch, P., Wagner, T., Emmerich, M.T., Bäck, T., Konen, W.: Efficient multi-criteria optimization on noisy machine learning problems. Appl. Soft Comput. **29**, 357–370, New Jersey (2015)
13. Łaniewski-Wołłk, Ł., Obayashi, S., Jeong, S.: Development of expected improvement for multi-objective problem. In: Proceedings of 42nd Fluid Dynamics Conference/Aerospace Numerical Simulation Symposium (2010)
14. Mockus, J.: Bayesian Approach to Global Optimization: Theory and Applications, vol. 37. Springer Science & Business Media, New York (2012)
15. Mockus, J., Tiesis, V., Žilinskas, A.: The application of Bayesian methods for seeking the extremum. In: Towards Global Optimization, vol. 2, pp. 117–129. North-Holland, Amsterdam (1978)
16. Shimoyama, K., Sato, K., Jeong, S., Obayashi, S.: Comparison of the criteria for updating Kriging response surface models in multi-objective optimization. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE, New Jersey (2012)
17. Shimoyama, K., Sato, K., Jeong, S., Obayashi, S.: Updating Kriging surrogate models based on the hypervolume indicator in multi-objective optimization. J. Mech. Des. **135**(9), 094503 (2013)
18. Shir, O.M., Emmerich, M., Bäck, T., Vrakking, M.J.: The application of evolutionary multicriteria optimization to dynamic molecular alignment. In: IEEE Congress on Evolutionary Computation, 2007, CEC 2007, pp. 4108–4115. IEEE, New Jersey (2007)
19. Stein, M.L.: Interpolation of Spatial Data: Some Theory for Kriging. Springer Science & Business Media, New York (2012)
20. Tesch, M., Schneider, J., Choset, H.: Adapting control policies for expensive systems to changing environments. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 357–364. IEEE, New Jersey (2011)
21. Törn, A., Žilinskas, A.: Global Optimization. Springer, New York (1989)
22. Vazquez, E., Bect, J.: Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. J. Stat. Plan. Inference **140**(11), 3088–3095 (2010)

23. Wagner, T., Emmerich, M., Deutz, A., Ponweiser, W.: On expected-improvement criteria for model-based multi-objective optimization. In: Parallel Problem Solving from Nature. PPSN XI, pp. 718–727. Springer, Berlin, Heidelberg (2010)
24. Zaefferer, M., Bartz-Beielstein, T., Naujoks, B., Wagner, T., Emmerich, M.: A case study on multi-criteria optimization of an event detection software under limited budgets. In: Evolutionary Multi-Criterion Optimization, pp. 756–770. Springer, Berlin, Heidelberg (2013)
25. Žilinskas, A., Mockus, J.: On one Bayesian method of search of the minimum. Avtomatika i Vychislitel'naya Teknika **4**, 42–44 (1972)

Understanding the Impact of Constraints: A Rank Based Fitness Function for Evolutionary Methods

Eric S. Fraga and Oluwamayowa Amusat

Abstract There are design problems where some constraints may be considered objectives as in "It would be great if the solution we obtained had this characteristic." In such problems, solutions obtained using multi-objective optimisation may help the decision maker gain insight into what is achievable without fully satisfying one of these constraints. A novel fitness function is introduced into a multi-objective population based evolutionary optimisation method, based on a plant propagation algorithm extended to multi-objective optimisation. The optimisation method is implemented and applied to the design of off-grid integrated energy systems for large scale mining operations where the aim is to use local renewable energy generation, coupled with energy storage, to eliminate the need for transporting fuel over large distances. The latter is a desired property and in this chapter is treated as a separate objective. The results presented show that the fitness function provides the desired selection pressure and, when combined with the multi-objective plant propagation algorithm, is able to find good designs that achieve the desired constraint simultaneously.

Keywords Multi-objective optimization • evolutionary methods • plant propagationalgorithm • process design • integrated energy systems • Pareto extremes

Introduction

Model based process design is often formulated as an optimization problem. The problem definition includes one or more objective functions and both equality and inequality constraints. In process design, many of the constraints originate from underlying physical laws. For instance, the temperature of a distillation plate at equilibrium is described by Raoult's law, an equality constraint, or the amount of

E.S. Fraga (✉) • O. Amusat
Centre for Process Systems Engineering, Department of Chemical Engineering,
University College London, London, UK
e-mail: e.fraga@ucl.ac.uk; oluwamayowa.amusat.13@ucl.ac.uk

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_13

mass in a vessel must be greater than 0, an inequality constraint. These constraints cannot be violated if the design obtained is to be realizable physically.

However, there are other types of constraints. Some constraints are of the form of

It would be great if the solution we obtained had this characteristic.

Examples include: the temperature of the room in the dwelling should be 20 °C; the purity of this by-product should be at least 90 %; the pressure changes should be less than some amount specified; and so on. Constraints such as these can be reformulated as objectives and then either incorporated into a single objective function using a penalty term or the problem is transformed into a multi-objective problem. The use of penalty terms (or weighted objective functions equivalently for multi-objective formulations) using difficult due to the need to choose the penalty weights. Defining the problem as a multi-objective optimization problem and using multi-objective solution methods is therefore more attractive.

There are many methods for multi-objective optimization; see, for instance, [3] for a selected review, concentrating on evolutionary methods. Multi-objective optimization methods will traditionally attempt to generate a population of solutions that consist of *non-dominated* solutions and therefore represent an approximation to the Pareto front [8], assuming that they are able to identify globally optimal solutions [10]. As such, in principle, any multi-objective method is suitable for tackling the types of problems noted above. On the other hand, multi-objective problems that are derived from relaxing a *desirable* constraint are subtly different from more general multi-objective problems: the original single objective is somehow more important. Therefore, it is desirable to have a multi-objective optimization method that emphasizes solutions that have more favorable values for that objective. The aim is to provide the design engineer with insight into how the relaxation of the desirable constraint affects the main objective function.

The aim of this chapter is to present a fitness function that provides the selection pressure on population based evolutionary optimization methods to generate solutions with a preference for those that improve the main objective function but not at the expense of ignoring the other objectives completely. We illustrate this fitness function using a new multi-objective evolutionary method adapted from an existing single objective *plant propagation algorithm* [9].

The novel multi-objective optimization method is presented and is applied to the problem of designing an integrated energy generation and storage system for off-grid mining operations. Although the design objective is to minimize capital cost, there is a requirement imposed to design the system so that it needs no fuel brought in from off-site. This requirement is a desired property and in this chapter is treated as a separate objective. The results presented show that a multi-objective approach does provide for a better understanding of what is possible and hence what may be desirable as the final solution.

A Multi-Objective Rank Based Fitness Function for Pareto Extremes

Consider the set of points shown in Fig. 1 which plots the points in the space of objective function values for a bi-criteria problem. The assignment of fitness values to these points can be done in a variety of ways. Typically, the non-dominated solutions will all be given the same fitness value, the best fitness when compared with the fitness assigned to dominated points. The dominated points will then be assigned fitness values in different ways. One approach is to remove the non-dominated points from the set, find the now non-dominated points, and assign these all the same fitness value, one that is worse than that assigned to the original non-dominated points. Then repeat the same process until no points are left. An alternative is to assign a fitness to the originally dominated points based on the distance of these points to either an approximation to the Pareto front defined by the piecewise linear fit to the non-dominated points or by the distance to the nearest non-dominated point [5]. The aim of these fitness methods is to emphasize the non-dominated points and hence drive an evolutionary algorithm towards a good approximation to the Pareto front.

Although the Pareto front is of interest, for the design problems described above, we are particularly interested in at least one of the end-points of this front. End-points correspond to the solution of individual single criterion problems. We are interested in these end-points because they correspond either to the original design criterion or to one or other design constraints that we have relaxed to gain an understanding of the impact of these criteria. Therefore, although the Pareto front as a whole is of interest, the end-points are particularly relevant. We therefore wish to define a fitness function that emphasizes the end-points and hopefully drives the evolutionary algorithm to improve these as much as possible but without sacrificing

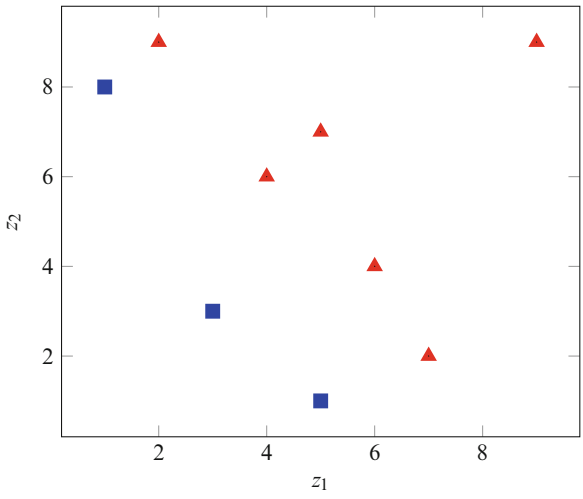

Fig. 1 Simple scatter plot with non-dominated points indicated by *blue squares* and dominated points by *red triangles*, assuming that the goal is to minimize both objectives, z_1 and z_2

Table 1 Points and their fitness for the illustration example in Fig. 1 sorted in decreasing order according to the rank based fitness

z_1	z_2	I_1	I_2	$I_1 \odot I_2$	f
5	1	6	1	6	0.93
1	8	1	7	7	0.91
3	3	3	3	9	0.89
7	2	8	2	16	0.80
2	9	2	8	16	0.80
4	6	4	5	20	0.75
6	4	7	4	28	0.65
5	7	5	6	30	0.63
9	9	9	9	81	0.00

the Pareto front completely. If the Pareto front were not of interest, we could simply solve a set of single criterion problems.

To achieve the desired fitness values that emphasize not only the Pareto front but especially the end-points of that front, we have defined a rank based fitness function which combines the ranks assigned to each point with respect to each criterion individually:

$$f = 1 - \frac{I_1 \odot I_2 \odot \ldots \odot I_{n_c}}{n_p^{n_c}}$$

where $\odot$ represents the element-wise or Hadamard product of two vectors. The vectors I_j, $j = 1, \ldots, n_c$, each of length n_p, are the indices for each point of their position when the points are sorted with respect to criterion j. If two or more points have the same objective function value, they are implicitly coalesced prior to the assignment of rank for that objective function and hence given the same ranking. The product of the individual rankings denotes the fitness $f_i \in [0, 1)$ for $i = 1, \ldots, n_p$ where larger values indicate better fitness. n_p is the number of points and n_c the number of criteria.

Table 1 illustrates the values of the various vectors for the points shown in Fig. 1 sorted according to the fitness value assigned to the points. The best fitness values are for the two extreme points (5,1) and (1,8) with the next best point being the remaining non-dominated point (3,3).

In general, the largest fitness value achievable is

$$1 - \frac{1}{n_p^{n_c}}$$

which approaches 1 asymptotically as $n_p \to \infty$. This value can only be achieved if one point dominates all the rest. The lowest fitness value is 0, illustrated in the table by the last row, a point that is worst in both criteria. For a population that consists solely of non-dominated points, the fitness along the Pareto front would start at a maximum value and decrease until the middle point of the set is reached and would then start increasing again. The maximum fitness achievable, in this case, will be

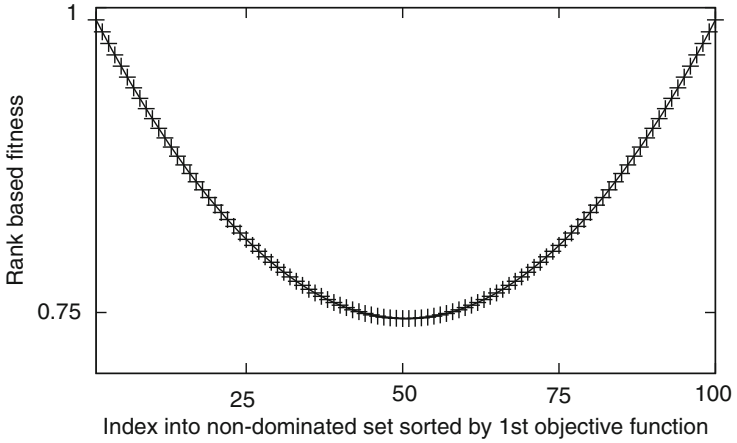

Fig. 2 The fitness of solutions along the Pareto front if all solutions are non-dominated. In the plot, the solutions have been sorted according to the first objective function

$$f_{\max} = 1 - \frac{1}{n_p^{n_c-1}}$$

and the minimum fitness

$$f_{\min} = 1 - \frac{1}{4n_p^{n_c-2}} \quad .$$

With $n_c = 2$ and $n_p = 100, f_{\max} = 0.99$ and $f_{\min} = 0.75$. This is illustrated in Fig. 2.

A Multi-Objective Plant Propagation Algorithm

The fitness function defined in the previous section could be used with most population based evolutionary algorithms, such as a genetic algorithm. However, we have had good experience with the *Strawberry* algorithm [9], an implementation of a *plant propagation* nature inspired evolutionary method. Further evidence of the power of this approach has been provided by Merrikh-Bayat [6].

The Strawberry algorithm was originally implemented for single criterion optimization so it has been extended here for multi-objective problems. This extended algorithm is shown in Algorithm 6. The basic premise is that plants that are in a good position (fertile soil and plenty of water) will reproduce with greater probability but will tend to do so in the vicinity of where they are. Less often, plants which are not well situated will reproduce through longer distance methods. In the Strawberry algorithm, both single and multiple objective versions, each member of the population can generate a number of runners, proportional to that member's

Algorithm 6 The Strawberry plant propagation algorithm [9] extended for multi-objective optimization

Given: $f(x)$, a vector function; n_g, number of generations to perform, n_p, the propagation size; n_r, maximum number of runners to propagate.
Output: z, vector approximation to Pareto front.

```
p ← initial random population of size n_p
for n_g generations do
    prune population p, removing similar solutions
    N ← fitness(p)                                  ▷ Use rank based fitness
    p̃ ← φ                                           ▷ Empty set
    for i ← 0 ... n_p do
        x ← select(p, N)                            ▷ Tournament fitness based selection
        for each runner to generate do              ▷ Number proportional to fitness rounded up
            x̃ ← new solution(x, 1 − N)              ▷ Distance inversely proportional to fitness
            p̃ ← x̃ ∪ p̃                               ▷ Add to new population
        end for
        p ← p\x                                     ▷ Remove from old population
    end for
    p ← p̃ ∪ Nondominated(p)                         ▷ New population with elitism
end for
z ← Nondominated(p)
```

fitness, to define new points a distance away proportional to 1 minus the fitness, with all values randomly chosen.

Case Study: Off-Grid Energy Systems Design with Renewable Energy

Mining operations often are located in geographically remote regions of the planet. These operations are seldom connected to grid supplies of energy, either electrical or fuel. As a result, the operations typically require transport of fuel, e.g., diesel, over large distances using trucks or equivalent. There is a desire to reduce the need for the transport of fuel and one possibility is the use of local energy generation. This local generation can be one of solar photovoltaic (PV), solar thermal or wind turbines, or a combination of these. Local generation may or may not have economic benefits but such generation will usually have a positive environmental impact when compared with transporting fuel.

Beyond the economics of the choice of generation technology, whether to generate locally at all is the key issue of continuous operation. Many mining sites operate 24 h a day. A distinguishing factor of the renewable energy generation technologies mentioned above is that they have variable output. Solar based technologies obviously do not generate energy when the sun is not in the sky. Wind turbines can generate energy through day and night but the amount will vary from one moment to the next.

The variability of generation requires changing the design of the mining operation to incorporate energy storage. A number of storage options can be considered for large scale operations: molten salts, pumped hydraulic and compressed air. The design problem then requires identifying the appropriate combination of both generating and storage technologies to minimize the cost of the mining operation. We have previously addressed the minimization of capital cost [2].

The optimization problem for the design problem included a constraint that specified that the power and heat demands of the mining operation had to be satisfied fully from local generated energy from renewable sources. However, in practice, due to the variability of the energy supplies, even with storage, designing for complete reliance on local generation will lead to over-design. Instead, it is more appropriate to design for almost complete reliance on local generation but allowing for the use of fuel brought in from off-site. In other words, from an optimization point of view, it may be useful to relax the constraint and gain insight into how much impact allowing the use of off-site energy sources, hopefully infrequently, may have.

An analysis of the impact of the constraint was undertaken using a scenario based approach [1]. In this approach, a set of scenarios was generated, with each having a different solar profile over the period of time considered. Each profile was generated randomly. The single objective optimization problem was solved for each scenario and the resulting design analyzed in terms of how likely it was to not meet the demand under different solar profiles. Although this approach was useful, it did not necessarily represent the best solutions possible in a probabilistic sense. A more rigorous approach would be to consider relaxing the demand satisfaction constraint and treating the design problem as a bi-criteria optimization problem, as discussed above. The demand constraint is a desired attribute of a design but not a hard constraint.

Using the models developed by Amusat et al. [1] and adding the probability of not meeting demand as a second objective, the problem is now

$$\min_{d} z = \begin{cases} c(d) \\ p(d) \end{cases} \tag{1}$$

where d are the design variables, $c(\cdot)$ is the capital cost of the energy generation and storage systems, and $p(\cdot)$ is the probability of not meeting the demand fully with local generation sources. The probability is 1 minus the reliability. Reliability is a measure of the ability of an energy system to deliver power to all points of consumption with the frequency, the duration, and the extent required by the operation [7].

The evaluation of the objective functions proceeds as follows:

1. For a given d, the generation and storage technologies are defined, resulting in an energy system for the mining operation.
2. The resulting design is then evaluated over a number of randomly generated scenarios based on a probability distribution function describing the variability of solar irradiance for each hour of each day in the period of operation.

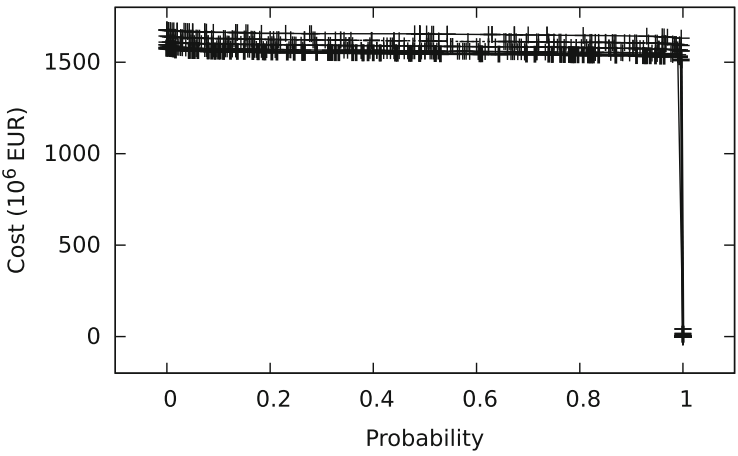

Fig. 3 Final sets of non-dominated points resulting from 5 attempts at the off-grid design problem using NSGA-II [4] where the probability is the likelihood of not satisfying the energy demands of the mining operation with only local energy generation

3. The probability of not meeting demand is simply the ratio of the number of scenarios where the demand was not met for design d, for at least one time period, and the number of total scenarios. A value of 0 means that the design is able to meet the demand under all likely solar conditions; a value of 1 means that the design never fully meets the demand, always requiring the import of off-site fuel for at least one time period over the full duration for each scenario.

As a starting point, we have solved this bi-criteria problem of dimension 8 using NSGA-II [4] with population size 100, 150 generations, crossover rate of 0.25, mutation rate of 0.25, with binary tournament selection, intermediate crossover, and Gaussian mutation. The non-dominated objective function values for 5 attempts are presented in Fig. 3. At the scale used, NSGA-II appears to identify the set of non-dominated designs well. From an engineering point of view, we do see that the designs are not that sensitive to the variability in solar irradiance. This is not entirely surprising as the case study consists of the Collahuasi mine located in the Atacama region of Chile where most days have completely clear skies and so the irradiance is relatively constant and predictable.

Figure 4 shows the different sets of non-dominated points for the left side of the plot shown in Fig. 3. It is this part of the plot we are most interested in as the designs here are those that will not very often require off-site fuel. By zooming in, we see that the solutions obtained are similar in objective function value from one attempt with NSGA-II to another.

For comparison, we have also applied the Strawberry method using the rank based fitness function described above with population size 100, 150 generations, and $n_r = 5$. The aim of this fitness function is to emphasize the end-points of the

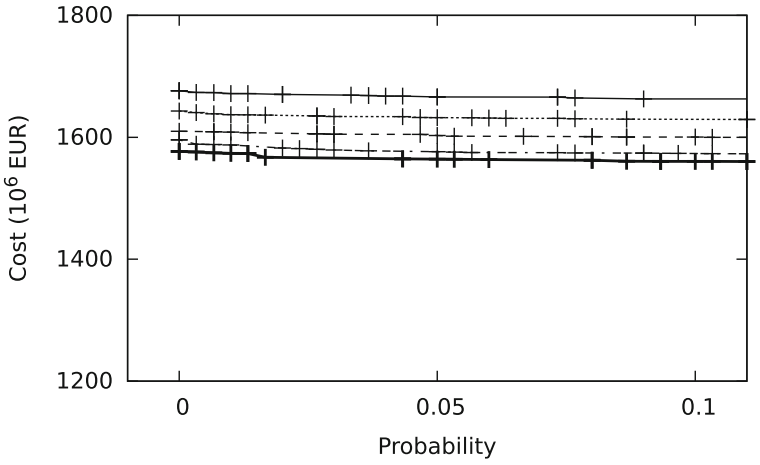

Fig. 4 A zoomed in view of Fig. 3

Pareto front and, due to the asymmetry in evaluation, particularly the left end-point. Figure 5 shows the resulting solutions. Of note,

1. The Strawberry algorithm is less consistent over the full range of probability values.
2. The number of points in the set of non-dominated points is small compared with the sets generated by NSGA-II.
3. The distribution of points is less even than it is for the NSGA-II case with points concentrated more towards the extremes of the Pareto front than towards the center. Note that the use of diversity control reduces the number of points at the right extreme.
4. The cost objective function values obtained are lower than those obtained using NSGA-II.

If we zoom in as we did for the NSGA-II results, we see (Fig. 6) that the cost objective function is often better than what is obtained with NSGA-II.

Analysis of the Designs

The aim of using a multi-objective approach to solve the constrained single objective design problem is to gain insight into the impact of the particular constraint on the designs obtained. In the off-grid design problem, the question the design engineer would like to answer is: Does the constraint of not using any off-site fuel lead to a significant over-design?

Table 2 shows the values of the first 6 design points, counting from the left, from the bottom graph in Fig. 6. The cost does not include the actual cost of the off-site

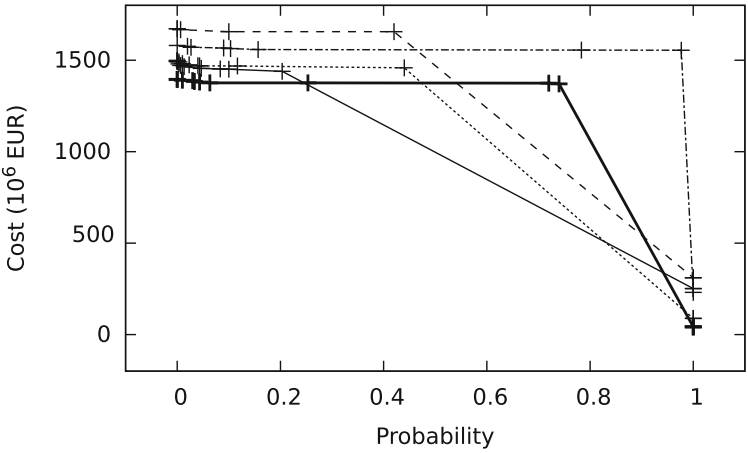

Fig. 5 Final sets of non-dominated points resulting from 5 attempts at the off-grid design problem using the new multi-objective Strawberry algorithm where the probability is the likelihood of not satisfying the energy demands of the mining operation with only local energy generation

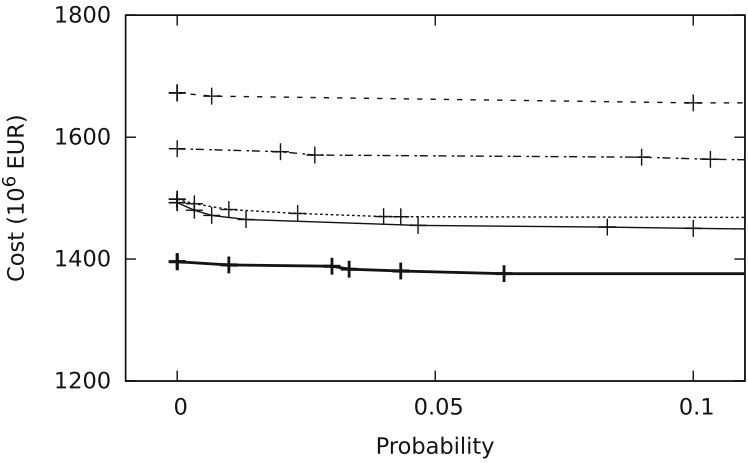

Fig. 6 A zoomed in view of Fig. 5

fuel and the cost of transporting that fuel so this table (and the results discussed earlier) only allows us to analyze the impact on the physical structure of the mining operation's energy systems.

The final row shows the variation as a percentage of the maximum value for each design variable and objective function. From the first design through to the last one, there is a change of 1 % in cost. Some variables are stable whereas the uses of photovoltaic generation and compressed air storage have large changes: the feasible design does not include either of these technologies. Introducing them allows the cost to be reduced but at the expense of not meeting demands in all scenarios.

Table 2 Design points with low probability of not meeting the demand for the mining operation

	G_{PT} (MW)	S_{MS} (MWh)	E_{MS} (MW)	G_{PV} (MW)	S_{PS} (MWh)	E_{PS} (MW)	S_{CA} (MWh)	E_{CA} (MW)	p	c 10^6
	1257	6022	180	0.00	2746	89.2	0.00	60.26	0.000	1396
	1246	6036	177	0.00	2740	94.8	0.00	64.80	0.010	1390
	1238	6000	179	0.00	2749	96.4	25.98	64.73	0.030	1388
	1235	6021	179	0.87	2839	93.5	1.44	58.34	0.033	1383
	1232	6029	177	1.54	2795	93.6	0.00	59.79	0.043	1381
	1228	6021	179	0.00	2736	89.8	0.00	65.10	0.063	1376
Δ (%)	2	1	2	100	4	7	100	10	100	1

The first row is a design which should *always* meet the demand and the second row is one in which the demand will be met all but 1 % of the possible solar profiles that could be encountered. The G columns are generation (power tower, PT, and photovoltaic, PV). The S columns refer to storage: MS for molten salts, PH for pumped hydro, and CA for compressed air. Finally, E columns indicate the peak electricity release rate from storage

For the design engineer, the main conclusion is that the feasible design is not over-specified. In fact, some of the slightly less expensive designs introduce more complexity by the incorporation of further alternative technologies. Generally, the simpler the design, the more attractive it is so the feasible design is favored even more.

The ability to perform this analysis enables the engineer to have the confidence necessary to move to the next stage of design: the detailed specification of the individual technologies and further modelling, simulation, and optimization.

Conclusions

The use of multi-objective optimization can provide useful insight into the impact of constraints on designs. By converting a constraint to an extra objective, the approximation of the Pareto front for the design problem will help determine, for instance, whether the "feasible" design is over-constrained or not. To ensure that the impact of the relaxation of a constraint is understood, it is necessary to have a good approximation to the Pareto front at the extremes. This motivates the definition of a fitness function to provide the appropriate selection pressure for evolutionary methods.

A rank based fitness function has been presented. An implementation has been incorporated in a new multi-objective plant propagation algorithm based on the existing single objective *Strawberry* algorithm [9]. The procedure has been applied to a problem in off-grid operation of large scale mining where there is a desire to reduce the cost and environmental impact of using fuel brought in from a long

distance away. The results demonstrate the effectiveness of both the fitness function and the multi-objective Strawberry method.

Acknowledgements The authors would like to acknowledge the funding provided by the Nigerian government through the Presidential Scholarship for Innovation and Development (PRESSID) scheme.

References

1. Amusat, O., Shearing, P., Fraga, E.S.: Optimal integrated energy systems design incorporating variable renewable energy sources. In: Roskilly, T., Rajendran, K., Ling-Chin, J. (eds.) Proceedings of Sustainable Thermal Energy Management (SusTEM), pp. 245–253 (2015)
2. Amusat, O., Shearing, P., Fraga, E.S.: System design of renewable energy generation and storage alternatives for large scale continuous processes. In: Gernaey, K.V., Huusom, J.K., Gani, R. (eds.) 12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering, pp. 2279–2284. Elsevier B.V., Amsterdam (2015)
3. Coello Coello, C.A., Landa Becerra, R.: Evolutionary multiobjective optimization in materials science and engineering. Mater. Manuf. Process. **24**(2), 119–129 (2009)
4. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**, 311–338 (2000)
5. Fiandaca, G., Fraga, E.S., Brandani, S.: A multi-objective genetic algorithm for the design of pressure swing adsorption. Eng. Optim. **41**(9), 833–854 (2009)
6. Merrikh-Bayat, F.: The runner-root algorithm: a metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. Appl. Soft Comput. **33**, 292–303 (2015)
7. Osborn, J., Kawann, C.: Reliability of the US electricity system: Recent trends and current issues. Technical Report, Energy Analysis Department, Ernest Orlando Lawrence Berkeley National Laboratory, LBNL-47043, Berkeley (2001)
8. Pardalos, P.M., Žilinskas, A., Žilinskas, J.: Non-Convex Multi-Objective Optimization. Springer, New York (2016)
9. Salhi, A., Fraga, E.S.: Nature-inspired optimisation approaches and the new plant propagation algorithm. In: Proceedings of ICeMATH 2011, The International Conference on Numerical Analysis and Optimization, Yogyakarta, pp. K2:1–8 (2011)
10. Törn, A., Žilinskas, A.: Global Optimization. Springer, Berlin (1989)

Estimating the Pareto Front of a Hard Bi-criterion Competitive Facility Location Problem

Algirdas Lančinskas, Pascual Fernández, Blas Pelegrín, and Julius Žilinskas

Abstract We deal with the location problem for a franchise type expanding firm in competition with other firms in a geographical area. The firm aims at maximization of the market share captured by the new facilities and minimization of the lost market share of the old facilities caused by the entering of the new facilities in the market. The market share of each facility is estimated assuming that customers are served by the most attractive facility. A new tie breaking rule is introduced to serve the customers for which there are more than one facility with the maximum attraction, which leads to a hard nonlinear bi-objective optimization problem. A heuristic algorithm is proposed which obtains a good approximation of the Pareto front when the new facilities have to be selected from a finite set of candidates.

Keywords competitive facility location; firm expansion; multi-objective optimization

Introduction

The location of facilities is a strategic decision for a firm that competes with other firms to provide goods or services to the customers in a given geographic area. Different location models and solution procedures have been proposed to cope with these problems which vary depending on the ingredients to be considered such as location space, facility attraction, customer patronizing behavior, demand function, decision variables, and so on (see, for instance, survey papers [10, 13, 26, 29]).

Most of the models in the literature deal with the location problem for an *entering firm* that will compete for the market in a certain region where other firms

A. Lančinskas (✉) • J. Žilinskas
Institute of Mathematics and Informatics, Vilnius University, Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: algirdas.lancinskas@mii.vu.lt; julius.zilinskas@mii.vu.lt

P. Fernández • B. Pelegrín
Department of Statistics and Operations Research, University of Murcia, Murcia, Spain
e-mail: pfdez@um.es; pelegrin@um.es

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_14

offering the same goods or services are already established. The entering firm aims at maximization of market share, or profit, by taking into account the patronizing behavior of customers. Some variants of the attraction model proposed by Huff [18] have been used as choice rules to estimate the market share among the competing facilities. In this type of models, the attraction of a facility depends on the distance between the customer and the facility, as well as on some characteristics of the facility defined through a parameter called the quality of the facility. The most common customer choice rules are the ones called *proportional* and *binary*. The proportional rule means that the customer patronizes all the facilities in proportion to facility attraction (see, for instance, [8, 24, 31]). The binary rule means that the customer patronizes the most attractive facility (see [16, 34, 35]).

The case of an *expanding firm* has received less attention in the location literature. In this case the firm is already in the market, but it is interested in increasing its market share by opening new facilities. As a result of the entrance of new facilities, the old facilities owned by the firm might lose part of their market. This effect is called *cannibalization* and it was studied for the first time in franchise systems in [14]. In discrete location space, the cannibalization effect is considered in a single objective location model with variable expenditure functions in [2]. In this model the effect is not explicitly presented, but the cannibalized facilities may increase their demand due to market expansion. Another model was studied in [1] where the cannibalization effect is considered to simultaneously optimize location and quality of the new facilities. In planar location space, the problem has been studied for a single new facility as a bi- objective optimization problem by using a lexicographic approach and interval analysis (see [11, 27]).

Real-world facility location problems are often multi-objective; i.e., several criteria, such as market share, costs for establishment and maintenance, or any undesirable effects, are taken into account when determining locations for the new facilities. See [7, 9] for recent instances of multi-objective facility location problems as well as [3, 4, 41]—for concept and developments in solution of multi-objective optimization problems.

The multi-objective nature also occurs in the location problem for an expanding firm when the facilities in the expanding firm have different owners, as it occur in franchise systems. Then the aim of the owners of the new facilities is market share maximization while the aim of the owners of the old facilities is cannibalization minimization, which lead to a bi-objective optimization problem. In this paper we deal with this bi-objective competitive location problem when multiple new facilities are located in discrete location space and customers use the binary choice rule. When the binary rule is used one open question is how to solve ties when more than one facility have the maximum attraction for a given customer. Usually ties are broken by assigning $1/2$ of the customer demand to a new facility with the maximum attraction. In this case some integer linear programming formulations of the problem for an entering firm have been proposed (see, for instance, [32]). However, the number of tied facilities may vary depending on the location of the new facilities. Thus, if there are three tied facilities, one of them is owned by the expanding firm and the other two are owned by its competitors, the expanding firm

would get $1/3$ of the demand instead of $1/2$. We consider a different approach for tie breaking proposed in [25], which consists of dividing the customer demand among the tied facilities. This leads to solve a bi-objective nonlinear optimization problem which is hard to solve even in discrete location space.

Finding the Pareto-optimal front is difficult and time consuming, therefore algorithms providing an approximation of the true Pareto front are used. One well known class of such algorithms are evolutionary algorithms (EAs), which require little knowledge about the problem being solved—to solve a certain optimization problem, it is enough to be able to evaluate the objective functions for a given set of input parameters. Furthermore EAs are well suited to multi-objective optimization problems as they are fundamentally based on biological processes which are inherently multi-objective. Multi-Objective EAs (MOEAs) can yield to a whole set of potential solutions—which are all optimal in some sense—and give the option to assess the trade-offs between different solutions. A number of different MOEAs have been proposed in the literature [5]; however, most popular of them are Vector Evaluated Genetic Algorithm (VEGA) [30], Strength Pareto Evolutionary Algorithm (SPEA and SPEA2) [39, 40], Pareto Archived Evolutionary Strategy (PAES) [19], and Non-dominated Sorting Genetic Algorithm (NSGA and NSGA-II) [6, 33].

Various MOEAs have been applied to solve various multi-objective optimization problems in facility location. For example, Redondo et al. [28] proposed a general multi-objective optimization heuristic algorithm, suitable to continuous multi-objective optimization problems; Huapu and Jifeng [17] utilized SPEA to solve a bi-level programming model to optimize the location problem of distribution centers, where the upper level consists of two objectives—minimization of cost of construction and distance between distribution center and customers, whereas the lower level minimizes the transportation cost; Villegas et al. [36] utilized NSGA-II to solve a bi-objective facility location problem by minimizing operational cost of Colombian coffee supply network and maximizing the demand; Liao [22] used NSGA-II to optimize the location for distribution centers with respect to two objectives—maximization of customers service and minimization of the total cost; Medaglia et al. [23] utilized hybrid NSGA-II and with mixed-integer programming approach to solve bi-objective obnoxious facility location problem related to the hospital waste management network.

This paper shows that NSGA-II with local search is able to find a good approximation of the Pareto front of the proposed bi-objective location problem. The problem will be formulated in the next section. Section “Approximation of the Pareto Front” describes preliminaries of application of NSGA-II to solve discrete optimization problems and its improvement by our proposed local search strategy. Section “Numerical Experiments” consists of description and discussion of the results of the experimental investigation performed to investigate the performance of the derived algorithm by solving different instances of the discrete bi-objective competitive facility location problem. Finally, some conclusions are presented in section “Conclusions”.

The Location Problem

Let us consider several firms which provide some goods or services to customers in a certain geographical area. One of the firms, called the expanding firm, denoted by A, wants to locate new facilities in order to increase its market share in that area. The new facilities may capture customers who were served by the other firms as well as customers who were served by the old facilities of the expanding firm. Therefore, the firm A is also interested in minimizing the loss of market share of its old facilities caused by the expansion (cannibalization effect). Thus the firm A is interested in the Pareto solutions to this bi-objective location problem.

The customers are spatially separated and they are aggregated at geographic demand points in order to make the problem computationally tractable (see [12]). The customers patronize the most attractive facility to be served. If ties in the maximum attraction for a customer occur, then the buying power is equally divided among the most attractive facilities. This new rule for tie breaking allocates a variable market share to the expanding firm depending on the number of tied facilities owned by the firm. This rule is different from the classical tie breaking rules for which the market share allocated to the expanding firm is fixed. Without loss of generality the other existing competing firms can be considered as one firm called B.

Notation

The following notation will be used in this paper.

Indices:

i—index of the demand point and
j—index of the facility.

Data:

$I = \{1, 2, \ldots, n\}$—the set of the demand points;
n—the number of the demand points;
F_A—the set of existing facilities owned by the firm A;
F_B—the set of existing facilities owned by the firm B;
n_A—the number of existing facilities owned by the firm A;
n_B—the number of existing facilities owned by the firm B;
$L = \{1, 2, \ldots, k\}$—the set of location candidates for the new facilities;
k—the number of location candidates for new facilities;
r—the number of new facilities to be located;
a_{ij}—attraction of the i-th demand point for the j-th facility;
d_{ij}—distance from the location of the i-th demand point to the j-th location; and
w_i—buying power at the i-th demand point.

Decision variable:

X—the set of locations for the new facilities.

Miscellaneous:

F_X—the set of new facilities;
$a_i(F) = max\{a_{ij} : j \in F\}$—the maximum attraction of demand point i for facilities in set $F \subset (F_A \cup F_B \cup F_X)$;
$n_i(F) = |\{j \in F : a_{ij} = a_i(F)\}|$—the number of facilities in F which are the most attractive to demand point i.

Market Share

The set of demand points which are totally captured by the new facilities after the expansion is

$$I^t(X) = \{i \in I : a_i(F_X) > a_i(F_A \cup F_B)\}. \tag{1}$$

The set of demand points which are partially captured by the new facilities (share their demand with other preexisting facilities) is

$$I^p(X) = \{i \in I : a_i(F_X) = a_i(F_A \cup F_B)\}. \tag{2}$$

Then the market share captured by the new facilities is given by

$$m(X) = \sum_{i \in I^t(X)} w_i + \sum_{i \in I^p(X)} \frac{n_i(X)}{n_i(X) + n_i(F_A \cup F_B)} \cdot w_i \tag{3}$$

Cannibalization

Before the expansion, the set of demand points which are totally captured by the firm A is

$$I^t = \{i \in I : a_i(F_A) > a_i(F_B)\} \tag{4}$$

and the set of demand points which share their buying power between firms A and B is

$$I^p = \{i \in I : a_i(F_A) = a_i(F_B)\}. \tag{5}$$

A part of the market share of the old facilities of the firm A may be lost due to the entrance of the new facilities. The lost market share is obtained as follows:

(i) if $i \in I^t$, the lost market share is

$$\begin{cases} w_i & \text{if } \ a_i(F_X) > a_i(F_A), \\ \frac{n_i(F_X)}{n_i(F_X)+n_i(F_A)} \cdot w_i & \text{if } \ a_i(F_X) = a_i(F_A); \end{cases} \tag{6}$$

(ii) if $i \in I^p$, the lost market share is

$$\begin{cases} \frac{n_i(F_A)}{n_i(F_A)+n_i(F_B)} \cdot w_i & \text{if } \ a_i(F_X) > a_i(F_A), \\ \frac{n_i(F_A)n_i(F_X)}{(n_i(F_A)+n_i(F_B))\cdot(n_i(F_X)+n_i(F_A)+n_i(F_B))} \cdot w_i & \text{if } \ a_i(F_X) = a_i(F_A). \end{cases} \tag{7}$$

Let

$$I^a_>(X) = \{i \in I^a : a_i(F_X) > a_i(F_A)\} \tag{8}$$

and

$$I^a_=(X) = \{i \in I^a : a_i(F_X) = a_i(F_A)\}, \tag{9}$$

where $a = t, p$.

Then the lost market share of the old facilities is given by

$$\begin{aligned} c(X) = & \sum_{i \in I^t_>(X)} w_i + \sum_{i \in I^t_=(X)} \frac{n_i(F_X)}{n_i(F_X)+n_i(F_A)} \cdot w_i \\ & + \sum_{i \in I^p_>(X)} \frac{n_i(F_A)}{n_i(F_A)+n_i(F_B)} \cdot w_i \\ & + \sum_{i \in I^p_=(X)} \frac{n_i(F_A)n_i(F_X)}{(n_i(F_A)+n_i(F_B))\cdot(n_i(F_X)+n_i(F_A)+n_i(F_B))} \cdot w_i. \end{aligned} \tag{10}$$

Bi-objective Problem

Thus the firm A is interested in finding the Pareto solutions to the following bi-objective optimization problem:

Maximize $m(X)$,
Minimize $c(X)$,
s.t. $X \subset L$, $|X| = r$.

To our knowledge, this bi-objective location problem for an expanding firm is considered for the first time in the location literature with the new tie breaking rule.

Approximation of the Pareto Front

The simplest way to determine the Pareto front is to enumerate all possible solutions and select non-dominated ones. The number of possible function evaluations is the number of combinations C_k^r, which means that this method can be very time consuming and sometimes it is impossible to consider (or even define) all possible solutions in a reasonable time. Moreover, for some practical problems it is not necessary to find the exact Pareto front, but rather its approximation. Therefore heuristic algorithms, such as MOEAs, can be used to solve multi-objective optimization problems.

Although there exist many techniques to design MOEAs which have different characteristics, the advantages of a particular technique can be better utilized by hybridizing two or more different techniques; see [15, 21, 23] for examples of hybrid MOEAs.

Non-dominated Sorting Genetic Algorithm

One of the first and best-known evolutionary algorithms for global multi-objective optimization is NSGA, which was presented by Srinivas and Deb [33]. However it had some drawbacks such as a high computational complexity of non-dominated sorting of population of candidate solutions, lack of elitism, and need for specifying a sharing parameter.

An updated version of the algorithm—NSGA-II—has been presented by Deb et al. [6]. The NSGA-II can be applied to solve various optimization problems as it requires a little knowledge about the problem being solved, as well as it is suitable for parallel computing; see [20] for the example of application of the NSGA-II to solve multi-objective competitive facility location problem using the high performance computing system.

The algorithm begins with an initial population Q_1 consisting of N randomly generated solutions. Genetic operators *crossover* and *mutation* are used to generate a child population Q_2, usually of the same size N as the parent population. The crossover operator is based on combining elements (variables) of two different solutions in order to obtain a third one, and the mutation is a small change of each variable with a predefined probability. A particular way to perform the crossover and mutation in solving the relevant discrete BCFLP will be described in detail in section "Application of NSGA-II to Solve BCFLP". After a new child population is generated, both populations Q_1 and Q_2 are combined into one population $Q = Q_1 \cup Q_2$, which is used to produce a new parent population for the next generation. The new population is formed using the specific *non-dominated sorting* procedure which is based on breaking down the general population Q into subsets of equally dominated (having the same number of dominators) individuals and selection of N individuals with the lowest number of dominators. In case of choosing between

equally dominated solutions, the *Crowding Distance* operator is applied to choose the most promising ones. The latter procedure as well as Crowding Distance operator is described in detail in [6].

Application of NSGA-II to Solve BCFLP

One of the main problems in the usage of evolutionary algorithms is representation of solutions—how the solutions will be represented in numerical fashion. Since the relevant optimization problem deals with the selection of a combination of different locations from a given set of candidates $L \subset I$, it is natural to represent the solution as a vector of indices of candidate locations for the new facilities. For example, in the case of selecting a combination of three locations from a set L consisting of 100 candidate locations, the solution x can be represented as a vector $x = (1, 15, 98)$, representing that the 1st, 15th, and 98th candidate locations in L will form the solution. Note that both objectives of the problem are commutative since two solutions $(1, 15, 98)$ and $(98, 1, 15)$ will provide the same market share and cannibalization. We will also assume that no more than one new facility can be located per candidate demand point, i.e., solutions representing two or more equal demand points are not feasible. These two main restrictions should be taken into account while generating a new solution.

An initial population Q_1 of N solutions is randomly generated taking into account the latter properties of the problem. Each member of a new child population Q_2 is generated by applying crossover and mutation operators to two different solutions $x^{(1)}$ and $x^{(2)}$ randomly chosen from parent population Q_1. Each element $x_i^{(3)}$ of the new child solution $x^{(3)} = (x_1^{(3)}, x_2^{(3)}, \ldots, x_r^{(3)})$ is chosen at random from $\{x_i^{(1)}, x_i^{(2)}\}$, where r denotes the number of problem variables (the number of new facilities expected to locate), and $i = 1, \ldots, r$.

The second step in generation of child solution is mutation which plays an important role in obtaining a good approximation of the Pareto front. The idea of mutation is to make changes to elements of a child solution to extend the variance in exploring the search space. Usually each element of the solution is mutated with a predefined probability called the *mutation rate*. A larger mutation rate leads to more significant mutation and more global search. On the other hand if the mutation rate is too small the search is restricted to only recombination of elements of the parent population. In our experimental investigation we will use the value of the mutation rate equal to $1/r$. Such a value of the mutation rate leads to an average mutation of one variable.

Mutation of the i-th element of the child solution $x_i^{(3)}$ is performed by changing the index of location $x_i^{(3)} \in L$ to the index of another location from L, which is not represented in $x^{(3)}$:

$$x_i^{(3)} \in \{l \in L : l \neq x_j^{(3)}, j = 1, \ldots, r\}. \tag{11}$$

On one hand the new location l can be chosen at random from all possible candidate locations; however, such a strategy leads to a chaotic mutation. We propose to define a neighborhood of $x_i^{(3)}$—a subset $L^{(h)}(x_i^{(3)}) \subset L$ of h locations which are nearest to the location $x_i^{(3)}$—and perform mutation within the neighborhood as follows:

$$x_i^{(3)} \in \{l \in L^{(h)}(x_i^{(3)}) : l \neq x_j^{(3)}, j = 1, \ldots, r\}. \tag{12}$$

An important factor is the value of h which can vary from 1 to $|L|$. Usage of large values leads to chaotic mutation similar as using (11), while usage of smaller values of h leads to inclusion of less scattered locations. An experimental investigation on different values of h has been made and its results will be discussed hereinafter.

Improvement by Local Search

In this section we will describe a strategy to apply a local search in solving discrete multi-objective optimization problems using the NSGA-II algorithm. Although the strategy could be applied to solve various discrete multi-objective optimization problems, in this paper we will focus on the previously described BCFLP.

Local search is based on selection of a neighbor solution and is aimed at improving the Pareto front obtained by the NSGA-II algorithm which was described in section "Non-dominated Sorting Genetic Algorithm". A single iteration of the local search algorithm consists of randomly choosing a single solution x from an approximation of the Pareto set $\tilde{P}$ found so far, and generating a neighbor solution x' using (12). Each variable of the solution is changed with a predefined probability equal to $1/r$. In the case of no change the generation procedure is repeated again till at least one variable is mutated. The subset $L^{(h)}(x)$ is constructed in the same way as for mutation in NSGA-II algorithm—indices of h locations from L which are nearest to the location we want to change. After a neighbor solution x' is generated and the values of objectives are evaluated, the dominance relation against other members of $\tilde{P}$ is evaluated. If x' is not dominated by any other member of the approximation of the Pareto set, it is added to $\tilde{P}$ and all solutions which are dominated by x' are removed from $\tilde{P}$:

$$\tilde{P} \leftarrow (\tilde{P} \cup \{x'\}) \setminus \{x \in \tilde{P} : x' \succ x\}. \tag{13}$$

Depending on the number of function evaluations we want to devote for the local search, the number of local search iterations must be defined in advance. Results of experimental studies in choosing an appropriate values of the parameter h and the number of local search iterations will be presented and discussed hereinafter.

Numerical Experiments

Description

The performance of the algorithm has been experimentally investigated by solving the BCFLP formulated in section "The Location Problem". Real geographical and population data of 6961 municipalities in Spain has been used (see Fig. 1).

It was supposed that each of two firms has three preexisting facilities located in the most populated demand points. In such a case there are $C_6^3 = 20$ possibilities to locate the preexisting facilities each of which has been investigated. A set of candidate demand points L consists of required number of the remaining largest demand points including those in which facility of the firm B is already located.

Four different configurations of parameters of the problem have been investigated (see Table 1) which vary on the number of facilities expected to locate (the number of variables, r) and the number of candidate locations (size of the search space, k).

The value of attraction that the i-th demand point feels to the j-th facility has been calculated by

$$a_{ij} = \frac{q_j}{1 + d_{ij}}, \tag{14}$$

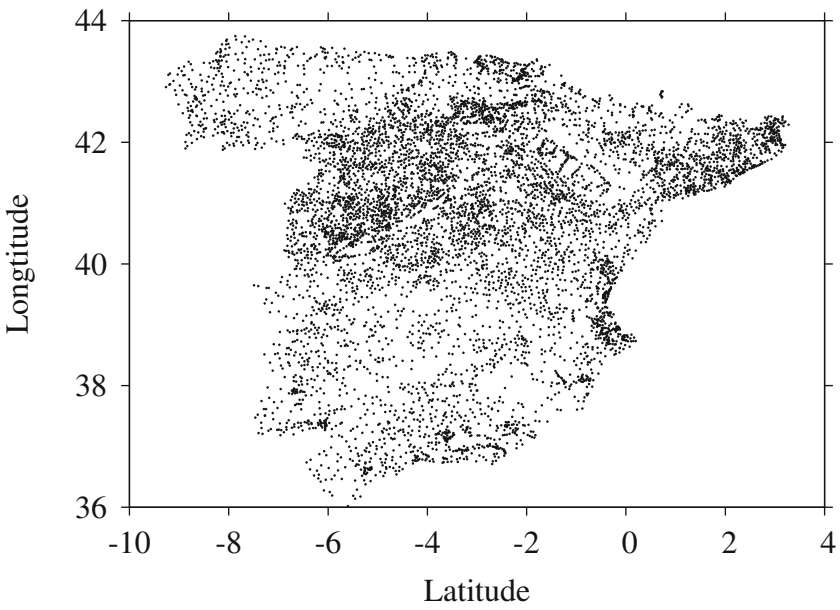

Fig. 1 Illustration of experimental data: 6961 municipalities in Spain

Table 1 Four configurations of the parameters of the BCFLP

	I	II	III	IV
r	3	3	5	10
k	100	500	100	500

where d_{ij} is a distance between the i-th demand point and the j-th facility, and q_j is a predefined value of quality of j-th facility. Since we want the impact of quality would be competitive with the impact of the distance, and assume that the customers go up to 100 km to purchase a service, the quality values of all preexisting facilities should be chosen from the interval $[30, 70]$. The values equal to $(57, 60, 59, 45, 56, 36)$ have been randomly chosen for our experimental investigation. All new facilities have been assumed to be of the same quality q_n, but different values have been investigated: $35, 45, 55$, and 65. In particular case, if the quality values of all facilities (new and preexisting) are equal, the patronizing behavior becomes distance-based, which means that all customers from a particular demand point are served by their closest facility. This variant of the model has been included in the experimental investigation as well.

Metrics of Performance

Performance of the algorithms has been evaluated by two performance metrics: Hyper-Volume and Inverted Generational Distance (IGD).

The hyper-volume [38] metric measures the volume of a region made by the members of the obtained approximation of the Pareto front (in space of objectives) and the given reference point. The larger HV value means the better quality of the approximation of the Pareto front. The difference between the HV of the optimization problems containing different Pareto fronts we use Inverted Normalized Hyper-Volume (INHV) metric, which is described as

$$\mathrm{INHV} = \frac{\mathrm{HV_T} - \mathrm{HV_O}}{\mathrm{HV_T}}, \tag{15}$$

where $\mathrm{HV_T}$ denotes the HV of the true Pareto front, and $\mathrm{HV_O}$—of the obtained approximation—both are normalized so that the true Pareto front would fit to a square $[0, 1]^2$. All points which, after the normalization, belong to the approximation, but do not fit in the rectangle $[0, 1]^2$, are considered to be out of range and are not included when evaluating the INHV. The reference point $\mathbf{r} = (-0.1, 1.1)$ is used for the evaluation of $\mathrm{HV_T}$ and $\mathrm{HV_O}$. In contrast with the regular HV, smaller values of INHV means better precision of approximation $\tilde{P}$, and $\mathrm{INHV} = 0$ means that $\tilde{P} = P$. The difference between the HV of the normalized true Pareto and the normalized its approximation is illustrated in Fig. 2.

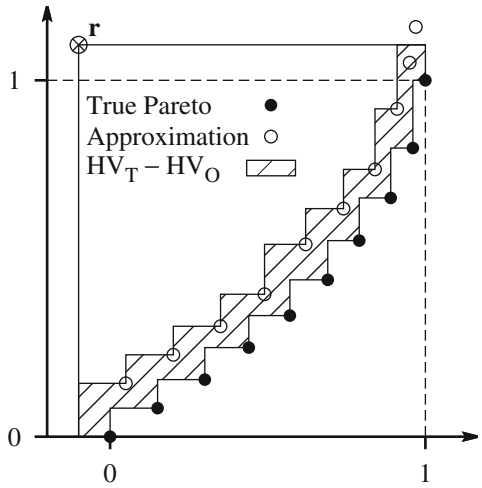

Fig. 2 Illustration of the concept of the INHV

IGD measures the average distance from the element of the true Pareto front P to the nearest element of the approximated Pareto front $\tilde{P}$ [37]:

$$\mathrm{IGD}(\tilde{P},P)=\frac{1}{|P|}\sum_{\mathbf{x}'\in P}\min_{\mathbf{x}\in\tilde{P}}\parallel F(\mathbf{x}')-F(\mathbf{x})\parallel, \tag{16}$$

where $F(\cdot)$ denotes a vector of the values of the objective functions. Since it is expected to find decision vectors which would be as close as possible to the members of the Pareto-optimal front, lower IGD value implies better quality of the approximation.

Results

A complete enumeration algorithm has been used to determine the true Pareto fronts of the BCFLP with the first three configurations of the parameters. The optimization problem seems to be quite simple when using the first configuration of parameters (see Table 1)—it requires C^3_{100} function evaluations to perform the complete enumeration. However it requires around 15 min using an Intel(R) Core(TM) i5 CPU 760 @ 2.80 GHz hardware. Increasing the value of k to 500 (second configuration of the parameters), the number of function evaluations increases to C^5_{500} and the complete enumeration lasts around 18 h. Increasing the number of facilities expected to locate from 3 to 5 (third configuration of the parameters), duration of computations increases to around 78 h.

NSGA-II with population size of 100 individuals and pure random mutation has been run for 250 generations to investigate the opportunities to approximate the Pareto fronts. Due to stochastic nature of the genetic algorithm, each experiment has been run for 100 times and average values of the performance metrics have been evaluated.

Results of the investigation showed that the Pareto front of any of the first three cases of the problem parameters can be quite precisely approximated by a classical version of NSGA-II within 25,000 function evaluations: the maximum values of INHV and IGD were less than 0.01 and 0.004, respectively. Comparison of the true and obtained Pareto fronts is illustrated in Fig. 3, where circles indicate points of the true Pareto front found by complete enumeration and crosses—their approximations.

More complicated was the BCFLP with the last configuration of the parameters—expecting to choose locations for 10 facilities from the set of 500 candidates. Since it was almost impossible to perform the complete enumeration in reasonable time, an approximation of the true Pareto front has been constructed from all approximations (around 1500) obtained by all experiments made during the investigation. Depending on the parameters of behavior of customers, the number of the Pareto-optimal solutions in the approximation varies from 703 ($q_n = 35$) to 71,730 ($q_n = 65$). When the behavior of customers is distance-based the approximation consists of 937 Pareto-optimal solutions. The average values of INHV and IGD were 0.056 and 0.0144, respectively.

In order to improve these values of performance metrics we proposed a local mutation strategy (see (12)) with different values of the parameter h: 3, 5, 10, 20, and 30. The results of the investigation are presented in Fig. 4, where average values of the INHV (uppermost) and the IGD (lowermost) with the confidence intervals with confidence level 0.05 are presented; different columns represent different models of the behavior of customers: distance-based and attractive-based with $q_n = 35$, $q_n = 45$, $q_n = 55$, and $q_n = 65$, respectively. One can see in the figure that change of the value has significant impact on both metrics of performance, but it is worth to use $h = 10$ as the usage of it gave the lowest values of INHV and IGD metrics for all models of customers' behavior investigated.

The same BCFLP has been solved by NSGA-II with the proposed local search strategy as described in section "Improvement by Local Search". The number of function evaluations devoted for the local search is another parameter of the algorithm. Different values of this parameter have been experimentally investigated: 1000, 5000, 10,000, and 15,000. In order to perform 25,000 function evaluations in total, the local search has been performed after performance of 24,000, 20,000, 15,000, and 10,000 function evaluations by NSGA-II, respectively. The parameter value $h = 10$ has been used as this choice gives the best results in previous experiments. Results of the investigation presented in Fig. 5 show that devotion of function evaluations to the local search can significantly improve the quality of the final Pareto front approximation. From the results we can see that it is worth to devote 5000–10,000 function evaluations.

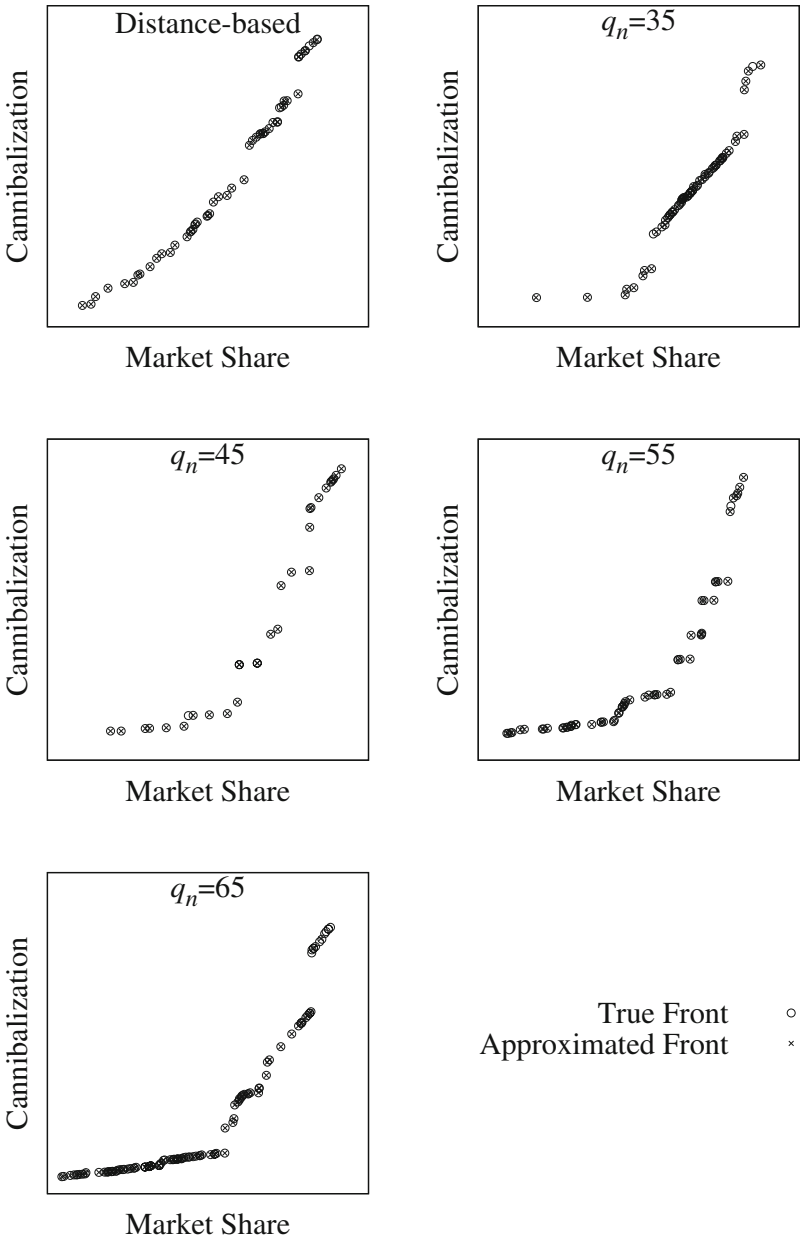

Fig. 3 Examples of the true Pareto fronts of the problem with different values of the parameters of customers' behavior, and their approximations

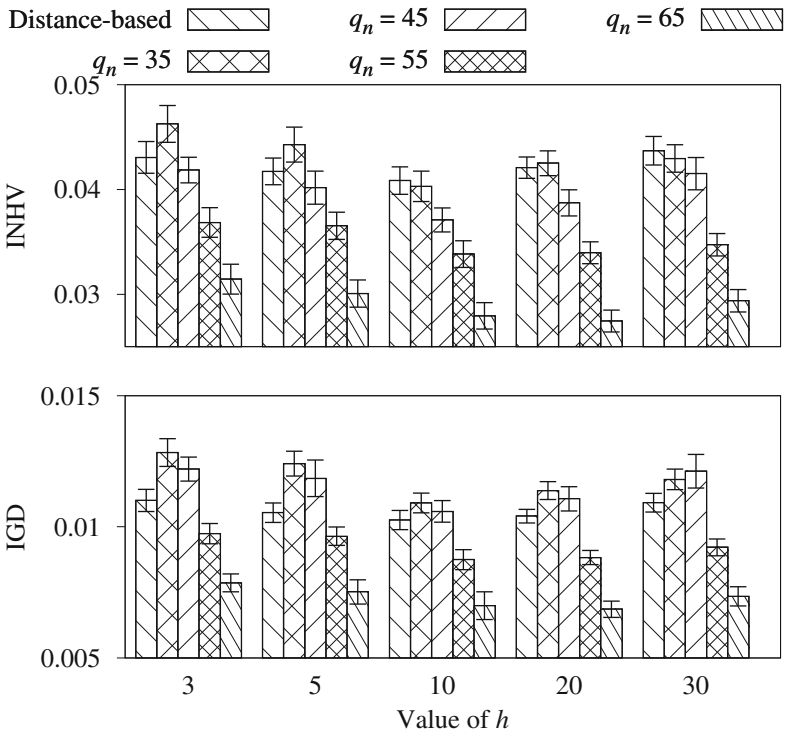

Fig. 4 Average values of INHV (*uppermost*) and IGD (*lowermost*) and their confidence intervals with confidence level 0.05, obtained using different values of parameter h

The average duration of the heuristic algorithm is around 173 s using an Intel(R) Core(TM) i5 CPU 760 @ 2.80 GHz hardware. The duration is independent on whether the classical NSGA-II or one of its improved versions (with the local mutation or with the local mutation and the local search) is used as the main computational effort that is allocated for the evaluation of the objective functions' values.

Conclusions

In this paper we have studied the location problem for an expanding firm with the aim of market share maximization and cannibalization minimization where the customers choose the most attractive facility to be served. A new tie breaking rule based on the allocation of variable market share to the tied facilities, which depends on the number of tied facilities, has been introduced.

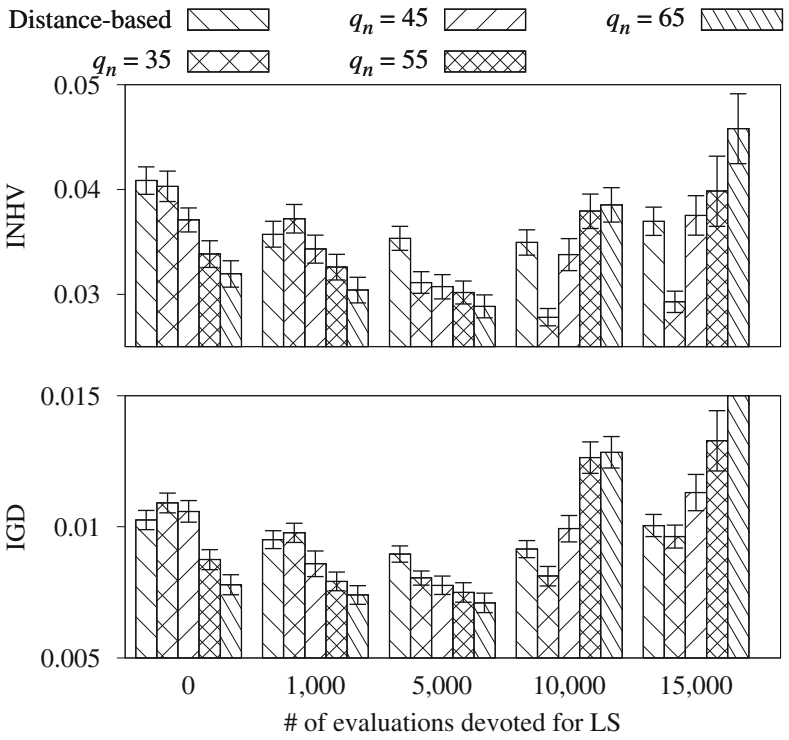

Fig. 5 Average values of INHV (*uppermost*) and IGD (*lowermost*) and their confidence intervals with confidence level 0.05, obtained using NSGA-II with the local mutation and different number of function evaluations devoted for the local search

In order to solve the resulting hard multi-objective optimization problem, Non-dominated Sorting Genetic Algorithm (NSGA-II) has been applied by proposing a new mutation strategy, specially adapted to solve facility location problems. The special local search strategy has been also proposed and incorporated in NSGA-II developing a hybrid global optimization algorithm for facility location problems.

Results of experimental investigation of the proposed algorithm showed that usage of the proposed strategies for mutation and local search in NSGA-II can improve the hyper-volume and IGD metrics from 10 to 50 % (depending on the problem instance), comparing with the values of the performance metrics, obtained by the classical version of NSGA-II.

Acknowledgements This research has been supported by the Ministry of Economy and Competitiveness of Spain (MTM2015-70260-P), the Program to Support Research of the Seneca Foundation (The Agency of Science and Technology of the Region of Murcia, 19241/PI/14).

References

1. Aboolian, R., Berman, O., Krass, D.: Competitive facility location and design problem. Eur. J. Oper. Res. **182**(1), 40–62 (2007)
2. Berman, O., Krass, D.: Locating multiple competitive facilities: spatial interaction models with variable expenditures. Ann. Oper. Res. **111**, 197–225 (2002)
3. Chinchuluun, A., Pardalos, P.M.: A survey of recent developments in multiobjective optimization. Ann. Oper. Res. **154**(1), 29–50 (2007)
4. Chinchuluun, A., Pardalos, P.M., Migdalas, A., Pitsoulis, L. (eds.): Pareto Optimality, Game Theory and Equilibria. Springer Optimization and Its Applications, vol. 17. Springer, New York (2008)
5. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York, NJ (2007)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002)
7. Doerner, K.F., Gutjahr, W.J., Nolz, P.C.: Multi-criteria location planning for public facilities in tsunami-prone coastal areas. OR Spectrum **31**(3), 651–678 (2009). doi:10.1007/s00291-008-0126-7. http://dx.doi.org/10.1007/s00291-008-0126-7
8. Drezner, T., Drezner, Z.: Finding the optimal solution to the Huff based competitive location model. Comput. Manag. Sci. **1**(2), 193–208 (2004)
9. Farahani, R.Z., SteadieSeifi, M., Asgari, N.: Multiple criteria facility location problems: a survey. Appl. Math. Modell. **34**(7), 1689–1709 (2010). doi:10.1016/j.apm.2009.10.005. http://www.sciencedirect.com/science/article/pii/S0307904X09003242
10. Farahani, R.Z., Rezapour, S., Drezner, T., Fallah, S.: Competitive supply chain network design: an overview of classifications, models, solution techniques and applications. Omega **45**(0), 92–118 (2014)
11. Fernández, J., Pelegrín, B., Plastria, F., Tóth, B.: Planar location and design of a new facility with inner and outer competition: an interval lexicographical-like solution procedure. Netw. Spat. Econ. **7**, 19–44 (2007)
12. Francis, R.L., Lowe, T.J., Tamir, A.: Demand point aggregation for location models. In: Drezner, Z., Hamacher, H. (eds.) Facility Location: Application and Theory, pp. 207–232. Springer, Berlin (2002)
13. Friesz, T.L., Miller, T., Tobin, R.L.: Competitive networks facility location models: a survey. Pap. Reg. Sci. **65**, 47–57 (1998)
14. Ghosh, A., Craig, C.S.: FRANSYS: a franchise distribution system location model. J. Retail. **67**(4), 466–495 (1991)
15. Goel, T., Deb, K.: Hybrid methods for multi-objective evolutionary algorithms. In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning, pp. 188–192 (2002)
16. Hakimi, L.: Location with spatial interactions: competitive locations and games. In: Drezner, Z. (ed.) Facility Location: A Survey of Applications and Methods, pp. 367–386. Springer, Berlin (1995)
17. Huapu, L., Jifeng, W.: Study on the location of distribution centers: a bi-level multi-objective approach. In: Logistics, pp. 3038–3043. American Society of Civil Engineers (2009)
18. Huff, D.L.: Defining and estimating a trade area. J. Market. **28**, 34–38 (1964)
19. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evol. Comput. **8**(2), 149–172 (2000)
20. Lančinskas, A., Žilinskas, J.: Solution of multi-objective competitive facility location problems using parallel NSGA-II on large scale computing systems. In: Manninen, P., Oster, P. (eds.) Applied Parallel and Scientific Computing. Lecture Notes in Computer Science, vol. 7782, pp. 422–433. Springer, Berlin, Heidelberg (2013). doi:10.1007/978-3-642-36803-5_31
21. Lančinskas, A., Ortigosa, P.M., Žilinskas, J.: Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm. Nonlinear Anal.: Modell. Control **18**(3), 293–313 (2013)

22. Liao, S.H., Hsieh, C.L.: A capacitated inventory-location model: formulation, solution approach and preliminary computational results. In: Chien, B.C., Hong, T.P., Chen, S.M., Ali, M. (eds.) Next-Generation Applied Intelligence. Lecture Notes in Computer Science, vol. 5579, pp. 323–332. Springer, Berlin, Heidelberg (2009)
23. Medaglia, A.L., Villegas, J.G., Rodríguez-Coca, D.M.: Hybrid biobjective evolutionary algorithms for the design of a hospital waste management network. J. Heuristics **15**(2), 153–176 (2009)
24. Peeters, P.H., Plastria, F.: Discretization results for the Huff and Pareto-Huff competitive location models on networks. Top **6**, 247–260 (1998)
25. Pelegrín, B., Fernández, P., García, M.D.: On tie breaking in competitive location under binary customer behavior, OMEGA-International Journal of Management Science **52**, 156–167 (2015)
26. Plastria, F.: Static competitive facility location: an overview of optimisation approaches. Eur. J. Oper. Res. **129**(3), 461–470 (2001)
27. Plastria, F.: Avoiding cannibalization and/or competitor reaction in planar single facility location. J. Oper. Res. Soc. Jpn. **48**, 148–157 (2005)
28. Redondo, J.L., Fernández, J., Álvarez, J.D., Arrondoa, A.G., Ortigosa, P.M.: Approximating the Pareto-front of continuous bi-objective problems: application to a competitive facility location problem. In: Casillas, J., Martínez-López, F.J., Corchado Rodríguez, J.M. (eds.) Management Intelligent Systems. Advances in Intelligent Systems and Computing, vol. 171, pp. 207–216. Springer, Berlin, Heidelberg (2012)
29. ReVelle, C.S., Eiselt, H.A., Daskin, M.S.: A bibliography for some fundamental problem categories in discrete location science. Eur. J. Oper. Res. **184**(3), 817–848 (2008)
30. Schaffer, J.D., Grefenstette, J.J.: Multi-objective learning via genetic algorithms. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence – Volume 1, IJCAI'85, pp. 593–595. Morgan Kaufmann Publishers, San Francisco, CA (1985)
31. Serra, D., Colomé, R.: Consumer choice and optimal locations models: formulations and heuristics. Pap. Reg. Sci. **80**(4), 439–464 (2001)
32. Serra, D., ReVelle, C.: Competitive location in discrete space. In: Drezner, Z. (ed.) Facility Location: A Survey of Applications and Methods, pp. 367–386. Springer, Berlin (1995)
33. Srinivas, N., Deb, K.: Multiobjective optimization using Nondominated Sorting in Genetic Algorithms. Evol. Comput. **2**, 221–248 (1994)
34. Suárez-Vega, R., Santos-Penate, D.R., Dorta-Gonzalez, P.: Discretization and resolution of the $(r|X_p)$-medianoid problem involving quality criteria. Top **12**(1), 111–133 (2004)
35. Suárez-Vega, R., Santos-Penate, D.R., Dorta-González, P.: The follower location problem with attraction thresholds. Pap. Reg. Sci. **86**(1), 123–137 (2007)
36. Villegas, J.G., Palacios, F., Medaglia, A.L.: Solution methods for the bi-objective (cost-coverage) unconstrained facility location problem with an illustrative example. Ann. Oper. Res. **147**, 109–141 (2006)
37. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: Proceedings of the Congress on Evolutionary Computation (CEC), pp. 3234–3241. IEEE Press, New York (2006)
38. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Eiben, A., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) Parallel Problem Solving from Nature — PPSN V. Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer, Berlin, Heidelberg (1998)
39. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. Trans. Evol. Comput. **3**(4), 257–271 (1999)
40. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., Tsahalis, D.T., Périaux, J., Papailiou, K.D., Fogarty, T. (eds.) Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, pp. 95–100 (2001)
41. Zopounidis, C., Pardalos, P.M. (eds.): Handbook of Multicriteria Analysis. Applied Optimization, vol. 103. Springer, Berlin, Heidelberg (2010)

On Sampling Methods for Costly Multi-Objective Black-Box Optimization

Ingrida Steponavičė, Mojdeh Shirazi-Manesh, Rob J. Hyndman, Kate Smith-Miles, and Laura Villanova

Abstract We investigate the impact of different sampling techniques on the performance of multi-objective optimization methods applied to costly black-box optimization problems. Such problems are often solved using an algorithm in which a surrogate model approximates the true objective function and provides predicted objective values at a lower cost. As the surrogate model is based on evaluations of a small number of points, the quality of the initial sample can have a great impact on the overall effectiveness of the optimization. In this study, we demonstrate how various sampling techniques affect the results of applying different optimization algorithms to a set of benchmark problems. Additionally, some recommendations on usage of sampling methods are provided.

Keywords Design of experiment • Space-filling • Low-discrepancy • Efficient global optimization

Introduction

A plethora of practical engineering problems involve multiple conflicting objectives which have to be optimized simultaneously. Solving such problems requires more effort than single-objective optimization as they usually have many (possibly infinite) optimal solutions; such solutions compose the so-called *Pareto optimal set*.

To add further to the challenge, for many real-world optimization problems there is also an absence of algebraic objective or response function definitions. Examples are crash tests, chemical reactions, many laboratory experiments, etc. Therefore

I. Steponavičė (✉) • M. Shirazi-Manesh • K. Smith-Miles • L. Villanova
School of Mathematical Sciences, Monash University, Wellington Road, Clayton, VIC 3800, Australia
e-mail: ingrida.steponavice@monash.edu; mojdeh.manesh-shirazi@monash.edu; kate.smith-miles@monash.edu; laura.villanova@monash.edu

R.J. Hyndman
Department of Econometrics and Business Statistics, Monash University, Wellington Road, Clayton, VIC 3800, Australia
e-mail: rob.hyndman@monash.edu

P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_15

an important challenge in optimization practice is how to solve an optimization problem in the absence of an algebraic model of the system to be optimized. Such optimization problems are called *black-box* as the available information is just input–output data without prior knowledge of the characteristics or physics of the relationships involved.

Due to the lack of an analytical description of the objective functions, derivatives are unavailable and derivative-based optimization methods cannot be used. Moreover, in many practical applications, the objective functions (or the associated constraints) are very costly to evaluate and it is desirable to limit the number of evaluations. Consequently, for *costly black-box* multi-objective optimization problems, the main concern is to find the Pareto optimal set with as few function evaluations as possible. Traditional derivative-free methods based on direct search or gradient estimation via numerical differentiation are not usually viable as they require many more function evaluations than can be comfortably afforded.

A popular and successful approach for derivative-free optimization of costly black-box functions is to construct response surface models known as *surrogate models* (or metamodels) that mimic the behavior of the real-word process as closely as possible while being less resource-demanding to evaluate. Surrogate-based optimization methods became popular a few decades ago even though they were proposed much earlier [20]. Among the various potential surrogate models, polynomial response surface models [3], kriging [36], and radial basis functions (RBF) [6] are widely used in solving costly black-box optimization problems.

In recent years, much attention has been devoted to develop multi-objective optimization methods (e.g., see [31, 41, 49]) to deal with real-world applications characterized as *costly multi-objective black-box* optimization problems using surrogate models to replace the unknown objective functions. However, little attention has been focused so far on the impact of the initial sample on the performance of the developed algorithms. Every black-box optimization algorithm starts the optimization process with an initial sample, usually a very limited one in the case of expensive function evaluations. The initial sample provides some knowledge for the method to further investigate the decision space with the aim of finding the global optimum. When the evaluation of objective functions is costly, these evaluated points are usually fed to a surrogate model to predict the real response function values of unevaluated points. An inexpensive surrogate model is constructed based on an initial sample; the model is then used in a search for the next points to evaluate. This approach decreases the number of resource-consuming function evaluations, but suggests that the initial sample selected to build a surrogate model can strongly impact the efficiency of optimization. This consideration motivates our analysis of the sampling effect on the optimization search.

Sampling methods have been used for a wide range of purposes ranging from censuses and surveys [1, 48] to numerical and computational studies [5, 9] to experimental investigations in industry and science [11, 24, 29, 35]. In general, sampling methods can be used in two main types of studies: observational and experimental studies [4]. Observational studies aim to draw inferences about an entire space from a sample [34], whereas experimental studies aim to identify the

cause–effect relationship between input and output variables through controlled experiments [40]. In the first case, sampling methods must provide a representative sample of the entire space; in the second case, sampling methods must provide a small informative sample selected from the set of feasible experiments (the decision space). It is the latter experimental scenario that is relevant when using sampling methods in an optimization context.

To our knowledge, there exist only a few studies investigating the impact of sampling methods in the context of single-objective surrogate-based optimization [26, 44] and multi-objective optimization [30]. With regard to the multi-objective field, Poles et al. [30] focused on evolutionary optimization algorithms. Evolutionary algorithms require thousands of function evaluations to achieve a good approximation of the Pareto optimal set; therefore, they are not suitable for costly optimization. Instead, we focus on methodologies that require hundreds of function evaluations. Indeed, this study aims to investigate the effect and importance of initial sampling techniques on methods suitable for *costly multi-objective black-box* optimization.

The remainder of this chapter is as follows. In section "Problem Description", we recall the basic concepts related to black-box and multi-objective optimization. Widely used sampling methods and the concepts behind them are outlined in section "Sampling Methods". In section "Experiments", we present the experimental setup used in the study and we illustrate the experimental results. Section "Conclusions" summarizes the results obtained, provides insights and suggestions for future research directions, and draws some final conclusions.

Problem Description

The *multi-objective optimization problem* comprises multiple objective functions which are to be minimized simultaneously. It can be expressed in the following form:

$$\min \boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\big)^T \qquad \text{subject to } \boldsymbol{x} \in S, \tag{1}$$

where $S \subset \mathbb{R}^d$ is the feasible set and $f_i :\to \mathbb{R}$, $i = 1, \ldots, m$ ($m \geq 2$), are objective functions to be minimized simultaneously. All objective functions are represented by the vector-valued function $\boldsymbol{f} : S \to \mathbb{R}^m$. A vector $\boldsymbol{x} \in S$ is called a *decision vector* and a vector $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$ is called an *objective vector*.

We assume that at least one of the functions f_i is "costly"; that is, its evaluation requires a significant amount of resources and no analytic expression is available. Therefore, problem (1) is called a *costly multi-objective black-box optimization problem*.

In multi-objective optimization, the objective functions $f_1, \ldots, f_m$ in (1) are typically conflicting. In that case, there does not exist a decision vector $\bar{\boldsymbol{x}} \in S$ such that $\bar{\boldsymbol{x}}$ minimizes f_i in S for all $i = 1, \ldots, m$, but there exist a number (possibly infinite)

of Pareto optimal solutions. In mathematical terms, a decision vector $\bar{\boldsymbol{x}} \in S$ and its image $\bar{\boldsymbol{z}} = \boldsymbol{f}(\bar{\boldsymbol{x}})$ are said to be *Pareto optimal* or *nondominated* if there does not exist a decision vector $\boldsymbol{x} \in S$ such that $f_i(\boldsymbol{x}) \leq f_i(\bar{\boldsymbol{x}})$ for all $i = 1, \ldots, m$ and $f_j(\boldsymbol{x}) < f_j(\bar{\boldsymbol{x}})$ for some $j = 1, \ldots, m$. If such a decision $\boldsymbol{x} \in S$ does exist, $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{z}}$ are said to be *dominated* by $\boldsymbol{x}$ and its image $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$, respectively. The Pareto optimal set in the objective space is also called the *Pareto optimal front*.

Sampling Methods

Sampling methods for experimental studies have been attracting a great deal of attention since the 1800s and have resulted in a dedicated field of research known as Design of Experiments. Their importance directly relates to the efficient collection of informative data, allowing for the quick delivery of robust results. This translates into considerable savings that minimize costs and time related to both physical (real-world) and computer-based experimentation.

Many sampling methods assume that the unknown objective function can be approximated by a simple model (e.g., linear or quadratic) and recommend samples located on the boundary of the design space. This assumption can be safely made if some knowledge exists of the objective function or if the approximation occurs locally (i.e., in a relatively small sub-area of the decision space) [17]. In black-box optimization, no knowledge exists regarding the objective function and the entire decision space is typically searched. Therefore, sampling methods are required to provide samples that are spread out across the entire decision space. Two such classes of methods are *space-filling methods* and *low-discrepancy sequences*. *Space-filling methods* aim to generate widespread samples using a range of different criteria including equally spaced intervals and distance measures. On the other hand, *low-discrepancy sequences* use a measure of uniformity (discrepancy) that minimizes the difference between the percentage of points falling in a particular region on a unit cube and the percentage of volume occupied by this region. The main space-filling designs and low-discrepancy sequences are reviewed below and investigated in our computational study.

Simple Random Sampling

In simple random sampling (SRS), N decision vectors are randomly sampled from the decision space [39]. Decision vectors have the same probability of being chosen; the constant chance of selection extends to pairs, triplets, and so on (e.g., any given pair of decision vectors has the same chance of selection as any other pair).

SRS is among the most popular sampling methodologies thanks to its simplicity and low computational demand. One drawback of SRS relates to its vulnerability to sampling error; indeed, the randomness of its selection process may result in

a sample that is not evenly spread throughout the entire decision space. This is particularly true for small samples in high-dimensional regions that often exhibit apparent clustering and poorly covered regions [37]. Systematic and stratified sampling techniques have been developed to overcome this issue and choose a "more representative" sample.

Latin Hypercube Sampling

Latin hypercube sampling (LHS) is a stratified sampling technique. LHS controls how random samples are generated from a given probability distribution (usually uniform). To generate a sample of N decision vectors, the domain of each decision variable is divided into N equally spaced and non-overlapping intervals; then, one value is selected at random from each such interval. Random permutation of the resulting values for all decision variables results in a random Latin hypercube sample.

LHS originated in 1979 for computer-based experiments in order to address the need for a better and more efficient coverage of the decision space [22]. The authors showed that LHS reduces the variance in their chosen application of Monte Carlo integration.

The main advantage of LHS over SRS derives from its one-dimensional projection property: a Latin hypercube sample projected into one dimension results in a set of evenly distributed points in all dimensions separately. Due to this property, LHS is the most commonly used stratified sampling technique in many areas of computer-based experiments. Despite this, different studies show that it is not always the best choice [43, 44]. Indeed, LHS does not guarantee a uniform coverage (i.e., a good spread) of the decision space as (sometimes large) areas of the decision space might remain unexplored. Two such examples are reported in Fig. 1 showing an LHS in two dimensions with six intervals per decision variable. In both cases, there is a large area of the decision space that is not explored; therefore, if we use such a sample to develop a prediction model, then the prediction will be poor in those unexplored areas. In the worst case scenario (b), LHS can generate a sample with two perfectly correlated decision variables; such a sample causes the effects of the two variables to be completely confounded. To overcome these limitations, LHS methods have been improved through the adoption of an additional criterion; such improvements resulted in maximin distance and minimum correlation LHS (C-LHS) methods described later on.

Maximin Sampling

Maximin sampling belongs to the class of distance-based sampling methods. Distance-based sampling methods make use of the Euclidean distance to prevent

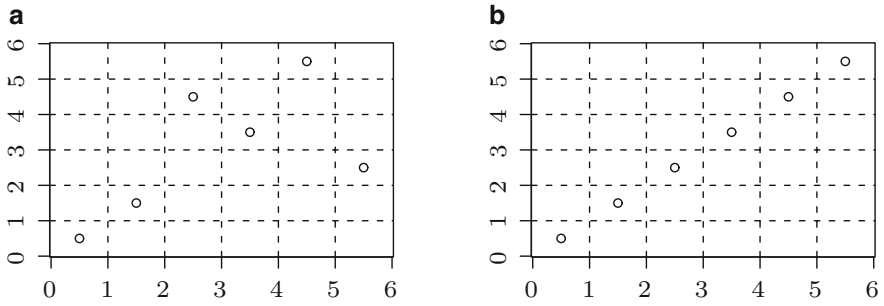

Fig. 1 Two LHS configurations with two variables in six intervals (**a**) a non space-filling LHS (**b**) a worst case LHS

sampled points from clustering too close together so that they over-represent some regions of the design space.

The aim of maximin sampling is to scatter points in the decision space such that the minimal pairwise distance between points is maximized. Let $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$ and $\boldsymbol{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kd})$ be two different decision vectors in a sample $D(N,S)$, where N is the sample size and S is the d-dimensional feasible set. The following mathematical problem must be solved:

$$\max \min s^2(\boldsymbol{x}_j, \boldsymbol{x}_k) \tag{2}$$

where

$$s^2(\boldsymbol{x}_j, \boldsymbol{x}_k) = \sum_{i=1}^{d} \left(\frac{x_{ji} - x_{ki}}{U_i - L_i} \right)^2$$

and $\boldsymbol{x}_j, \boldsymbol{x}_k \in D(N,S)$, $j,k = 1,2,\ldots,N$ $(j \neq k)$, U_i and L_i are the upper and lower limits of the ith variable. Therefore, a maximin sample of size N contains minimum pairwise distances that are maximum compared to any other N-sized sample.

Maximin sampling was first introduced by Johnson et al. [14]. It is among the best methods to obtain an even coverage of the decision space. However, it tends to prioritize decision vectors that are located near the boundary of the decision space. Also, despite computational efficiency in low dimensions, the method is very demanding in high dimensions. To overcome this issue, various approximate maximin sampling methods have been developed. Approximate methods use conventional nonlinear programming algorithms to reduce the computational cost of the procedure at the expense of potentially providing solutions that are not globally optimal, only locally optimal.

Here, we propose a simple approximation method we call "Nearly maximin" consisting of the following steps:

Step 1 Randomly generate a decision vector $\boldsymbol{x}_1$ from the decision space and choose it as the first element of the sample;

Step 2 Randomly generate n decision vectors from the decision space and for each vector, calculate the Euclidean distance to the closest element of the existing sample; and

Step 3 Choose the one having the maximum distance out of the n decision vectors as the next element of the sample;

Repeat Step 2 and Step 3 until the sample comprises N decision vectors.

In a preliminary study that will be published elsewhere, the Nearly maximin method showed extremely promising results. It will be used in our computational study to allow for the investigation of high-dimensional test problems.

Maximin LHS

Both LHS and maximin sampling produce samples with attractive properties. LHS guarantees that the one-dimensional projection of the sample presents an even spread in the variables' domains; maximin guarantees that no two elements (decision vectors) in the sample are close together. However, both methods suffer from limitations. In particular, LHS might occasionally generate samples with points that are close to each other as in the examples of Fig. 1, whereas maximin tends to select samples that are located near the boundary of the decision space.

To overcome these limitations, Morris and Mitchel [25] suggested that LHS be combined with the maximin criterion. The resulting method is known as Maximin LHS (M-LHS). It consists of (a) generating the maximum number of possible LHS samples, (b) measuring their maximin distances, and (c) selecting the most evenly spread sample (optimal sample).

M-LHS preserves the one-dimensional projection property of LHS while ensuring that no two points in the LHS design are very close to each other. Therefore, a good spread of decision vectors is achieved not just in each single variable domain but also in the entire decision space. Also, the decision vectors in the sample are preferentially located in the interior of the decision space thus providing a compromise between maximin property and good projective properties in each dimension (as guaranteed by Latin hypercubes) [25]. Unfortunately, constructing samples by M-LHS can be quite time consuming when the number of dimensions and design points increase. Indeed, there exist $(N!)^{d-1}$ LHS samples for N divisions and d dimensions; for each such sample, the maximin distances need to be calculated in order to identify the optimal one.

Correlation LHS

To find optimal LHS Iman and Conover [12], Owen [28], and Tang [42] proposed to use a criterion minimizing correlation between the factors. This is useful in applications requiring a sample to be composed of decision vectors without (or with small) correlation. Owen proposed to measure the goodness of LHS with respect to a criterion of minimum pairwise correlations which is defined as follows:

$$\rho^2 = \frac{\sum_{i=2}^{d} \sum_{j}^{i-1} \rho_{ij}^2}{d(d-1)/2}, \tag{3}$$

where ρ_{ij} is the pairwise correlation between columns i and j of the design, and $\rho_{ij} \in [0,1]$. The smaller the ρ^2 is, the weaker the pairwise correlation is.

In C-LHS method suggested by Owen [28], the sum of between-column squared correlation is decreased by alternating forward and backward Gram–Schmidt orthogonalization. In our computational study we used the Matlab implementation of Owen's method.

One might think that minimizing the correlation should spread out the points and maximizing the distance between the points should reduce the correlation. However in practice, there is no one-to-one relationship between the two, and designs obtained by these two criteria can be quite different [15]. In other words, C-LHS not necessarily provides a well spread sample.

Halton Sequence Sampling

The Halton sequence sampling method generates quasi-random numbers of high-dimensionality with a high level of uniformity across the space. Halton sequence is constructed according to a deterministic method that uses different prime bases for different dimensions to create a d-dimensional low-discrepancy sequence [7, 19, 21]. The method is based on the fact that each non-negative integer can be expanded using a prime base. Construction of Halton sequence in d-dimensional space is as follows:

1. Choose d prime integers $p_1, p_2, \ldots, p_d$ (usually the first primes $p_1 = 2, p_2 = 3, \ldots,$ are selected).
2. To generate the i-th sample, consider the base p representation for i in which:

$$i = a_0 + a_1 p + a_2 p^2 + a_3 p^3 + \ldots$$

   where each a_j is an integer in $[0, p-1]$.
3. The next point in $[0,1]$ is achieved by reversing the order of the bits and moving the decimal point:

$$r(i,p) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \frac{a_3}{p^4} + \dots$$

4. Starting from $i = 0$, the i-th sample in the Halton sequence is

$$(r(i,p_1), r(i,p_2), \dots, r(i,p_d)) \tag{4}$$

Halton is an extension of the Van der Corput sequence, which was originally introduced for one dimension and a base of 2. The Van der Corput sequence is obtained by using $p = 2$. However, Halton sequences based on large primes $(d > 10)$ can be highly correlated, and their coverage can be worse than that of the pseudo-random uniform sequences.

Hammersley Sequence Sampling

The Hammersley sequence [8, 21] belongs to the class of low-discrepancy sequences, and is closely related to the Fibonacci series. The Hammersley sequence is an adaptation of the Halton sequence (4) when the required sample size N is known. In such a case, a better uniformly distributed sample can be obtained by using only $d - 1$ distinct primes. In a Hammersley sequence with N elements and starting from $i = 0$, the i-th d-dimensional vector will be

$$\left(\frac{i}{N}, r(i,p_1), r(i,p_2), \dots, r(i,p_{d-1})\right) \quad \text{for } i = 0,1,2,\dots,N-1. \tag{5}$$

Hammersley sequence sampling provides better uniformity properties over LHS [23]; in particular, the chance of samples with clustered decision vectors is lower. Also, compared to other conventional techniques, Hammersley sampling requires far smaller samples to approximate the mean and variance of distributions based on empirical studies [16].

Sobol Sequence Sampling

Sobol sequence sampling is an improved version of the Halton and Hammersley methods. Indeed, despite the Halton and Hammersley methods being relatively simple and efficient, they suffer from a common pitfall—the performance of these two sampling methods degrades substantially in higher dimensions. Sobol sequences have been proposed to approximate the integral over the d-dimensional unit cube:

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1}f(x_i)=\int_{[0,1]^d}f(x)\,\mathrm{d}x$$

where f is a real integrable function over a d-dimensional unit hypercube and $x_0,\ldots,x_{n-1}$ are n points in $[0,1]^d$ comprising a "Sobol sequence." The Sobol sequence, as originally defined by Sobol [38], is generated from a set of special binary vectors of length w bits, v_i^j, $i=1,2,\ldots,w$, $j=1,2,\ldots,d$. These numbers, v_i^j, are called direction numbers. To generate them for dimension j, one should begin with a primitive polynomial over the finite field $\mathscr{F}_2$ with elements $\{0,1\}$. Let us assume that the primitive polynomial is

$$p_j(x)=x^q+a_1x^{q-1}+\cdots+a_{q-1}x+1.$$

Then we use its coefficients to define a recurrence relation for calculating v_i^j, the direction number in dimension j. It is generated using the following q-term recurrence relation:

$$v_i^j(x)=a_1v_{i-1}^j\oplus a_2v_{i-2}^j\oplus\cdots\oplus a_{q-1}v_{i-q+1}^j\oplus v_{i-q}^j\oplus(v_{i-q}^j/2^q),$$

where $i>q$, $\oplus$ denotes the bitwise XOR operation, and the last term is v_{i-q} shifted right q places. The initial numbers $v_1^j\cdot 2^w, v_2^j\cdot 2^w,\ldots,v_q^j\cdot 2^w$ can be arbitrary odd integers smaller than $2,2^2,\ldots,2^q$, respectively. The Sobol sequence x_n^j ($n=\sum_{i=0}^{w}b_i2^i$, $b_i\in\{0,1\}$) in dimension j is generated by

$$x_n^j=b_1v_1^j\oplus b_2v_2^j\oplus\cdots\oplus b_wv_w^j.$$

Different primitive polynomials should be used to generate Sobol sequence in each dimension. Currently there are more efficient ways of generating Sobol sequences proposed in the literature (see, e.g., [27]).

Summary of Sampling Methods

This section outlines the main characteristics of sampling methods discussed above. Table 1 summarizes the sampling methods in terms of their main features. Samples consisting of 32 points in two-dimensional space, and generated by different sampling methods, are presented in Fig. 2 for a visual comparison.

An important consideration relates to the methods' computational cost in the context of costly optimization. For those methods that demand moderate to intensive computational efforts, it is important to investigate the compromise between (a) the time required to generate the sample and (b) the sample quality. The sample quality is its ability to decrease the number of further function evaluations without affecting the results of the optimization procedure. Obviously, if function evaluations are

Table 1 Characteristics of sampling methods

Sampling method	Simple	Low computational cost	One-dimensional projection	Uniform coverage	Suitable for high dimensions	Stochastic	Additional features
Random	+	+	–	–	–	+	Minimum bias
LHS	+	+	+	–	–	+	Stratified
Maximin	–	–	–	+	–	+	Maximum minimum distance
M-LHS	–	–	+	+	–	+	Stratified, maximum minimum distance
C-LHS	–	–	+	–	–	+	Stratified, low correlation
Halton	+	+	–	+	–	–	Minimum discrepancy
Hammersley	+	+	–	+	–	–	Minimum discrepancy
Sobol	–	+	–	+	+	–	Minimum discrepancy

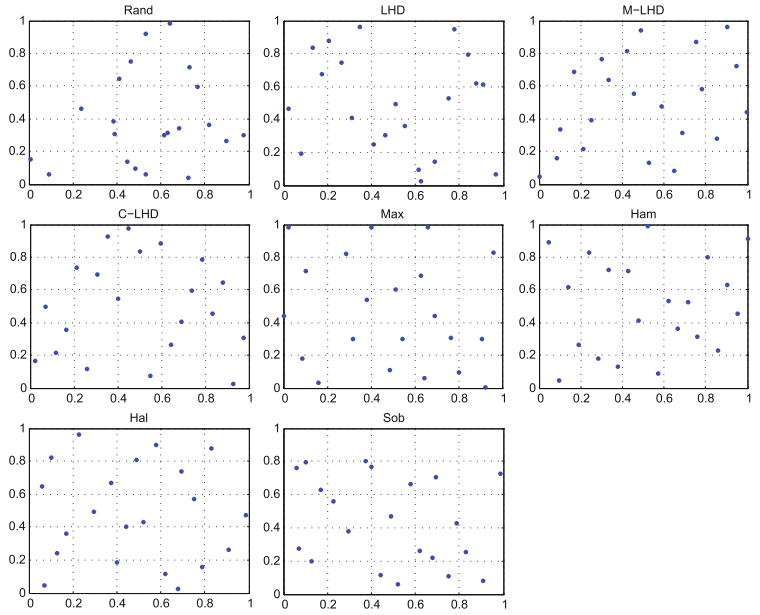

Fig. 2 Initial sample in 2D, number of points = 32

highly costly and involve resources other than time, even a small decrease in the number of function evaluations justifies the higher computational time required to generate the optimal sample.

Experiments

The impact of sampling method on multi-objective optimization algorithm efficiency is evaluated by means of a comprehensive benchmark problem set. The design of the experimental study is described in section "Experimental Setup". Section "Test Problems" gives an overview of the benchmark problems. The major part of this section is devoted to discussion of the obtained results and the appropriate observations. This is covered in section "Results".

Optimization Algorithms Considered

In our study, we considered three algorithms designed for costly multi-objective optimization problems, namely ParEGO, SMS-EGO, and ϵ-EGO. These algorithms were selected due to their available implementation in the R package **mlrMBO** [2].

ParEGO is a state-of-art algorithm developed by Knowles [18]. It uses the augmented Tchebycheff norm to convert a multi-objective problem into a scalarized

one:

$$f_\lambda(\boldsymbol{x}) = \max_{j=1,\ldots,m} \left(\lambda_j f_j(\boldsymbol{x})\right) \pm \rho \sum_{j=1}^{m} \lambda_j f_j(\boldsymbol{x}), \tag{6}$$

where $\rho > 0$ is a small positive number and λ is a weight vector. ParEGO randomly selects w from a uniformly distributed set in each iteration. Then a surrogate model is fitted to the respective scalarized problem. At each iteration of the algorithm, a different weight vector is drawn uniformly at random from the set of evenly distributed vectors allowing the model to gradually build up an approximation to the true Pareto set. Before scalarization, the objective functions are normalized with respect to the known (or estimated) limits of the objective space to the range $[0, 1]$. At each iteration, the method uses a genetic algorithm to search for the solutions that maximize an infill criterion, called expected improvement, with respect to a surrogate model. Only the best solution is evaluated on the actual problem. After evaluation of the selected solution on the real expensive function, ParEGO updates the GP surrogate model of the landscape and repeats the same steps.

The other two algorithms do not convert a multi-objective optimization problem to a single optimization problem but use a multi-objective optimization of infill criteria on each objective in order to obtain a candidate set for evaluation. SMSEGO [31] optimizes the hypervolume and ϵ-EGO [45] looks at search solutions with respect to the additive ϵ-indicator which has been introduced by Zitzler et al. [51]. An additive ϵ-indicator of approximation set A gives the minimum value ϵ by which each point in the real front R can be added such that resulting transformed approximation is dominated by A.

Test Problems

The test set consists of different benchmark problems with a variety of characteristics in both the decision and objective spaces. The objectives of test problems can be either unimodal (U) or multimodal (M). Multimodal problems are more difficult than unimodal problems, and more representative of real-world problems. The Pareto optimal front can be convex, linear, concave, disconnected, or some combination of the former. It is well known that the type of Pareto front can directly affect the performance of the optimization algorithms. For example, disconnected Pareto fronts can increase the likelihood that an algorithm will fail to find all regions of the Pareto optimal front. The fitness landscape may be one-to-one or many-to-one and the latter property impacts some algorithms' ability to find multiple, otherwise equivalent optima. For a more detailed discussion on test problems properties we refer readers to [10].

Our test set includes the following benchmark problems:

- *OKA2* $m = 2, d = 3$. The true Pareto optimal set for this problem is a spiral shaped curve in the decision space, and the density of the Pareto optimal solutions in the objective space is low.
- *Kursawe* This problem has a scalable number of decision variables. In our experiment we used $d = 3$, $m = 2$. Its Pareto optimal set is disconnected and symmetric in the decision space, and disconnected and concave in the objective space.
- *Viennet* $m = 3, d = 2$. The true Pareto optimal set is convex in the objective space.
- *ZDT family*: ZDT problems share such characteristics as multimodality, discontinuity, and possession of multiple Pareto fronts; for all problems, $m = 2$ and d is scalable, however we used d values suggested by the authors.
  - ZDT1: $d = 30$; Pareto optimal set in the objective space is convex.
  - ZDT2: $d = 30$; Pareto optimal set in the objective space is nonconvex.
  - ZDT3: $d = 30$; Pareto optimal set is disconnected in both objective and decision spaces. Pareto optimal set consists of one mixed convex/concave component and several convex components in the objective space.
  - ZDT4: $d = 10$; first objective function is unimodal, while the second objective function has multiple local optima and therefore is highly multimodal. Its Pareto optimal set in the objective space is convex [10].
  - ZDT6: $d = 10$; it has a nonuniform search space, i.e., the Pareto optimal solutions in the decision space are non-uniformly distributed along the global Pareto set, and also the density of the solutions is lowest near the Pareto optimal set and highest away from it. Pareto optimal set in the objective space is concave.
- DTLZ1: It is a scalable problem in both objective and decision space and has multiple global optima. Thus, the only difficulty provided by this problem is convergence to the Pareto optimal hyperplane. We solved three sizes of this problem: (1) $m = 4$ and $d = 13$; (2) $m = 6$ and $d = 15$; and (3) $m = 8$ and $d = 17$.

The major characteristics of the selected benchmark problems are summarized in Table 2.

Performance Assessment

In multi-objective optimization, the definition of solution quality is substantially more complex than for single-objective problems as the optimization goal itself consists of several objectives such as convergence to the true Pareto frontier, uniform distribution of obtained nondominated solutions, and maximum extent of obtained nondominated set with respect to each objective. Therefore, a number of quality metrics usually taking into account one solution quality characteristic have been proposed (see, e.g., [13, 47]). The most widely used performance metric is a hypervolume (HV) indicator (also known as an *S*-metric) [50] which defines the

Table 2 Summary of test problems characteristics

Problem	No of objectives	No of variables	Modality	Convex	Concave	Disconnected	Linear	Pareto many-to-one
OKA2	Bi-objective	3	U	+	–	–	–	–
Kursawe		3	U	–	+	+	–	–
ZDT1		30	U	+	–	–	–	–
ZDT2		30	U	–	+	–	–	–
ZDT3		30	M	+	–	+	–	–
ZDT4		10	M	+	–	–	–	–
ZDT6		10	M	–	+	–	–	+
Viennet	3	2	U	+	–	–	–	–
DTLZ1	4	13	M	–	–	–	+	+
DTLZ1	6	15	M	–	–	–	+	+
DTLZ1	8	17	M	–	–	–	+	+

size of the region dominated by the relevant Pareto set approximation. As such it provides information about closeness and diversity at the same time. In addition, it possesses a desirable property: whenever one approximation completely dominates another approximation, the HV of the former will be greater than the HV of the latter [52]. The HV metric corresponds to the size of the region of the objective space bounded by a reference point. In our study, we calculated the HV metric using normalized values of the objective functions.

Experimental Setup

In this study we control: (a) the size of the initial sample, (b) the optimization budget, (c) the dimension of decision space, and (d) the dimension of objective space.

The initial design size was set to $n_{\text{init}} = 11d - 1$ based on the recommendations in [18]. An example of the initial samples generated by different sampling methods for two decision variables is given in Fig. 2. The number of optimization iterations was restricted to 200 resulting in a total budget of $n_{\text{total}} = 200 + 11d - 1$. Taking into account the different number of dimensions, the algorithms were evaluated on the 11 test problems discussed in section "Test Problems".

The Pareto front approximations of the algorithms were compared not only at the last iteration ($n = 200$) but as well at intermediate iterations ($n = 50$, 100, and 150) with respect to the HV metric. For each test function the reference point was estimated based on the nondominated set of initial samples.

With regard to the sampling methods, it is important to point out the following aspects. It can be computationally very expensive to achieve optimal maximin and low C-LHS samples; therefore, we have chosen the best sample out of 1000 randomly generated LHS samples with regard to the corresponding criterion (maximin or minimum correlation). The low-discrepancy sampling methods (i.e., Hammersley, Halton, and Sobol sequences) are deterministic (rather than stochastic) as there is no run-to-run difference between generated samples; therefore, we have used the "random-start sequence" trick [46]. By defining random starts for generation of these samples, the sequences differ in each run resulting in stochasticity of the samples.

Results

This section elaborates on a detailed analysis of the results from the experiments performed. In each numerical experiment an initial sample was generated by one sampling method running it 100 times. Then an optimization search was performed by each algorithm over these runs. Their performance has been estimated with respect to the HV metric.

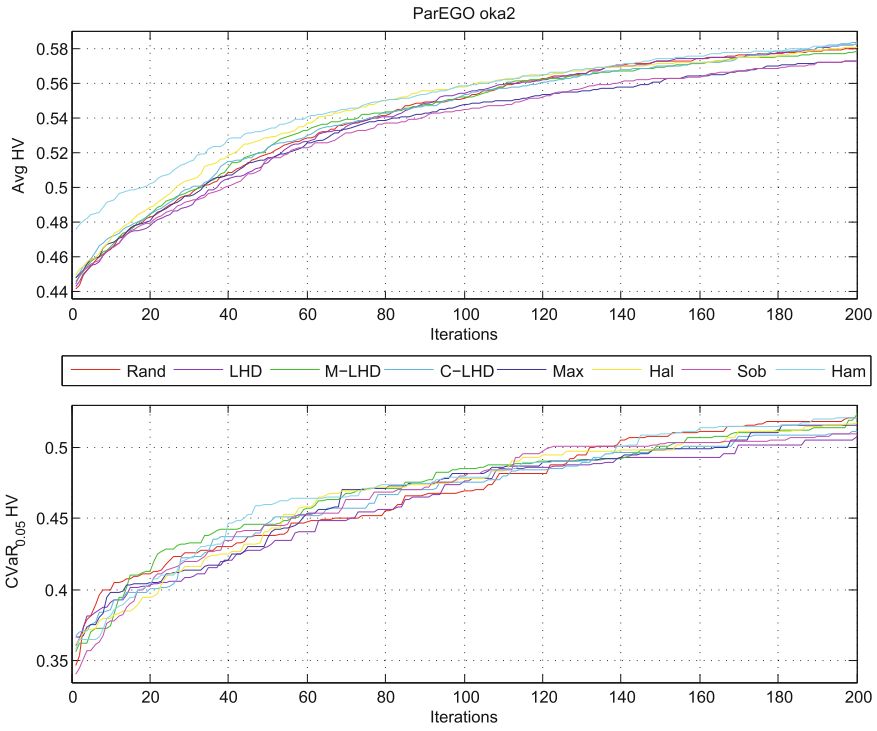

Fig. 3 ParEGO performance on OKA2 problem

Corresponding results of each sampling method were compared to those of the other seven methods, to determine if its results had a statistically significant advantage. Comparisons were performed using the unpaired *t*-test [32] and differences were deemed statistically significant at the 0.01 significance level. The significance was tested multiple times, i.e., after 50, 100, 1500, and 200 function evaluations.

The average performance of optimization algorithms against sampling methods on different test problems is represented in Figures 3, 4, 5, 6, and 7. However, in real-world applications, to know the average performance is not enough and usually the worst case scenarios are taken into account as well. The worst case scenario provides some additional information but sometimes it is considered as too conservative. Instead of the worst case scenario, we may want to know the mean of the realizations above a specified quantile; i.e., the conditional value-at-risk (CVaR) introduced in [33]. We calculated CVaR with a selected confidence level of 0.05, giving the average value over a distribution tail consisting of the 5 % worst realizations. Due to space limitations, we could not provide the graphs of all the optimization algorithms and test problems considered. Therefore, we have selected the ones providing most of the information and supporting the main observations.

We have noticed that the largest variability over 100 runs is produced by Hammersley, Sobol, and Halton sequences which by nature are deterministic

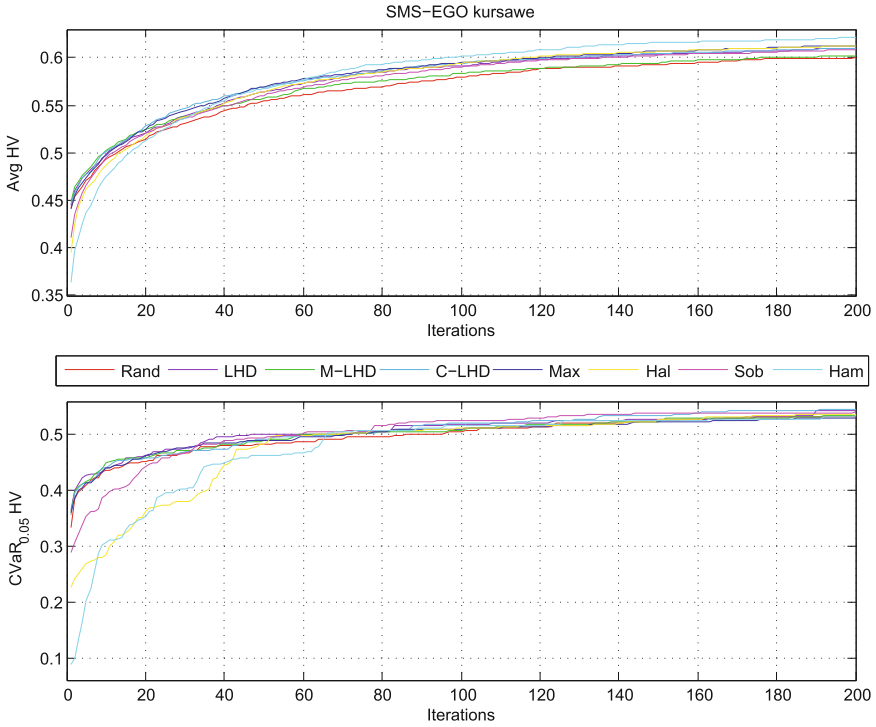

Fig. 4 SMS-EGO performance on Kursawe problem

methods as sequences are finite. However, in order to generate 100 runs we used Matlab functions with various *leap* and *skip* parameters values which produce different subsets of these sets, sometimes not fully covering the whole decision space. To our knowledge, there is no recommendation on how to select these parameters. Therefore, the average performance of the deterministic sampling methods is influenced by some initial samples not spread throughout the entire space.

According to the obtained results regarding the average performance of the optimization algorithms based on a normalized HV metric, we can observe some trends. Generally, it can be noticed that sampling methods do not affect optimization algorithm performance significantly (the difference of HV metric values over 100 runs is not statistically significant at the 1 % level) on the problems with objective and decision space dimensions both lower or equal to three. Also, we discovered that algorithm performance is very similar when using samples generated by stochastic sampling methods (namely, SRS, LHS, M-LHS, and C-LHS) on the bi-objective problems with high-dimensional decision space, and there is no statistically significantly difference among them as illustrated in Figs. 5 and 6. The ParEGO method with initial samples generated by LHS has not demonstrated the best performance on any single problem, while its improved versions (i.e., M-LHS and C-LHS) have

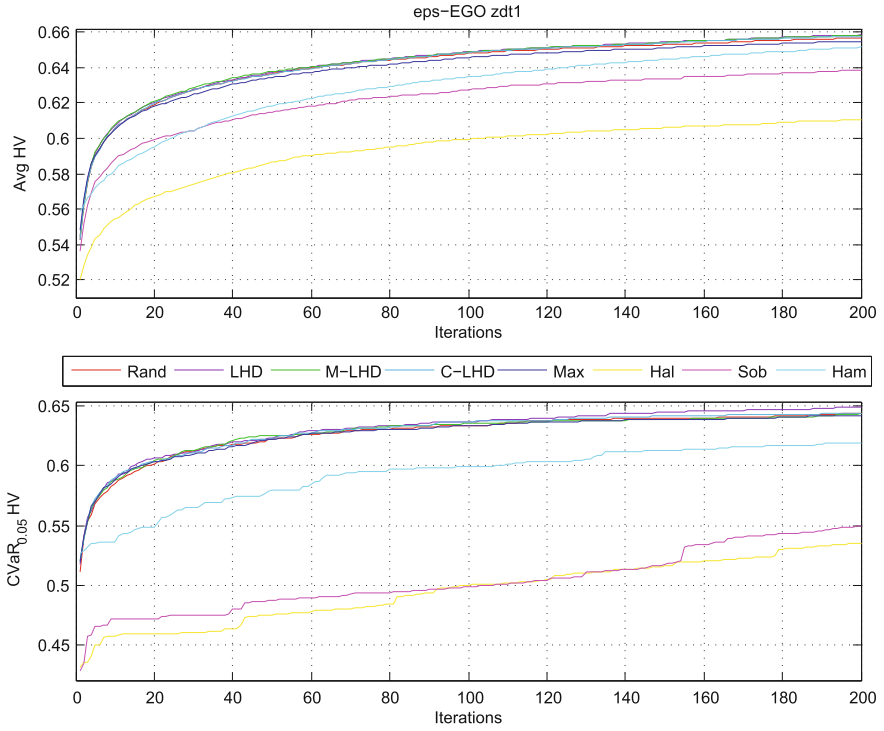

Fig. 5 ϵ-EGO performance on ZDT1 problem

shown very good performance on bi-objective problems with a larger number of decision variables. Hammersley sequence sampling and ParEGO showed the best performance or close to it on unimodal bi-objective problems of low-dimensionality with continuous Pareto front. Halton sequence sampling in conjunction with any of the considered optimization algorithms in most of the cases performed poorly, especially with a larger number of decision variables, except for low-dimensional problems with a convex Pareto front. Also, it has been outperformed by the other two deterministic techniques quite a number of times. M-LHS sampling technique paired with SMS-EGO proved to behave well on the problems with a larger amount of decision variables and is outperformed by other sampling techniques on smaller problems. All optimization algorithms demonstrated better average performance on problems with more than three objectives when using maximin for initial sampling (see, e.g., Fig. 7). SRS method can be considered an appropriate choice because the performance of selected optimization algorithms in most cases was not significantly worse than other stochastic sampling methods. Although, there is one exception for Kursawe problem optimized with SMS-EGO algorithm (see Fig. 4), where its initial samples lead to the statistically significant worst performance with respect to Hammersley and Halton sequences as well as maximin sampling method.

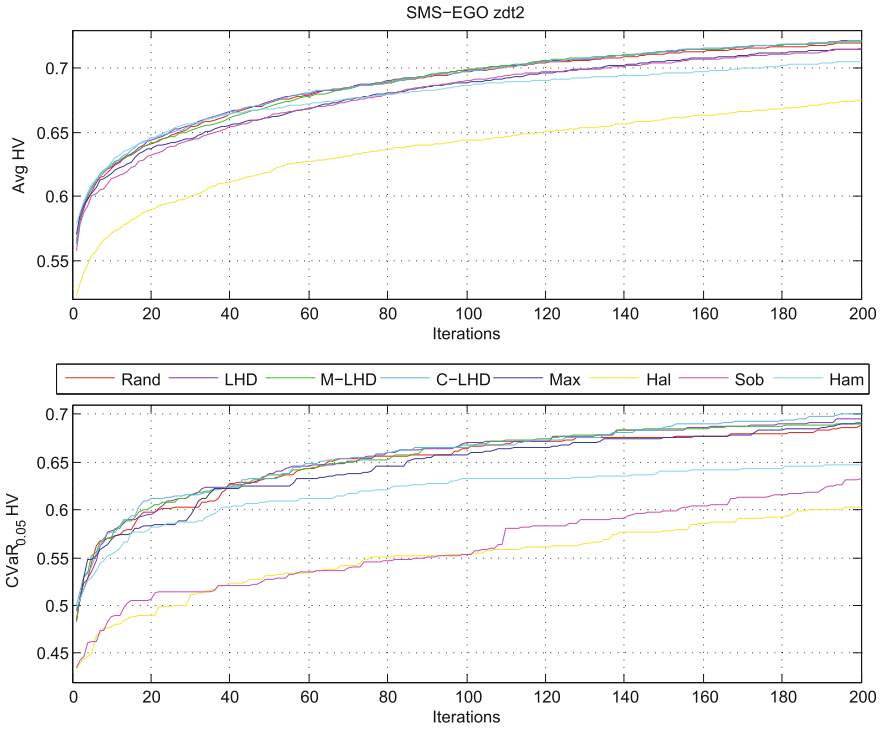

Fig. 6 SMS-EGO performance on ZDT2 problem

Conclusions

This section draws some conclusions and provides some recommendations based on the experiments performed and the results obtained. In addition, we shed some light on the impact of the selected sampling techniques on costly black-box multi-objective optimization, and discuss some future research questions which we leave for further investigation.

To summarize, sampling methods have no statistically significant impact (with a significance level 0.01) on the algorithm performance measured by the HV metric for low-dimensional problems, i.e., $m, d \leq 3$. Therefore, SRS can be considered as an appropriate choice for low-dimensional problems.

Also, when using deterministic sampling methods, one has to check that the initial sample is a good representative sample in the sense of covering the entire decision space. Otherwise a "bad" initial sample can cause the optimization outcome to deteriorate significantly; i.e., the variance of the deterministic methods is larger than the stochastic sampling methods. Although, LHS is often used as a default sampling method in multi-objective optimization, the obtained results did not confirm it to outperform other sampling methods; one could use M-LHS or C-LHS

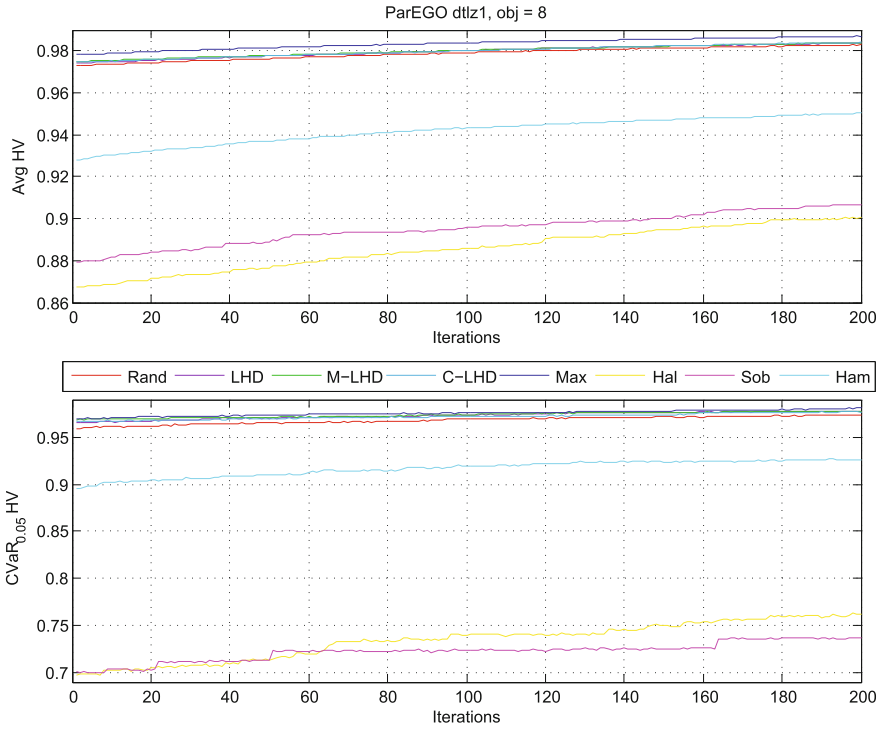

Fig. 7 ParEGO performance on DTLZ1 problem with 8 objectives and 17 decision variables

instead as these sampling methods obtain better results in many cases. For high-dimensional problems, in both objective and decision spaces, deterministic methods led to large variability which resulted in significantly lower average algorithm performance compared to stochastic sampling methods. In particular, the maximin sampling method outperformed other stochastic methods though this advantage was not statistically significant.

We plan to continue research on a larger set of test problems with a larger number of both objective and decision variables possessing a variety of properties. Hopefully, this will provide greater insights and enable us to determine more concrete recommendations. Our conclusion for now is that choice of initial sample matters in higher dimensions. In this work, we have studied the algorithm performance with respect to the most widely used performance metric HV. It would be interesting to investigate the impact of sampling methods with respect to other metrics. Moreover, the question of what initial sample size one should use and how it affects the optimization results is also open. Clearly, there is a trade-off involved between the size of an initial sample and the number of evaluations used to run an optimization algorithm when dealing with costly real-world optimization problems. Thus, this research direction will be considered in the future as well.

Acknowledgements This research was partly financially supported by the Linkage project "Optimizing experimental design for robust product development: a case study for high-efficiency energy generation" funded by the Australian Research Council.

References

1. Altmann, J.: Observational study of behavior: sampling methods. Behaviour **49**(3), 227–266 (1974)
2. Bischl, B., Bossek, J., Horn, D., Lang, M.: mlrMBO: Model-Based Optimization for MLR (2015). R package v1.0. https://github.com/berndbischl/mlrMBO
3. Box, G.E., Draper, N.R.: Empirical Model-Building and Response Surfaces. Wiley, New York (1987)
4. Cox, D.R., Reid, N.: The Theory of the Design of Experiments. CRC, Boca Raton (2000)
5. Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. Stat. Comput. **10**(3), 197–208 (2000)
6. Fang, H., Horstemeyer, M.F.: Global response approximation with radial basis functions. Eng. Optim. **38**(04), 407–424 (2006)
7. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Math. **2**(1), 84–90 (1960)
8. Hammersley, J.M.: Monte Carlo methods for solving multivariable problems. Ann. N. Y. Acad. Sci. **86**(3), 844–874 (1960)
9. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
10. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
11. Ilzarbe, L., Álvarez, M.J., Viles, E., Tanco, M.: Practical applications of design of experiments in the field of engineering: a bibliographical review. Qual. Reliab. Eng. Int. **24**(4), 417–428 (2008)
12. Iman, R.L., Conover, W.: A distribution-free approach to inducing rank correlation among input variables. Commun. Stat. Simul. Comput. **11**(3), 311–334 (1982)
13. Jiang, S., Ong, Y.S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Trans. Cybern. **44**(12), 2391–2404 (2014)
14. Johnson, M.E., Moore, L.M., Ylvisaker, D.: Minimax and maximin distance designs. J. Stat. Plann. Infer. **26**(2), 131–148 (1990)
15. Joseph, V.R., Hung, Y.: Orthogonal-maximin Latin hypercube designs. Stat. Sin. **18**(1), 171 (2008)
16. Kalagnanam, J.R., Diwekar, U.M.: An efficient sampling technique for off-line quality control. Technometrics **39**(3), 308–319 (1997)
17. Khuri, A.I., Mukhopadhyay, S.: Response surface methodology. Wiley Interdiscip. Rev. Comput. Stat. **2**(2), 128–149 (2010)
18. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. **10**(1), 50–66 (2006)
19. Kocis, L., Whiten, W.J.: Computational investigations of low-discrepancy sequences. ACM Trans. Math. Softw. **23**(2), 266–294 (1997)
20. Kushner, H.J.: A versatile stochastic model of a function of unknown and time varying form. J. Math. Anal. Appl. **5**(1), 150–167 (1962)
21. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
22. McKay, M.D., Beckman, R.J., Conover, W.J.: Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)

23. Meckesheimer, M., Booker, A.J., Barton, R.R., Simpson, T.W.: Computationally inexpensive metamodel assessment strategies. AIAA J. **40**(10), 2053–2060 (2002)
24. Montgomery, D.C.: Design and Analysis of Experiments. Wiley, Hoboken (2008)
25. Morris, M.D., Mitchell, T.J.: Exploratory designs for computational experiments. J. Stat. Plann. Infer. **43**(3), 381–402 (1995)
26. Müller, J., Shoemaker, C.A.: Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. J. Glob. Optim. **60**(2), 123–144 (2014)
27. Niederreiter, H.: Random number generation and quasi-Monte Carlo methods. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63. SIAM, Philadelphia (1992)
28. Owen, A.B.: Controlling correlations in Latin hypercube samples. J. Am. Stat. Assoc. **89**(428), 1517–1522 (1994)
29. Panse, V.G., Sukhatme, P.V.: Statistical Methods for Agricultural Workers. Indian Council of Agricultural Research, New Delhi (1954)
30. Poles, S., Fu, Y., Rigoni, E.: The effect of initial population sampling on the convergence of multi-objective genetic algorithms. In: Multiobjective Programming and Goal Programming, pp. 123–133. Springer, Berlin (2009)
31. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathscr{S}$-metric selection. In: Parallel Problem Solving from Nature–PPSN X, pp. 784–794. Springer, Berlin (2008)
32. Rice, J.: Mathematical Statistics and Data Analysis. Cengage Learning, Belmont (2006)
33. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**, 21–42 (2000)
34. Rosenbaum, P.R.: Observational Studies. Springer, New York (2002)
35. Roy, R.K.: Design of Experiments Using the Taguchi Approach: 16 Steps to Product and Process Improvement. Wiley, New York (2001)
36. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**, 409–423 (1989)
37. Santner, T.J., Williams, B.J., Notz, W.I.: Space-filling designs for computer experiments. In: The Design and Analysis of Computer Experiments. Springer Series in Statistics, pp. 121–161. Springer, New York (2003)
38. Sobol, I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput. Math. Math. Phys. **7**, 86–112 (1967)
39. Starnes, D.S., Tabor, J., Yates, D., Moore, D.S.: The Practice of Statistics, New York, 5th edn. W.H. Freeman (2014)
40. Steinberg, D.M., Hunter, W.G.: Experimental design: review and comment. Technometrics **26**(2), 71–97 (1984)
41. Steponavičė, I., Hyndman, R.J., Smith-Miles, K., Villanova, L.: Efficient identification of the Pareto optimal set. In: Learning and Intelligent Optimization, pp. 341–352. Springer, New York (2014)
42. Tang, B.: Selecting Latin hypercubes using correlation criteria. Stat. Sin. **8**(3), 965–977 (1998)
43. Tenne, Y.: An analysis of the impact of the initial sample on evolutionary metamodel-assisted optimization. Appl. Artif. Intell. **27**(8), 669–699 (2013)
44. Tenne, Y.: Initial sampling methods in metamodel-assisted optimization. Eng. Comput. **31**(4), 661–680 (2015)
45. Wagner, T.: Planning and multi-objective optimization of manufacturing processes by means of empirical surrogate models. Vulkan, Essen (2013)
46. Wang, X., Hickernell, F.J.: Randomized Halton sequences. Math. Comput. Model. **32**(7), 887–899 (2000)
47. Wu, J., Azarm, S.: Metrics for quality assessment of a multiobjective design optimization solution set. Math. Comput. Model. **123**(1), 18–25 (2001)
48. Yates, F.: Sampling Methods for Censuses and Surveys. Charles Griffin & Co. Ltd., London (1949)
49. Žilinskas, A.: A statistical model-based algorithm for 'black-box' multi-objective optimisation. Int. J. Syst. Sci. **45**(1), 82–93 (2014)

50. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Parallel Problem Solving from Nature - PPSN-V, pp. 292–301. Springer, Heidelberg (1998)
51. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)
52. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: on the design of Pareto-compliant indicators via weighted integration. In: Evolutionary Multi-Criterion Optimization, pp. 862–876. Springer, Berlin (2007)