PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON GLOBAL OPTIMIZATION

Edited by

I. García, L.G. Casado
University of Almería

E.M.T. Hendrix
University of Wageningen

B. Tóth
University of Murcia

Preface

Global Optimization, the field including theory, methods and applications of optimization techniques aimed at detecting a global optimum for difficult mathematical programming problems in which many local optima might exist, is a rich area of research. The subject generates many papers published in qualified scientific journals and books; a journal and a series of monographs explicitly dedicated to the field exist now for more than a decade. Research done by PhD students, young researchers and seniors is scattered over the world and we see GO applied in many scientific fields. The workshop aims at bringing together researchers from various fields that are dealing with this topic.

The organising committee initiated the idea of the workshop at the MPS meeting in Kopenhagen. Inspired by the style of earlier workshops on GO in Sopron (1990), Szeged (1995) and Firenze (1999) the idea was to attract young researchers to be able to interact with experienced researchers in Global Optimization. The workshop is organised in single stream sessions, in order to give all participants the opportunity to enjoy each of the presentations, implying a limited number of participants. The number of accepted contributions was close to the maximum, such that finally the program of the workshop is filled completely.

The workshop might not have been possible without the supporting help of many people and organisations. Among these, we would like to express our sincere thanks to the contribution of:

- Ministerio de ciencia y tecnología
- Consejería de innovación, ciencia y empresa. Junta de Andalucía
- Universidad de Almería
- Diputación de Almería
- Ayuntamiento de Nijar

Organising committee:

- Emilio Carrizosa, Sevilla
- Tibor Csendes, Szeged
- Inmaculada García, Almería
- Eligius Hendrix, Wageningen
- Panos Pardalos, Florida

Local organisers:

- Miguel Cobo
- Leocadio Casado
- José Antonio Martínez
- Pilar Ortigosa
- Boglárka Tóth
- Consolación Gil
- Raúl Baños
- Juana Redondo

Contents

Plenary Talks

Extended Abstracts

PLENARY TALKS

Infinity computer and calculus

Yaroslav D. Sergeyev

Full Professor, Dipartimento di Elettronica
Informatica e Sistemistica
Universitá della Calabria
Via P.Bucci, Cubo 41C
87036 Rende (CS), Italy
Full Professor, Lobatchevsky State University,
Nizhni Novgorod, Russia
yaro@si.deis.unical.it

All the existing computers are able to execute arithmetical operations only with finite numbers. Operations with infinite and infinitesimal quantities could not be realized. This tutorial introduces a new positional system with infinite radix allowing us to write down finite, infinite, and infinitesimal numbers as particular cases of a unique framework. The new numeral system gives possibility to introduce a new type of computer able to operate not only with finite numbers but also with infinite and infinitesimal ones. The new approach both gives possibilities to execute calculations of a new type and simplifies fields of mathematics where usage of infinity and/or infinitesimals is necessary (for example, divergent series, limits, derivatives, integrals, measure theory, probability theory, etc.).

Particularly, the new approach and the infinity computer are able to do the following:

- to substitute symbols +infinity and -infinity by spaces of positive and negative infinite numbers, to represent them in the computer memory and to execute arithmetical operations with them as with normal finite numbers;
- to substitute qualitative description of the type "a number tends to zero" by precise infinitesimal numbers, to represent them in the computer memory and to execute mathematical operations with them as with normal finite numbers;
- to introduce a new definition of continuity that is closer to the real world than the traditional one;
- to calculate limits as arithmetical expressions;
- to calculate indeterminate forms in limits;
- to calculate sums of divergent series;
- to calculate improper integrals of various types;
- to calculate number of elements of infinite sets (and not only to distinguish numerable sets from continuum as it happens in the traditional approach);
- to evaluate functions and their derivatives at infinitesimal, finite, and infinite points (infinite and infinitesimal values of the functions and their derivatives can be also calculated);
- to study divergent processes at infinity;

- to work with sets having measure zero as with normal sets;
- to calculate lengths, areas, and volumes of fractals at infinity;
- to define and calculate volumes of objects having parts of different dimensions in a unique framework;
- to elaborate new mathematical models working simultaneously at micro and macro levels in a unique framework.

References

[1] Ya.D. Sergeyev, Arithmetic of infinity, Edizioni Orizzonti Meridionali, CS, 2003.

[2] Ya.D. Sergeyev, Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them, Patent filed in 2004.

[3] Ya.D. Sergeyev, Infinity computer and calculus, Opening plenary lecture at the congress SInfinity in Mathematics, Physics, and Philosophy T, Pisa, Italy, 2004.

[4] Ya.D. Sergeyev, A new computational paradigm, Invited plenary lecture at the VI-th International Congress on Mathematical Modelling, Nizhni Novgorod, Russia, 2004.

[5] Ya.D. Sergeyev, Infinity computer and calculus, Invited plenary lecture, International Conference SNumerical Analysis: the State of the Art T, 2005.

[6] Ya.D. Sergeyev, Infinity computer and calculus, Invited plenary lecture, International Conference on Difference Equations, Special Functions and Applications, 2005.

Recent Advances and Trends in Deterministic Global Optimization

Panos M. Pardalos

Distinguished Professor, Co-Director
Center for Applied Optimization
Industrial and Systems Engineering Department
303 Weil Hall, University of Florida
PO Box 116595
Gainesville, FL 32611-6595
pardalos@ufl.edu
http://www.ise.ufl.edu/pardalos

Global optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have 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 global optimization is the constantly increasing interdisciplinary nature of the field. This talk will cover the following topics: Nonconvexity and discreteness in optimization, DC (difference of convex functions) and monotonic optimization, Multiquadratic programming and applications, Multi-variable partition approaches and Hierarchical (multi-level) optimization, Optimization with massive datasets, Supply Chain, E-commerce, and E-manufacturing.

EXTENDED ABSTRACTS

A Trust-Region Algorithm for Global Optimization

Bernardetta Addis[1] and Sven Leyffer[2]

[1] *Università di Firenze, Firenze, Italy* b.addis@ing.unifi.it

[2] *Argonne National Laboratory, Argonne, IL, USA* leyffer@mcs.anl.gov

Abstract We consider the global minimization of a box-constrained function with a so-called funnel structure. We develop a two-phase procedure that uses sampling, local optimization, and Gaussian smoothing to construct a smooth model of the underlying funnel. The procedure is embedded in a trust-region framework that avoids the pitfalls of a fixed sampling radius. We present a numerical comparison to popular methods and show that the new algorithm is robust and uses fewer local minimizations steps.

Keywords: Global optimization, smoothing, trust region.

1. Introduction

We consider the global optimization problem

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in S \subset \mathbb{R}^n, \end{cases} \tag{1}$$

where f is sufficiently smooth and $S \subset \mathbb{R}^n$ is a compact set with simple structure, such as a bounded box.

Problems of type (1) arise in diverse fields, in particular, well-known conformational problems such as protein folding and atomic/cluster problems. In these applications we are interested in finding the lowest free energy conformation in three-dimensional space. A box can be defined that eventually will contain all "interesting" molecular conformations.

If the problem allows the use of a sufficiently efficient local optimization algorithm, a two-phase procedure is a good candidate for global optimization [7]. Such a procedure involves sampling coupled with local searches started from the sampled points. We define the local minimization operator as

$$L(x) := \begin{cases} \underset{y}{\text{minimize}} & f(y) \quad \text{starting from} \quad x \\ \text{subject to} & y \in S. \end{cases} \tag{2}$$

We note that this operator is implicitly defined and depends on the local minimizer used. In general, $L(x)$ is a piecewise constant function whose pieces correspond to the basins of attraction of the local minima of $f(x)$.

Clearly, the global optimization problem (1) has the same optimal objective value as the following problem:

$$\begin{cases} \underset{x}{\text{minimize}} & L(x) \\ \text{subject to} & x \in S. \end{cases} \tag{3}$$

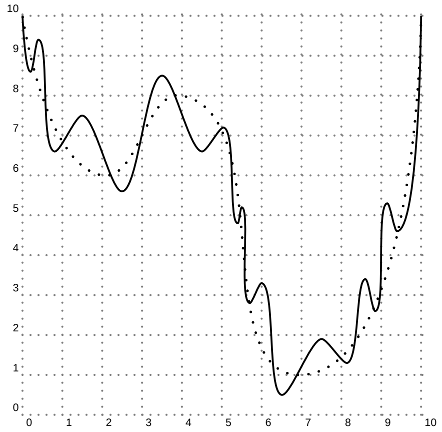


Figure 1. Example of a funnel function

We note that the piecewise constant nature of $L(x)$ implies that the minima of (1) and (3) need not agree. In fact, any global minimum of (1) is also a global minimum of (3), but not vice versa. Because $L(x)$ is implicitly defined, however, we can simply record

$$x_{min} := \mathcal{LS}(x) := \begin{cases} \underset{y}{\text{argmin}} & f(y) \quad \text{starting from} \quad x \\ \text{subject to} & y \in S. \end{cases} \tag{4}$$

It follows that x_{min} is also a local minimum of $f(x)$, and we can recover a global minimum of $f(x)$ by solving (3) in this way.

Multistart is an elementary example of a two-phase method aimed at minimizing $L(x)$; in practice, it reduces to a purely random (uniform) sampling applied to $L(x)$. It is in principle possible to apply any known global optimization method to solve the transformed problem (3), but many difficulties arise. First, function evaluation becomes much more expensive: we have to perform a local search on the original problem in order to observe the function $L(x)$ at a single point. Second, the analytical form of $L(x)$ is not available, and it is a discontinuous, piecewise constant function.

In many applications, such as molecular conformation problems [3], it is widely believed that the local optimum points are not randomly displaced but that the objective function $f(x)$ displays a so-called funnel structure. A univariate example of such a function is given in Figure 1, where the function to be minimized is represented by the solid line and the underlying funnel structure is given by the dotted line. In general, we say $f(x)$ has funnel structure if it is a perturbation of an underlying function with a low number of local minima. Motivated by examples of this kind, some authors [5,6,8] have proposed filtering approaches: if one can filter the high frequencies that perturb the funnel structure, then one can recover the underlying funnel structure and use a standard global optimization method on the filtered function (which is much easier to globally optimize) in order to reach the global optimum.

In contrast, we believe that it is better to filter the piecewise linear function $L(x)$ because it is less oscillatory than $f(x)$; Figure 2 shows $L(x)$ for the simple funnel function previously presented. This follows the approach of [2], and much of the analysis in [2] also applies here.

In this paper we make two important contributions to global optimization. First, we remove the need for the arbitrary parameters in [2] by interpreting these parameters as aradius. We embed the algorithm from [2], called ALSO, in aframework and show that our new algorithm is more robust than other methods. Second, we introduce the concept of global quality. This concept is motivated by the fact that theframework is essentially a local optimization scheme and therefore requires modifications to be effective as a global method.

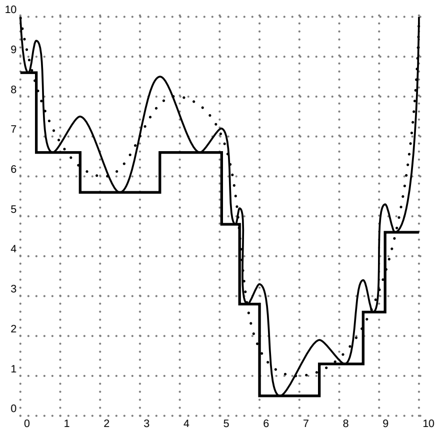


Figure 2. Example of the effect of local minimization

2. Summary

2.1 Gaussian Smoothing

We introduce a smoothing scheme to solve (3). The material of this section largely follows [2], although we give a different emphasis. We apply smoothing to $L(x)$ for two reasons. First, $L(x)$ is a piecewise constant function for which descent directions are difficult to define (first-order derivatives, when defined, are always zero). Second, we expect the smoothing to provide a more global view of the function. Given a real-valued function $L : \mathbb{R}^n \rightarrow \mathbb{R}$ and a smoothing kernel $g : \mathbb{R} \rightarrow \mathbb{R}$, which is a continuous, bounded, nonnegative, symmetric function whose integral is one, we define the g–transform of $L(x)$ as

$$\langle L \rangle_g(x) = \int_{\mathbb{R}^n} L(y) g(\|y - x\|) \, \mathrm{d}y. \tag{5}$$

The value $\langle L \rangle_g(x)$ is an average of the values of $L(x)$ in all the domain; in particular, the closer the points are to x, the higher is the contribution to the resulting value. Another important property is that $\langle L \rangle_g(x)$ is a differentiable function. Hence we can use standard smooth optimization methods to minimize it.

The most widely used kernel in the literature is the Gaussian kernel

$$g(z) \propto \exp\left(-z^2/(2\sigma^2)\right),$$

where we use the symbol $\propto$ to avoid writing a multiplicative constant that plays no role in the methods we present here. Clearly, one cannot explicitly apply the smoothing operator as in (5) because this approach requires the approximation of an n-dimensional integral. Instead, we restrict our attention to a ball of radius Δ around the current point x ($B(x, \Delta)$) and we construct a discretized version of the smoothing of $L(x)$,

$$\hat{L}_g^B(x) = \sum_{i=1}^{K} L(y_i) \frac{g(\|y_i - x\|)}{\sum_{i=1}^{K} g(\|y_i - x\|)}, \tag{6}$$

where y_i, $i = 1...K$, are samples in $B(x, \Delta)$. This function has interesting properties: it is a continuous function and a convex sum of the values of the samples. In particular, the weight associated with each sample is larger if we are closer to the sample point. In other words, the more confident we are in the sample value, the greater is the weight associated with it. In [2], the model (6) is used to choose the new candidate point x_+, starting from a point (say,

x_k). The model in the subset of given radius Δ, $B(x_k,\Delta)$, is solved by using a constrained optimization procedure. The use of a fixed value for Δ is restrictive for the length of the steps we are allowed to take. In addition, it is not clear a priori what value Δ should take. We do know that a small radius results in small steps and, hence, in slow convergence, but large values of Δ can result in poor agreement between the model and the function and, hence, in useless candidate points. Therefore, we propose to embed ALSO within a trust-region framework that adjusts the radius Δ automatically.

2.2 Trust-Region Framework

To extend the trust-region framework to global optimization of $L(x)$, we need to modify the smooth local trust-region algorithm. Specifically,we have to account for the fact that $L(x)$ is nonsmooth and that $\hat{L}_g^B(x)$ does not agree with $L(x)$ to first order. More important, we wish to avoid getting trapped in local minima. Clearly, it is not sufficient to simply reduce the trust-region radius if we reject the step. At each iteration we construct the model $\hat{L}_g^B(x)$ around our current iterate x_k. Specifically, we choose K samples (uniform) inside the current$B(x_k,\Delta_k)$ and perform a local minimization of $f(x)$ from each sample. If we find a new best point during this phase, we simply move to the new point and construct a new model. Otherwise, we apply a local minimization to the model inside the trust region and obtain

$$x_+ = \operatorname*{argmin}_{x\in B(x_k,\Delta_k)} \hat{L}_g^{B_k(x)}.$$

To decide whether to accept a step, we compute the ratio of the actual to the predicted reduction, namely,

$$\rho = \frac{L(x_k) - L(x_+)}{\hat{L}_g^{B_k}(x_k) - \hat{L}_g^{B_k}(x_+)},$$

noting that the predicted reduction $\hat{L}_g^{B_k}(x_k) - \hat{L}_g^{B_k}(x_+)$ is always nonnegative. We accept the new point x_+ if we observe sufficient decrease, that is $\rho \geq \eta_1 > 0$. If the step is very successful, $\rho \geq \eta_2 > \eta_1$, and theis active, $\|x_k - x_+\| \geq \Delta$, then we increase the trust region radius for the next iteration. As long as $\rho \geq \eta_1$, we refer to the iteration as successful, otherwise ($\rho < \eta_1$) the iteration is referred to as unsuccessful. Unsuccessfuliterations require special attention in the global optimization setting. In smooth local optimization, reducing Δ is guaranteed to improve the agreement between the model and the objective function. The same is not true in the global optimization context. Hence, we introduce a measure for the global quality of our model $\hat{L}_g^{B_k(x)}$,

$$q(\hat{L}_g^{B_k(x)}) = \frac{\max_{i\in M} \mid \{y_j : L(y_j) = L(y_i)\} \mid}{M}, \tag{7}$$

where M is the set of collected samples, that is, the largest number of samples with the same objective value, divided by the total number of samples. Clearly, $0 \leq q(\hat{L}_g^{B_k(x)}) \leq 1$, and a value close to 1 means that a large number of samples have the same function value and stem from the same "flat region" of $L(x)$. A smaller value of $q(\hat{L}_g^{B_k(x)})$ implies that the samples represent the global nature of the function $L(x)$ better.

In our algorithm, we compute $q(\hat{L}_g^{B_k(x)})$ at every unsuccessful iteration. If it is larger than a fixed value $\bar{q}$, we remove all but one sample from the largest set, increase theradius, and obtain new uniform samples in $B(x_k,\Delta_{k+1})\backslash B(x_k,\Delta_k)$. The motivation for this step is twofold: it improves the global nature of the model $\hat{L}_g^{B_k(x)}$, and it increases σ, thus smoothing the model.

The increase of σ arises because we have adopted the following formula for calculating the smoothing parameter, depending on theradius Δ and the number of samples K:

$$\sigma = \frac{\Delta}{K^{1/n}}, \tag{8}$$

where n is the dimension of the problem.

2.3 Conclusions

We presented a two-phase procedure for the global optimization of funnel functions. The approach builds on ALSO [2] and combines sampling with local searches. ALSO constructs a local smooth model from the samples by applying Gaussian smoothing. We demonstrated how to embed ALSO within a trust-region framework that adaptively updates the sample radius.

To extend the trust-region framework to global optimization, we introduced the concept of global quality, which triggers a model improvement step. Global quality measures the largest number of samples that have the same objective value and stem from the same basin of attraction. If global quality is large, then a model improvement step removes all but one sample from the largest set and generates a new set of uniform samples.

We compared our algorithm to ALSO and variants of monotone basin hopping (MBH). The new algorithm is more robust than ALSO and MBH on a range of test problems, and faster in terms of the number of local searches it requires per successful run.

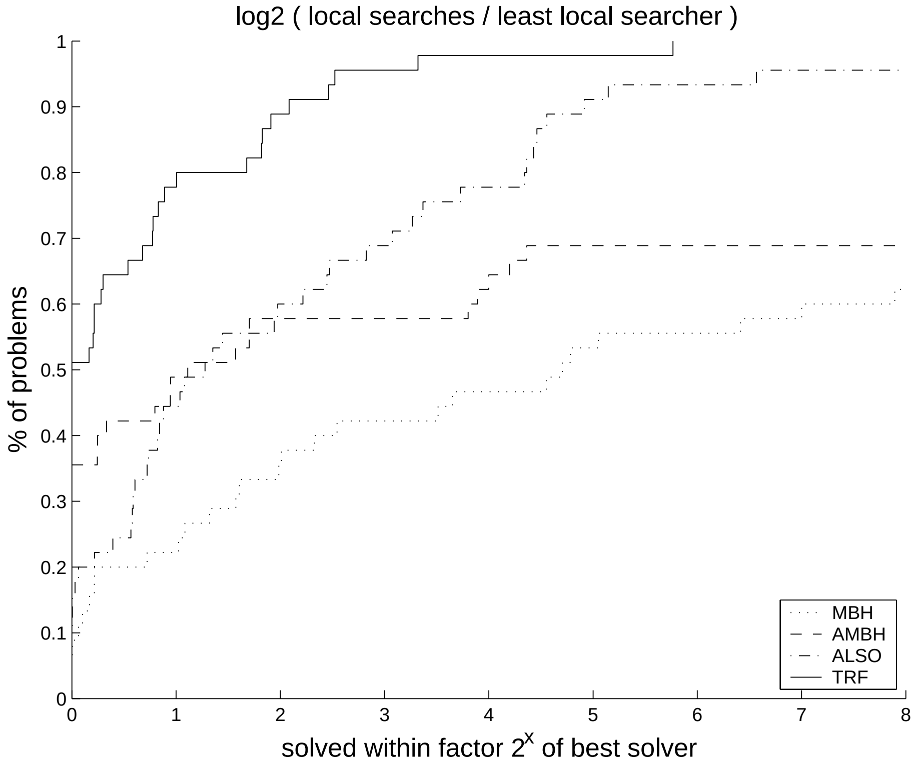


Figure 3. Performance profile of the average number of local searches

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38. The first author was also supported by the Italian national research program "FIRB/RBNE01WBBBB Large Scale Nonlinear Optimization".

References

[1] Addis B., and Leyffer S. (2004) A Trust-Region Algorithm for Global Optimization, *Preprint ANL/MCS-P1190-0804*, MCS Division, Argonne National Laboratory, Argonne, IL.

[2] Addis, B., Locatelli, M., and Schoen, F. (2003). *Local Optima Smoothing for Global Optimization*. To appear in Optimization Methods and Software.

[3] Doye, J. P. K. (2002). Physical perspectives on the global optimization of atomic clusters. In J. D. Pinter, editor, *Selected Case Studies in Global Optimization*, in press. Kluwer Academic Publishers, Dordrecht.

[4] Leary, R. H. (2000). Global optimization on funneling landscapes. *J. Global Optim.*, 18(4):367–383.

[5] Moré, J. J., and Wu, Z. (1996). Smoothing techniques for macromolecular global optimization. In G. D. Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 297–312. Plenum Press, New York.

[6] Moré, J. J., and Wu., Z. (1997). Global continuation for distance geometry problems. *SIAM J. Optim.*, 7:814–836.

[7] Schoen, F. (2002). Two-phase methods for global optimization. In P. Pardalos and E. H. Romeijn, editors, *Handbook of Global Optimization Volume 2*, pages 151–178. Kluwer Academic Publishers, Dordrecht.

[8] Shao, C. S., Byrd, R. H., Eskow, E., and Schnabel, R. B. (1997). Global optimization for molecular clusters using a new smoothing approach. In L. T. Biegler, T. F. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large Scale Optimization with Applications: Part III: Molecular Structure and Optimization*, pages 163–199. Springer, New York.

Global Optimization for Problems with a Huge Number of Local Optima

Bernardetta Addis[1], Marco Locatelli[2], and Fabio Schoen[1]

[1] *Dip. Sistemi e Informatica, Univ. di Firenze, via di Santa Marta, 3, 50139 Firenze (Italy)* (b.addis,schoen)@ing.unifi.it

[2] *Dip. Informatica, Univ. di Torino, Torino (Italy)* locatell@di.unito.it

Keywords: Global Optimization, Basin Hopping, Monte Carlo with minimization, Lennard Jones clusters, Morse clusters, protein-protein docking, circle packing

In this paper we will present a unifying framework for solving many difficult, large scale, global optimization problems with a huge number of local optima. Of course it is hopeless to pretend to develop a single general purpose algorithm for all kind of large scale global optimization problems. However, as we will show, if the objective function possesses some special structure, than it is possible to build optimization algorithms which are extremely efficient even in the presence of a number of local optima which increases exponentially with the problem dimension.

In computational chemistry and biology one such common structure is known as "funnel structure" and is commonly related to the fact that many energy functions which model complex many-body interactions, although being highly multimodal, can be optimized quite efficiently using methods of the Basin-hopping (BH) (or Montecarlo with Minimization) type. In this paper we review some of our recent results in applying BH-like methods to several important problems like:

- global optimization of atomic clusters interacting via the Lennard-Jones potential
- global optimization of short-range Morse clusters
- Protein-protein rigid docking
- Two-dimensional circle packing

While the first three problems, although quite different from each other, all originate from the field of computational chemistry and biology, the last one is a totally different one and is concerned with the optimal placing of N identical and non-overlapping circles of maximum radius in the unit square. Not only the origin of this well known problem are different from that of cluster optimization and protein docking, but also the structure of the problem is radically different. In fact, while in the first 3 problems, we are dealing with unconstrained global optimization problems, in the last case the problem is constrained, with a non convex feasible region. It is somewhat surprising that simple algorithmic ideas used in the context of molecular optimization can be quite easily ported to the constrained case.

In the paper we will review some of the main ingredients of BH-like methods for the problems discussed and, in particular, we will consider the following characteristics:

- the use of Monotonic Basin-Hopping as a general purpose algorithm for descending towards a funnel's bottom (provided a suitable definition of a neighborhood structure is available)

- the use of two-phase local searches to enlarge the basin of attraction of good local optima
- the effect of coordination obtained through the embedding of BH-like methods in a population-based framework in which different local optima are maintained in such a way as to guarantee a sufficiently high degree of dissimilarity between them.

Careful definition of the above three elements is clearly problem-dependent, although some general guidelines can be given for specific *classes* of problems. We will present some of the results recently obtained and the details on the implementation of the above steps, for all of the problems introduced. In particular, by means of a suitable definition of the neighborhood structure of BH and a two-phase approach in which penalties inspired by the geometrical structure of good solutions are introduced in the first phase, we could implement a single unbiased global optimization method which is capable of confirming all of the putative global optima for Lennard-Jones and Morse clusters deposited at the Cambridge Cluster Database [3, 5]. With a population-based approach we could further improve the efficiency of the method and were able to discover several new putative optima for Morse cluster at very short range [4].

Again, through a clever use of two-phase local optimization and a quite standard BH approach we obtained encouraging results for large protein-protein docking problems [2]: here, however, given the extremely high computational cost associated to each energy evaluation, only limited computational experience has been collected.

Finally we will report on the recent extension of BH-like methods to Circle Packing. Here we have been very quicly able to code a quite simple, although not elementary, version of BH (i.e., without two-phase local searches and with no population) which turned out to be extremely competitive with much more sophisticated methods. In very short time we have been able to improve more than 20 putative optima. In particular we obtained improved configurations for a number of circles N equal to $53, 59, 66, 68, 73, 77, 78, 85, 86, 88$ and many others for larger N values [1]. Work is in progress towards including a two-phase and/or a population based approach also in this context. However it can be quite safely assumed that the use of BH even in the simplest configuration enables to quickly discover excellent quality local solutions to difficult global optimization algorithms.

References

[1] B. Addis, M. Locatelli and F. Schoen, *Packing Circles in a Square: New Putative Optima Obtained Via Global Optimization*, on Optimization On Line
`http://www.optimization-online.org/DB_HTML/2005/03/1088.html`

[2] B. Addis and F. Schoen, *A randomized global optimization method for protein-protein docking*, available on Optimization On Line
`http://www.optimization-online.org/DB_HTML/2003/04/638.html`, (2003).

[3] J. P. K. Doye, R. H.Leary, M. Locatelli and F. Schoen, *The global optimization of Morse clusters by potential transformations, INFORMS J. Computing*, 16, 371-379 (2004)

[4] A. Grosso, M. Locatelli and F. Schoen, *A population based approach for hard global optimization problems based on dissimilarity measures*, submitted, available on Optimization On Line
`http://www.optimization-online.org/DB_HTML/2005/02/1056.html` (2005).

[5] M.Locatelli and F.Schoen, *Efficient algorithms for large scale global optimization: Lennard-Jones clusters, Computational Optimization and Applications*, 26, 173-190 (2003).

Exact Algorithms for Robust k-Path Routing Problems

April K. Andreas[1] and J. Cole Smith[2]

[1] *Department of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ,* april@email.arizona.edu

[2] *Department of Industrial and Systems Engineering, The University of Florida, Gainesville, FL,* cole@ise.ufl.edu

Abstract Many routing problems in the literature examine stochastic problems in which costs and/or travel times are the uncertain elements. However, an implicit assumption is often made that all arcs and nodes operate with perfect reliability. Suppose instead that there exist independent arc failure probabilities. We may wish to ensure that a path exists from some origin to some destination that survives with a certain threshold probability. One manner of doing this is to construct k origin-destination paths at a minimum cost, while ensuring that at least one path remains operational with a sufficiently large probability. However, the reliability constraint induces a nonconvex feasible region (even when the integer variable restrictions are relaxed). We begin by examining the Robust Two-Path Problem, which seeks to establish two paths between a source and destination node wherein at least one path must remain fully operable with some threshold probability. We consider the case where both paths must be arc-disjoint and the case where arcs can be shared between the paths. This paper begins by exploring various strategies for solving the resulting nonlinear integer program, including pruning, coefficient tightening, lifting, and branch-and-bound partitioning schemes. We then examine the Robust k-Disjoint-Path problem, and note that the previous scheme for two-path problems does not generalize well to the k-path situation. Instead, we propose an alternative model for the problem in which the variables correspond to paths and the nonconvex reliability constraints are replaced by an exponential set of linear constraints. Accordingly, we propose a branch-and-cut-and-price algorithm for handling the new problem.

Keywords: Integer programming, nonlinear programming, branch-and-bound, branch-and-price-and-cut.

1. Introduction

Traditional routing problems, such as shortest path and travelling salesman problems, assume perfect operability of all arcs and nodes. However, real-world applications often call for a more diverse routing strategy. For example, consider a cell phone company that has determined that its customers will tolerate having up to 5% of their calls dropped. The cell phone company can respond by either sending each call over one highly reliable path, or by sending two signals per call along less-reliable paths and requiring that at least one signal reaches the destination with a probability of 95%. It might be cheaper for the company to buy two relatively unreliable paths than to buy one highly-reliable path. Such redundant routing takes place in Synchronous Optical Networks, for instance, by routing along rings such that a single line break will not affect client service. Another application arises in achieving military missions, such as destroying a particular target that can be accessed via any number of paths, each path with its own risks. A commander may choose to deploy two groups along different paths such that at least one of the groups will arrive at and destroy the target within some acceptable level of reliability.

Hence, when these independent arc failure probabilities exist, a secondary constraint would retain some measure of expected functionality, introducing nonlinear, nonconvex constraints. In this abstract, we introduce mathematical programming techniques to impose these requirements on path selection problems on a directed graph $G(N, A)$, where N is the set of nodes $\{1, \cdots, n\}$ and A is the set of arcs.

Single shortest path reliability problems have received much attention in the literature, both in their methodological development and in their applications. Bard and Miller [2] address a research and development project selection problem, in which spending additional money on projects could increase their probability of success. Zabarankin, Uryasev, and Pardalos [12] consider a related problem in optimizing path risk in the context of aircraft flight trajectory through a threat environment. Elimam and Kohler [4] describe some unique applications of the resource-constrained shortest-path problem, such as the determination of optimal wastewater treatment processes and thermal resistance of building structures.

Too many papers exist to cite on multiple-path routing; we briefly mention two of the most relevant ones here. Suurballe [9] presents a polynomial-time labeling algorithm to find k node-disjoint paths between two given nodes. Fortune, Hopcroft, and Wyllie [5] provide an important characterization of NP-complete vertex-independent routing problems on directed graphs via the graph homeomorphism problem.

This problem can also be viewed in the context of a network survivability problem. In networks for which survivability is of concern, an assumption is made that the probability of two arcs failing is sufficiently small. Hence, they create active and backup paths that do not share any arcs. (If vertices are subject to failure is well, the paths must be node-disjoint as well, except at the origins and destinations.) Therefore, the objective is to minimize the total bandwidth reserved to meet a given demand, allowing the sharing of the backup links among disjoint connections. Li et al. [6] approach the problem in terms of networking for multi-protocol label switched networks, developing extensions to the signaling protocol to distribute additional link usage information. The authors also explain how to extend the algorithm to account for single node failures (multiple link failures) and fiber span failures. Several other groups explore this problem, including Liu and Tipper [7], Xu, Qiao, and Xiong [10], and Yee and Lin [11].

2. Models and Algorithms

The authors present their research on two-path routing problems in [1]. We can begin this analysis by examining a shortest path problem with the side constraint that the probability of successfully traversing the path is at least as large as some threshold probability. The natural formulation of this problem is a mixed-integer nonlinear program, though with some care the problem can be transformed to a mixed-integer linear program. In any case, these side constraints prevent a polynomial-time application of Dijkstra's algorithm, and make this problem ordinarily NP-hard.

An even more interesting and applicable version of this problem seeks to establish two arc-independent paths between two nodes subject to some minimum reliability measure. Many practical scenarios simply require that at least one path survives between a pair of nodes, a problem which we call the Robust Two-Path Problem with Disjoint arcs (RTP-D). In this case, we stipulate that the probability that at least one path survives is sufficiently large. This probability is given by one minus the probability that both paths fail, which is an inherently nonlinear constraint. Even worse, such a constraint induces a nonconvex feasible region. The problems are aggravated further if we allow high-reliability, low-cost arcs to be shared between the paths (RTP-S). Note that in this case, the failure of a shared arc causes both paths to fail. We show in [1] that both RTP-D and RTP-S are strongly NP-hard.

Let the network be denoted by $G(N, A)$, and let the cost and reliability of each arc $(i, j) \in A$ be given by c_{ij} and p_{ij}, respectively. Define $FS(i)$ and $RS(i)$ to be the forward and reverse stars of node $i \in N$, respectively (i.e., the set of arcs exiting and entering node i). Finally, let τ be the minimum permissible reliability that at least one path survives.

We can define x_{ij}^q for all $(i,j) \in A$ and $q = 1,2$ to be a binary variable equal to one if path number q utilizes arc (i,j), and zero otherwise. Also, let variables s_i^q represent the probability that path q successfully reaches node i from node 1, given that path $q = 1,2$ visits node i. The RTP-D can be thus modelled as a union of minimum cost flow problem constraints, linear path-probability calculation constraints, and a single nonlinear, nonconvex constraint as follows.

$$\text{minimize} \sum_{(i,j)\in A} c_{ij}\left(x_{ij}^1 + x_{ij}^2\right) \tag{1}$$

$$\text{subject to} \sum_{j\in FS(1)} x_{1j}^q = 1 \quad \forall q = 1,2 \tag{2}$$

$$\sum_{j\in FS(i)} x_{ij}^q = \sum_{h\in RS(i)} x_{hi}^q \quad \forall i \in \{2,\cdots,n-1\},\ \forall q = 1,2 \tag{3}$$

$$x_{ij}^1 + x_{ij}^2 \le 1 \quad \forall (i,j) \in A \tag{4}$$

$$s_j^q \le p_{ij}s_i^q + (1 - x_{ij}^q) \quad \forall q = 1,2,\ \forall (i,j) \in A \tag{5}$$

$$s_1^1 = s_1^2 = 1 \tag{6}$$

$$s_n^1 + s_n^2 - s_n^1 s_n^2 \ge \tau \tag{7}$$

$$s_n^1 \ge s_n^2 \tag{8}$$

$$x_{ij}^q \in \{0,1\} \quad \forall q = 1,2,\ \forall (i,j) \in A. \tag{9}$$

The first part of this formulation is straightforward and standard in the literature. Our strategy in (5) and (6) uses a set of constraints to compute the s-variables. Note that the path-probability calculation constraints are linear, but employ binary variables that give the problem its combinatorial nature. The nonlinear constraint (7) enforces the condition that at least one path remains survivable with sufficiently large probability. Constraint (8) removes some problem symmetry in the model to improve its solvability.

Our approach to handling the single nonlinear constraint constructs a convex hull relaxation of the feasible region by relaxing the integrality constraints, and by replacing (7) with

$$(1 - \sqrt{1-\tau})s_n^1 + (\sqrt{1-\tau} - 1 + \tau)s_n^2 \ge \tau(1 - \sqrt{1-\tau}). \tag{10}$$

We can then utilize a divide-and-conquer recursive approach to force the final solution to reside within the original nonconvex feasible region. Our approach is to reinstate the integrality constraints and solve the problem by relaxing only the constraint (7) and replacing it with (10). If the resulting solution is feasible to the original problem, an optimal solution has been identified. Else, we can tighten the relaxation by creating a disjunction in which at least one of two cutting planes is valid. With this in mind, we solve the relaxed problem in a branch-and-bound fashion in which each subproblem is an integer program over some particular interval. We provide an illustration of this technique in Figure 1.

We can enhance the model further by using a simple variation of Dijkstra's algorithm to establish an upper bound on the probability of successfully reaching a given node, as well as a lowest-permissible probability of reaching an intermediate node and still reaching node n with the required probability. Information about these probabilities can then be used to reduce graph $G(N,A)$ appropriately for each path. Directed acyclic graphs provide yet another opportunity for pruning, employing a variation on Dijkstra's algorithm where the least reliable path to (or from) each node is stored. We can also use this information to tighten our constraints over all feasible solutions by using coefficient tightening. Our computational experience supports the strategy of using pruning and coefficient tightening, but shows that using lifting on the constraints is not effective.

If the solution in a given interval is infeasible to the original problem, we must recursively solve the problem over the interval by splitting it up into smaller intervals. There are numer-

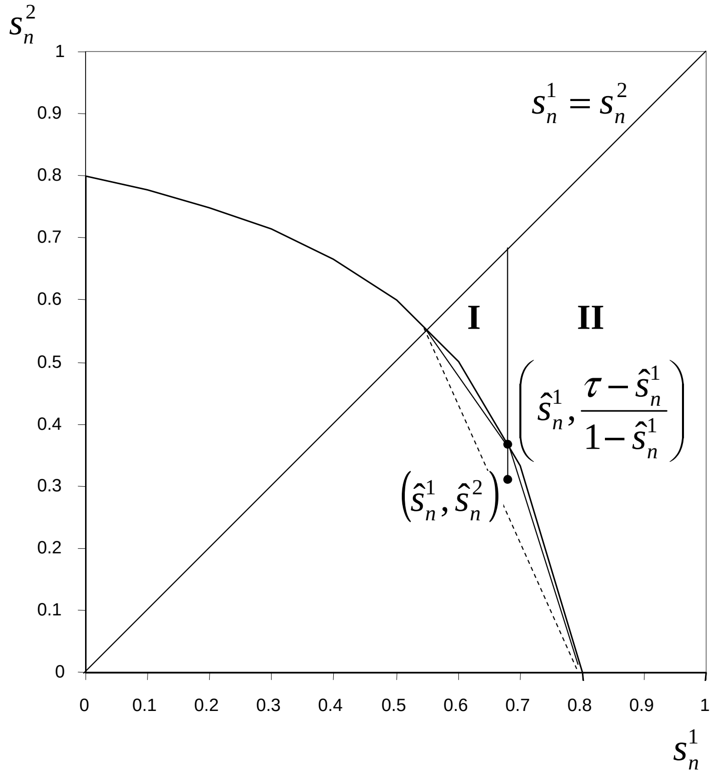


Figure 1. Illustration of First Iteration of Vertical Branch-and-Bound Algorithm

ous ways that the interval could be split to prevent the nonlinear-infeasible solution from recurring (e.g., Figure 1 shows a "vertical branching" rule). We examine the advantages and disadvantages of these methods and present computational results to support the use of one particular method; specifically, choosing a partitioning for which the distance from the nonlinear-infeasible solution to either partitioning plane is equal, thereby maximizing the minimum distance to either partitioned region. This comparatively effective method of branching does not rely on the axes to generate the cuts, and generalizes the Reformulation-Linearization Technique [8] (for continuous-variables optimization problems) on this particular problem.

Relaxing the edge-disjoint constraints will necessitate adjustments to be made to RTP-D. The formulation and algorithm adjustments to solve the shared-arcs variation of our problem (RTP-S) must then be modified. Our approach now requires the introduction of an additional decision variable and moves our linear chord relaxation into a three-dimensional space. (Details are omitted in this paper due to space limitations.) Model tightening via pruning and lifting, as well as preprocessing is also less straightforward for RTP-S, though it is still possible. A specialized partitioning procedure proves difficult, however, for this particular problem due to the fact that joint reliability constraint is now neither convex nor concave. We work around this problem by implementing the Reformulation-Linearization Technique [8] to assist in partitioning our space for the branch-and-bound procedure. We provide computational results to recommend the best pruning, lifting, and partitioning strategies for solving RTP-S.

A natural extension to the two-disjoint-path problem is the k-disjoint-path problem for $k > 2$. The previous approach would suffer due to the increasing difficulties of relaxing multiple nonlinear terms as k increases. An alternative approach is to consider a path-based formulation, in which variables x_p equal to one if path $p \in P$ is selected, and zero otherwise, where P is the entire set of origin-destination paths in the network. However, there are too many such paths to generate, and a column generation approach would have to deal with the nonlinear constraint governing reliability. Instead, we can replace the reliability constraint with a set of covers C^j, $|C^j| \geq k$, such that the reliabilities of paths in C^j are too small to be used in a feasible solution. Hence, we can restrict

$$\sum_{i \in C^j} x_i \leq k - 1 \quad \forall \text{ covers } C^j. \tag{11}$$

However, at this point, we have generated an exponential number of variables and constraints. Accordingly, we employ a branch-and-price-and-cut algorithm to solve the problem. Note that after generating some m covers and set of paths $\overline{P}$, and after relaxing integrality, we have the formulation

$$\text{Minimize} \sum_{p \in \overline{P}} C_p x_p \tag{12}$$

$$\text{subject to} \sum_{p \in \overline{P}} x_p = K \tag{13}$$

$$\sum_{p \in \overline{P}(i,j)} x_p \leq 1 \quad \forall (i,j) \in A \tag{14}$$

$$\sum_{p \in \overline{P}} a_{rp} x_p \leq k - 1 \quad \forall r = 1, ..., m \tag{15}$$

$$0 \leq x_p \leq 1 \quad \forall p \in \overline{P}, \tag{16}$$

where C_p is the total cost of a path (sum of its edge costs), $\overline{P}(i,j)$ is the set of paths generated thus far that use arc (i,j), and a_{rp} is a constant equal to 1 if p belongs to cover r, and zero otherwise. The column generation phase of the algorithm attempts to find variables that enter the basis of problem (12)-(16), but is complicated by the presence of constraints (15). Those constraints require us to determine whether or not a new path (column) is a member of an existing cover constraint. We can solve this problem in pseudopolynomial time, however, by using algorithms designed for resource-constrained shortest path problems. After this problem is solved, the row generation portion is straightforward. This process continues until no columns or rows can be generated.

Following the row/column generation phase, the branching phase must occur in a manner that forces the algorithm to converge. Simply branching on an x_p-variable is problematic, because a variable that is forced to equal to zero will reappear under a different index in the column generation subroutine that follows the branching phase. Hence, we prove that if the previous solution to the linear programming relaxation is fractional, there either exists an optimal solution in which at least one arc has a fractional flow, or an optimal integer solution can be found based on this fractional solution. In the former case, we can ascertain that fractional arc flow, and then branch by insisting that this arc either contains a total flow of either zero or one.

However, the latter case induces dual variables that could force the column generation subproblem to solve a shortest path problem with negative arc costs. While unlikely, this makes the column generation problem strongly NP-hard. Instead, we prescribe a slightly different multicommodity flow model based on the work of Barnhart, Hane, and Vance [3] for our problem. This new model avoids the NP-hard column generation routine, but at the expense of a much larger model with symmetry complications. We will examine the computational effectiveness of these methods in our future research.

3. Summary

We have examined the two-path routing problem with reliability considerations, which turns out to be NP-hard in the strong sense, as opposed to the ordinary NP-completeness of the single-path problem. Our algorithms for these two-path problems are based on relaxing a complicating nonconvex constraint and providing a (hopefully tight) series of disjunctive convex approximations to the problem. For the multiple path problem, we instead turn to a path-based formulation instead of the arc-based formulation for the two-path problem, and replace the reliability constraint with a set of linear cover constraints. We propose the use of a branch-and-price-and-cut framework to solve these problems. Our future research will involve providing the details of the branch-and-cut-and-price algorithm, and stating computational results comparing the efficacy of using our algorithm on model formulations.

Acknowledgments

The authors gratefully acknowledge the support of the Office of Naval Research under Grant Number N00014-03-1-0510 and the Air Force Office of Scientific Research under Grant Number F49620-03-1-0377.

References

[1] A.K. Andreas and J.C. Smith. Mathematical Programming Algorithms for Two-Path Routing Problems with Reliability Considerations. Working Paper, Tucson, AZ, 2005.

[2] J.F. Bard and J.L. Miller. Probabilistic Shortest Path Problems with Budgetary Constraints. *Computers and Operations Research*, 16(2):145–159, 1989.

[3] C. Barnhart, C.A. Hane, and P.H. Vance. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. *Operations Research*, 48(2):318–326, 2000.

[4] A.A. Elimam and D. Kohler. Case Study: Two Engineering Applications of a Constrained Shortest-Path Model. *European Journal of Operational Research*, 103:426–438, 1997.

[5] S. Fortune, J. Hopcroft, and J. Wyllie. The Directed Subgraph Homeomorphism Problem. *Theoretical Computer Science,* **10**:111–121, 1980.

[6] G. Li, D. Wang, C. Kalmanek, and R. Doverspike. Efficient Distributed Path Selection for Shared Restoration Connections. *IEEE INFOCOM 2002,* **1**:140–149, 2002.

[7] Y. Liu and D. Tipper. Successive Survivable Routing for Node Failures. *GLOBECOM 2001 - IEEE Global Telecommunications Conference* **1**:2093–2097, 2001.

[8] H.D. Sherali and C.H. Tuncbilek. A Global Optimization Algorithm for Polynomial Programming Problems Using a Reformulation-Linearization Technique. *Journal of Global Optimization,* 2:101–112, 1992.

[9] J.W. Suurballe. Disjoint Paths in a Network. *Networks,* **4**:125–145, 1974.

[10] D. Xu, C. Qiao, and Y. Xiong. An Ultra-fast Shared Path Protection Scheme-Distributed Partial Information Management, Part II. In *Proceedings of the 10th IEEE International Conference on Network Protocols,* pages 344–353, 2002.

[11] J.R. Yee and F.Y.S. Lin. A Routing Algorithm for Virtual Circuit Data Networks with Multiple Sessions Per O-D Pair. *Networks,* 22:185–208, 1992.

[12] M. Zabarankin, S. Uryasev, and P. Pardalos. Optimal Risk Path Algorithms. In *Cooperative Control and Optimization,* pages 273–298, Dordrecht; Boston: Kluwer Academic Publishers, 2001.

The Small Octagon with Longest Perimeter

Charles Audet[1], Pierre Hansen[2] and Frédéric Messine[3]

[1] *GERAD and Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal (Québec), H3C 3A7 Canada,* Charles.Audet@gerad.ca

[2] *GERAD and Département des Méthodes Quantitatives, HEC Montréal, 3000 Chemin de la côte Sainte Catherine, Montréal H3T 2A7, Canada, École des Hautes Études Commerciales, C.P. 6079, Succ. Centre-ville, Montréal (Québec), H3C 3A7 Canada,* Pierre.Hansen@gerad.ca

[3] *ENSEEIHT-IRIT, 2 rue C Camichel, 31071 Toulouse Cedex, France,* Frederic.Messine@n7.fr

Abstract The convex octagon with unit diameter and maximum perimeter is determined. This answers an open question dating from 1922. The proof uses geometric reasoning and an interval arithmetic based global optimization algorithm to solve a series of non-linear and non-convex programs involving trigonometric functions.

This article summarizes the complete proof published in [2].

1. Introduction and related problems

We answer in this work, a geometric problem opened since 1922 by Reinhardt, [14]. We call a *small $n-$gon* a polygon with n vertices (or n edges) and with a unit diameter (the longest distance between two extreme point is 1). In [14], Reinhardt studies the properties of maximal area and maximal perimeter of small polygons. He proves that the regular $n-$gons with equal sides and equal angles have the properties to be of both maximal area and maximal perimeter if n is odd. When n is even, Reinhardt proves that the square owns the property of maximal area and that there exists a regular polygon with only equal sides (based on a Reuleaux polygon) which has the property of maximal perimeter if $n = 2^s$ for $s \in \mathbb{N} \setminus \{0, 1\}$. Therefore, the open cases were the $4-$gon for the maximal perimeter and the hexagon for the problem of maximal area. In 1975, Graham proves that there exists an irregular small hexagon which has a maximal area about 4% superior to the regular one, [7]. For proving this, Graham bisected the global optimization problems into 10 small ones. Woodall proves in 1971, that the optimal solutions are in this 10 configurations which are called linear thrackleations, [17]. 9 of these cases were eliminated by using geometric arguments and the last case amounted to solve an univariate global optimization problem which gives the optimal solution for the problem of which unit diameter hexagon has the maximal area. The next open case: The largest small octagon was solved by Audet et al. [4], using the same way nevertheless this led to consider 31 thrackleation graphs corresponding to 31 sub-problems. Therefore, one proves in [4] that there exists an irregular small octagon with maximal area about 2.82% greater than the regular one. This problem was extremely difficult to solve because the most difficult case (case 31 which leads to the global solution about the 31 thrackleations) has 10 variables and needed 100 hours of computations for a specific global optimization algorithm dedicated to non-convex quadratic programs [1].

Now, considering the problem of the maximal perimeter. In 1997, Datta [6] proves that the small 4-gon based on a triangle of sides one with a supplementary vertex at the bisector of one angle and at distance one of the opposite vertex, has the property to be of maximal perimeter: $2 + 4\sin\frac{\pi}{12} \sim 3.03527618....$ In order to compare, the square has a perimeter about $2\sqrt{2} \sim 2.82842712...$

Then, the next open case studied here is the octagon, see [2].

Today, it seems to be unfeasible to solve directly this problem using only geometric arguments (open problem since 1922) and using only an exact global optimization tool (because the problem is too complicated). Our proof combines new geometric arguments and a specific interval global optimization algorithm due to Messine et al. [9–11] which can deal with trigonometric functions.

The following summarizes the way to find the small octagon with maximal perimeter; for the complete proof see [2].

2. Bisection into 31 sub-problems

As for the largest small octagon, [4] one uses here the decomposition in sub-problems using linear thrackleation graphs. A linear thrackleation graph is a graph such that there exists always a path joining two extreme points; an edge of this graph is obtained if the distance between two vertices is equal to 1 (this corresponds to a binding constraints). In [2], one proves that the solution is based on one of the 31 linear thrackleations given in [4]. One does not represent all these configurations but one uses the same numbering as in [4]. In Figure 1, the most important thrackleations (case 29 and case 31) are represented.

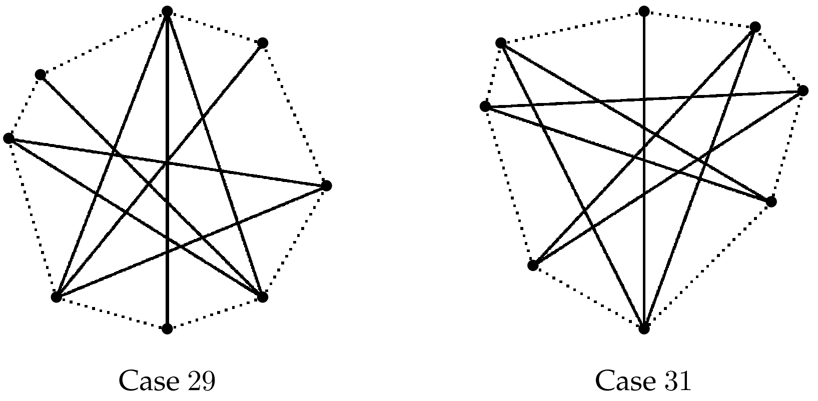


Figure 1. Thrackleations 29 and 31.

3. Bounds derived from Datta's results

In [6], Datta proves that all the optimal $n-$gons when $n \geq 5$ cannot have a side of length one. This lemma allows to eliminate 7 cases: thrackleations numbered 1, 2, 3, 4, 21, 22, 23 in [4].

Datta also proves in [6], that an upper bound for the perimeter of a small $n-$gon is

$$2n\sin\frac{\pi}{2n}.$$

This bounds are attained when $n \neq 2^s$ with $s \in \mathbb{N} \setminus \{0,1\}$.

Figure 2 represents a common and possible configuration inside a thrackleation. v_i, v_j, v_{j+1} and v_{j+2} are four vertices of the octagon and of the corresponding thrackleation, c_j and c_{j+1}

are the length of two sides of the octagon, and $[v_i, v_j], [v_i, v_{j+1}]$ and $[v_i, v_{j+2}]$ are edges of the thrackleation graph; i.e. $\|v_i - v_j\| = \|v_i - v_{j+1}\| = \|v_i - v_{j+2}\| = 1$.

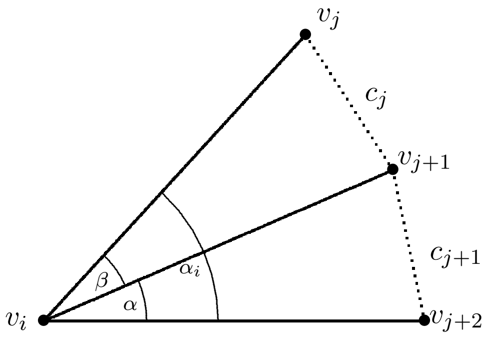


Figure 2. Equal angles.

Using the same demonstration, one proves in [2] the two following properties:

Proposition 1. *Consider a small octagon based on a thrackleation with maximal perimeter. If c_j is a type I edge, as represented in Figure 2 (v_i is an extreme point of two edges), and v_i its associated vertex, then $\angle v_j v_i v_{j+1} \in [0.317, 0.465]$. Moreover, if c_{j+1} is also a type I edge, consecutive to c_j and with the same associated vertex v_i then $\angle v_j v_i v_{j+2} \in [0.688, 0.881]$.*

The next result concerns thrackleations containing a pending diameter.

Proposition 2. *Consider a small octagon based on a thrackleation with maximal perimeter, with two consecutive type I edges $c_j = [v_j, v_{j+1}]$ and $c_{j+1} = [v_{j+1}, v_{j+2}]$ sharing the same associated vertex v_i. Then $\angle v_j v_i v_{j+1} = \angle v_{j+1} v_i v_{j+2} = \frac{\angle v_j v_i v_{j+2}}{2}$; i.e. $\alpha = \beta = \frac{\alpha_i}{2}$.*

These two properties allow to reduce the remaining studied problems.

4. Exact algorithm

In that stage, one needs the use of an exact global optimization algorithm to discard some cases and to determine the global solution.

The global optimization algorithm used here is a Fortran 90/95 implementation of a branch-and-bound method where bounds are computed with interval analysis techniques. It is called IBBA (for Interval Branch and Bound Algorithm). All computations were performed on a cluster of thirty bi-processors PC's ranging from 1GHz to 2.4GHz at the University of Pau.

Interval analysis was introduced by Moore [12] in order to control the propagation of numerical errors due to floating point computations. Thus, Moore proposes to enclose all real values by an interval where the bounds are the two closest floating point numbers. Then expanding the classical operations - addition, subtraction, multiplication and division- into intervals, defines interval arithmetic. A straightforward generalization allows computation of reliable bounds (excluding the problem of numerical errors) of a function over a hypercube (or box) defined by an interval vector. Moreover, classical tools of analysis such as Taylor expansions can be used together with interval arithmetic to compute more precise bounds [12]. Other bounding techniques due to Messine et al. [11] combines linear underestimations (or hyperplanes) obtained at all vertices of the box [11].

The principle of IBBA is to bisect the initial domain where the solution is sought for into smaller and smaller boxes, and then to eliminate the boxes where the global optimum cannot occur:

- by proving, using interval bounds, that no point in a box can produce a better solution than the current best one;

- by proving (with interval arithmetic), that at least one constraint cannot be satisfied by any point in such a box.

To accelerate convergence, constraint propagation techniques are used in some steps of IBBA, see [9,10] for details. The principle is to use, a priori, the implicit relations between the variables (induced by the constraints) to reduce the size of a box.

5. Optimal solution

The optimal solution appeared by solving case corresponding to thrackleation 10, see Figure 3, which is a relaxation of case 29: Add edge $[v_0, v_4]$ to case 10 then you obtain thrackleation 29 given in Figure 1. This just changes the constraint $\|v_0 - v_4\| \leq 1$ into the inequality constraint $\|v_0 - v_4\| = 1$.

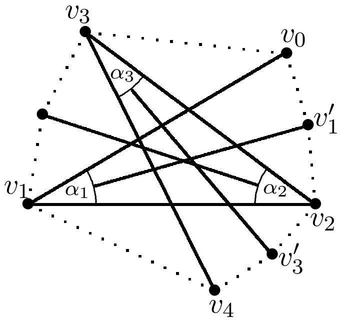


Figure 3. Thrackleation 10

The non-convex program is:

$$\begin{array}{ll} \max\limits_{\alpha} & 4\sin\frac{\alpha_1}{4} + 4\sin\frac{\alpha_2}{4} + 4\sin\frac{\alpha_3}{4} + ||v_1 - v_4|| + ||v_0 - v_3|| \\ \text{s.t.} & ||v_0 - v_4|| \leq 1 \\ & 0.688 \leq \alpha_i \leq 0.881 \qquad i = 1, 2, 3, \end{array}$$

where $v_0 = (\cos\alpha_1, \sin\alpha_1), v_1 = (0, 0), v_2 = (1, 0), v_3 = (1 - \cos\alpha_2, \sin\alpha_2)$ and $v_4 = (1 - \cos\alpha_2 + \cos(\alpha_2 + \alpha_3), \sin\alpha_2 - \sin(\alpha_2 + \alpha_3))$.

Solving this program by using Algorithm IBBA, one obtains in 3 hours the optimal solution which has a perimeter about $p^* = 3.121147$, with an error less than 10^{-6}. In this solution, the constraint $||v_0 - v_4|| \leq 1$ is binding and thus the optimal configuration corresponds to case 29. In [2], one found, using MAPLE^{TM}, an analytical solution for p^*.

Furthermore, by adding the following constraints derived from Proposition 2:

$$\frac{\partial\left(\|v_2 - v'_1\| + \|v'_1 - v_0\| + \|v_0 - v_3\|\right)}{\partial\alpha_1} = 0$$

and,

$$\frac{\partial\left(\|v_2 - v'_3\| + \|v'_3 - v_4\| + \|v_4 - v_1\|\right)}{\partial\alpha_3} = 0$$

where $v'_1 = \left(\cos(\frac{\alpha_1}{2}),\ \sin(\frac{\alpha_1}{2})\right)$ and $v'_3 = \left(x_3 + \cos(\alpha_2 + \frac{\alpha_3}{2}),\ y_3 - \sin(\alpha_2 + \frac{\alpha_3}{2})\right)$, IBBA showed in only 0.12 seconds that there are no feasible solution. Therefore, case 10 is eliminated.

6. Eliminating cases using IBBA

Now one uses IBBA in order to discard a lot of remaining tightened (by geometric arguments, see Section 3) cases corresponding to linear thrackleations.

In a first step, one considers configurations which has an isolated vertex. It is a particular configuration such that only one vertex is above (or below by symmetry) an edge, see Figure 4.

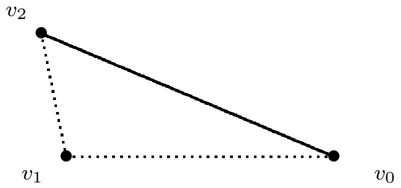


Figure 4. An octagon with an isolated vertex v_1

IBBA permit to eliminate all these configurations by adding the constraint that an upper bound of all the perimeters of all these configurations must be greater than $p^* = 3.121147$.

Therefore, 16 cases are eliminated together, corresponding to thrackleations numbered 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 30.

Considering the remaining cases: 16, 17, 19, 20 and 31, IBBA shows in less than 5 hours for each configurations, that all of them can have a perimeter superior to p^*.

It remains only case of thrackleation 18, which is a relaxation of 31 and also 29 (the optimal thrackleation). In that case, the optimal conditions derived from Proposition 2 are satisfied for 29, and then this problem can only be solved with a precision $\epsilon = 0.5 \times 10^{-4}$. IBBA shows in case 18, that it does not exist a better solution than $p^* + \epsilon$ for this configuration.

This completes the proof; see [2] for more details.

7. Conclusion

The small octagon with longest perimeter has been determined and the length of its perimeter is equal to 3.1211 (all decimals are exact), with an error guaranteed to be less than $\epsilon = 0.5 \times 10^{-4}$. As in the study of the small octagon with largest area [4], the diameter graph of an optimal octagon must be a connected linear thrakleation. This lead to 31 cases. The optimal one is case 29 for the longest perimeter while it was case 31 for the largest area. The diameter graph for the octagon with unit-length side and smallest diameter [3] was not connected. In all cases the optimal figure possesses an axis of symmetry. For comparison, Figure 5 illustrates three small octagons: The regular one, the one with maximal perimeter with equal sides [3], and the one studied in this paper.

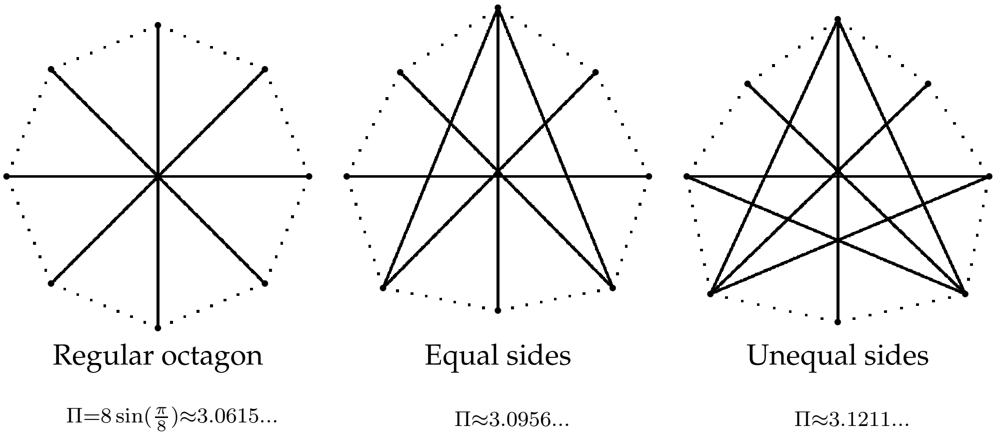
The above mentioned optimal octagons were obtained by combining geometric reasoning with global optimization algorithms [1, 10]. It appears difficult to answer any of these questions with one of those tools alone.

References

[1] C. Audet, P. Hansen, B. Jaumard, and G. Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Math. Program.*, 87(1, Ser. A):131–152, 2000.

[2] C. Audet, P. Hansen, F. Messine. The small octagon with longest perimeter. *Cahiers du GERAD*, http://www.gerad.ca/en/publications/cahiers.php, 2005.

[3] C. Audet, P. Hansen, F. Messine, and S. Perron. The minimum diameter octagon with unit-length sides: Vincze's wife's octagon is suboptimal. *J. Combin. Theory Ser. A*, 108:63–75, 2004.



Regular octagon

$\Pi = 8\sin(\frac{\pi}{8}) \approx 3.0615...$

Equal sides

$\Pi \approx 3.0956...$

Unequal sides

$\Pi \approx 3.1211...$

Figure 5. Three small octagons

[4] C. Audet, P. Hansen, F. Messine, and J. Xiong. The largest small octagon. *J. Combin. Theory Ser. A*, 98(1):46–59, 2002.

[5] A. Bezdek and F. Fodor. On convex polygons of maximal width. *Arch. Math.*, 74(1):75–80, 2000.

[6] B. Datta. A discrete isoperimetric problem. *Geometriae Dedicata*, 64: 55–68, 1997.

[7] R. L. Graham. The largest small hexagon. *J. Combinatorial Theory Ser. A*, 18:165–170, 1975.

[8] D. G. Larman and N. K. Tamvakis. The decomposition of the n-sphere and the boundaries of plane convex domains. In *Convexity and graph theory (Jerusalem, 1981)*, volume 87 of *North-Holland Math. Stud.*, pages 209–214. North-Holland, Amsterdam, 1984.

[9] F. Messine. Deterministic Global Optimization using Interval Constraint Propagation Techniques. *RAIRO Operations Research*, Vol. 38, N. 4, 2004.

[10] F. Messine. A Deterministic Global Optimization Algorithm for Design Problems. In *C. Audet, P. Hansen, G. Savard (editors), Essays and Surveys in Global Optimization, Kluwer*, pp. 267-294, 2005.

[11] F. Messine., J.L. Lagouanelle. Enclosure Methods for Multivariate Differentiable Functions and Application to Global Optimization. *Journal of Universal Computer Science*, Vol. 4, (N.6) Springer-Verlag, pp. 589-603, 1998.

[12] R.E. Moore. Interval Analysis. *Prentice Hall, Englewood Cliffs, NJ*, 1966.

[13] H. Ratschek, J. Rokne. New Computer Methods for Global Optimization. *Ellis Horwood, Chichester*, 1988.

[14] K. Reinhardt. Extremale polygone gegebenen durchmessers. *Jahresber. Deutsch. Math. Verein*, 31:251–270, 1922.

[15] N.K. Tamvakis. On the perimeter and the area of the convex polygon of a given diameter. *Bull. Greek Math. Soc.*, 28: 115–132, 1987.

[16] S. Vincze. On a geometrical extremum problem. *Acta Sci. Math. Szeged*, 12:136–142, 1950.

[17] D.R. Woodall Thrackles and Deadlock. In *Combinatorial Mathematics and Its Applications* (D. J. A. Welsh, Ed.), Academic Press, New York, 1971.

Analysis of a nonlinear optimization problem related to semi-on-line bin packing problems

János Balogh[1], József Békési[1], Gábor Galambos[1], and Mihály Csaba Markót[2]

[1] *University of Szeged, Faculty of Juhász Gyula Teacher Training College, Department of Computer Science, 6701 Szeged, POB. 396. Hungary*
{balogh, bekesi, galambos}@jgytf.u-szeged.hu

[2] *Advanced Concepts Team, European Space Agency, ESTEC, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands*
Mihaly.Csaba.Markot@esa.int

Abstract In the paper we deal with lower bounds constructed for the asymptotic performance ratio of semi-on-line bin packing and batched bin packing algorithms. We determine the bounds as the solutions of a related nonlinear optimization problem using theoretical analysis and a reliable numerical global optimization method. The results improves the lower bounds given in [9] for some special cases of the batched bin packing problem (fixed, finite number of different elements in two batches), answering a question raised in [9] regarding the optimal bounds.

Keywords: semi-on-line bin packing problems, nonlinear optimization, branch–and–bound, interval arithmetic

1. Introduction

Bin packing is a well-known combinatorial optimization problem. In the classical one–dimensional case we are given a set of items represented by a list $L = \{x_1, x_2, \dots, x_n\}$ of real numbers in $(0, 1]$, and an infinite list of unit capacity bins. Each item x_i has to be assigned to a unique bin such that the sum of the elements in each bin does not exceed 1. Our aim is to minimize the number of used bins. It is well-known that finding an optimal packing is *NP-hard* [7]. Consequently, large number of papers have been published which look for polynomial time algorithms with an acceptable approximative behavior. The *on-line bin packing algorithms* put items into a bin as they appear without knowing anything about the subsequent elements (neither the sizes nor the number of the elements). *Off-line algorithms* can use more information: most of them examine the entire list before they apply their strategy to pack the items. The so called *semi-on-line algorithms* [2] are between the on-line and off-line ones. For such algorithms at least one of the following operations is allowed: repacking of some items [4–6,11], lookahead of the next several elements [8], or some kind of preordering.

The efficiency of different algorithms is generally measured by two different methods: the investigation of the worst-case behavior, or – assuming some probability distribution of the elements – a probability analysis. In this paper we will concentrate on the asymptotic worst-case ratio which can be defined as follows: denote $A(L)$ the number of bins used by the (either repacking or non-repacking) algorithm A to pack the elements of a list L, and let L^* the number of bins in an optimal packing. If

$$R_A(k) := \max\left\{\frac{A(L)}{k} | L^* = k\right\} \tag{1}$$

denotes the maximum ratio of $A(L)/L^*$ for any list L with $L^* = k$, then the *asymptotic performance* *ratio (APR)* R_A of the algorithm A is defined as

$$R_A := \limsup_{k\to\infty} R_A(k). \tag{2}$$

The best known lower bound for the APR of any on-line bin packing algorithm A is 1.54014 (given by Van Vliet [17]), while for the current best algorithm has an APR of 1.58889 (Seiden, [16]).

The general semi-on-line bin packing problem with repacking was studied by Gambosi et al. [6]. Here the expression 'general' means that the number of repackable elements per step is not restricted (bounded) in a strict way by a constant. They gave an $O(n)$ time algorithm with an APR of 1.5 and an $O(n \log n)$ time algorithm with an APR of $\frac{4}{3}$. The latter algorithm was improved to an algorithm with APR of 1.25 by Ivkovič and Lloyd [12]. Note, that none of the above mentioned algorithms repack constant number of elements in a strict sense, because they define the cost of repacking a bundle of small items as a constant.

The only lower bound for this problem is proved by Ivkovič and Lloyd. This bound is $\frac{4}{3}$. In their lower bound construction the repacking of a constant number of items is allowed after the arrival of each new item. The $\frac{4}{3}$ lower bound is proved for fully dynamic bin packing with restricted repacking. *Dynamic bin packing* means that in each step not only the insertion of the arrived elements (Insert operation) is allowed, but in any step one element can be deleted (Delete operation) instead of Insert. The *fully dynamic bin packing* is such a version of the dynamic bin packing when repacking is allowed. Although the $\frac{4}{3}$ lower bound was constructed for this particular fully dynamic bin packing problem, the model can be easily applied for the similar classical (i.e. repacking, but not dynamic) semi-on-line bin-packing problem. For more details, see the survey of Csirik and Woeginger [3]).

A similar problem class called *batched bin packing* is defined by Gutin et al. [9]: in this case the items become available in batches, and each batch must be packed before the next batch arrives. By defining and solving a related nonlinear optimization problem we will generalize the solution method of [9] for deriving lower bounds.

In this paper we improve the lower bound $4/3$ for that of the variant of the on-line bin packing problem when in each step the repacking of constant number of elements is allowed. The same construction can be used for deriving the same lower bound for the similar version of the fully dynamic bin packing problem. Improved lower bounds will be obtained by solving a nonlinear optimization problem. The solution of the special cases of the same nonlinear optimization problem is interesting as well, because their solutions answer some questions raised in [9] for the batched bin packing problem, namely, for the two batched bin packing problem, when the number of different item sizes is at most p. In [9] it is shown that the lower bound $r(p)$ is the best possible in the case of $p = 2$, but it was not clear whether the bounds $r(p)$ given for cases of $p \geq 3$ might be the best possible bounds. Here we are answering the question regarding the optimality of the given bound for these values. We note that the lower bounds for these special cases are valid for the above mentioned two problems, too.

2. Constructing the lower bounds by a linear and a nonlinear optimization problem

The following theorem is proved in [1]:

Theorem 1. *[1] Let $k \geq 1$ and $c \geq 1$ be arbitrary integers and $x_1, x_2, \ldots, x_k$, $(\frac{1}{2} \leq x_1 < x_2 < \cdots < x_k < 1)$ be fixed real numbers. Let $y_i = 1 - x_i$, $(i = 1, \ldots, k)$ and $y_{k+1} = 0$. Then the solution of the following linear programming problem is a lower bound for the APR of an arbitrary semi-online* *bin packing algorithm with c-repacking:*

$$\begin{array}{rrcll}
 & \min & b, & & (3)\\
\text{subject to} & b & \geq & 1+y_i+2y_i\left(\sum_{j=1}^{i-1} z_j\left(\frac{1}{y_j}-1\right)-\sum_{j=i}^{k} z_j\right),\ (i=1,\ldots,k), & (4)\\
 & b & \geq & 1+\sum_{j=1}^{k} 2z_j\left(\frac{1}{y_j}-1\right), & (5)\\
 & z_i & \geq & 0,\ (i=1,\ldots,k), & (6)\\
 & \sum_{j=1}^{k} z_j & < & \frac{1}{2}. & (7)
\end{array}$$

Note that the expression c-repacking means that a given number c of elements can be repacked in each step.

We omit the proof of this theorem, but we introduce a list-construction as the basic idea behind the proof: consider a series of lists $L_1, ..., L_k$, where L_j $(j=1,\ldots,k)$ contains $\left\lceil \frac{n}{2y_j} \right\rceil$ items of size $x_j+\varepsilon_j$, where $\varepsilon_j := \frac{\varepsilon}{\left\lceil \frac{n}{2y_j} \right\rceil}$, and $\varepsilon < \min_{j=1,\ldots,k}\{y_j - y_{j+1}\}$ is an arbitrary positive number. L_0 is defined as a list of M items of size a, where $a < \frac{\varepsilon_k}{\left\lceil \frac{n}{y_k} \right\rceil c}$ and $M := \left\lfloor \frac{\frac{n}{2}-\varepsilon}{a} \right\rfloor$. It can be seen that $size(L_j)$, i.e. the sum of the elements in L_j is $\frac{n}{2}+\varepsilon$, while $size(L_0)$ is $\frac{n}{2}-\varepsilon$.

We can see that the size of the equal elements $a = a(\varepsilon)$ in L_0 is a very small number, and the total size of the repackable items in $\left\lceil \frac{n}{2y_j} \right\rceil$ steps is less than ε_j $(j = 1.\ldots,k)$. The key idea of the construction is the following: if a is small, – as it is defined above – then considering the list-concatenations $L_0L_1, L_0L_2, \ldots, L_0L_k$, the total size of the repackable small elements during the packing of the second list L_j is less than ε_j.

This way we "almost switch off" the role of the repacking. It is directly follows from the size of any "big" element x_j $(= 1 - y_j)$, that such a big element can be packed only in a bin, which level is at most $y_j - \varepsilon_j$. If we denote by $z_i n$ the cumulative size of the items that are packed in y_i-type bins, – we call a bin B y_i-type bin if $size(B) \in (y_{i+1}, y_i]$ – then because of the above reasoning a bin containing a big element had level of at most y_j after the packing of L_0.

Based on the principle of the above construction, we can now estimate (2) for L_0 and for $L_0L_j,\ j=1,\ldots,k$. Namely, equation (4) of Theorem 1 comes from the estimation of (2) for the list concatenations $L_0L_j, \forall j$, while equation (5) comes from the estimation of (2) for the case list L_0 (i.e. for the case when there is no list appearing after L_0). By picking any $y_1,\ldots,y_k$ values for which the condition of the Theorem 1 hold $(0 < y_k < \ldots < y_1 \leq 0.5)$ we get a lower bound for any of the three problems we are dealing with, and any such y-system will result in a lower bound for the problem (3)–(7).

First, consider the case of $k = 1$ and $y_1 = 0.5$. In this case we get a very simple system of equations, and the optimal solution will be $\frac{4}{3}$, getting back the result of [11] as a special case of our construction. It can be shown, that in the case of $k = 1$ this is the best lower bound, which can be given by our construction. On the other hand, by fixing larger k $(k \geq 2)$ and y_j $(j = 1,\ldots,k)$ values, for which the condition of the Theorem 1 hold $(0 < y_k < \ldots < y_1 \leq 0.5)$, the bound given in [11] can be improved. Note, that for $k \geq 2$ it is enough to consider $y_j \geq 0.25$, since in the case of $k = 1$ with $y_1 = 0.5$ the lower bound was $\frac{4}{3}$. Hence, to get better lower bounds the elements x_j $(j = 1,\ldots,k)$ in the lists $L_1,\ldots,L_k$ have to be smaller than 0.75 (otherwise the algorithm would put the larger elements alone into one bin). So in the following we can assume that

$$y_j \geq 0.25 \text{ for any } y_j, j = 1,\ldots,k,\ k \geq 1. \tag{8}$$

Concerning the choice of the y values, the most obvious method is to consider the $y_1, \ldots, y_k$ system as k equidistant points in the $(0.25, 0.5]$ interval, i.e. $y_j := 0.5 - \frac{0.25(j-1)}{k}$, $j = 1, \ldots, k$. For this y-system, solving the received LP (if we fix y-s, (9) is an LP), we get the lower bounds for (9). The results are displayed in the third column of our Table 1. In this way all values of the third column of our Table 1 improve the lower bound for the on-line bin packing problem with restricted repacking and for the same version of fully dynamic bin packing as well.

Our construction (without considering repacking) can be used for the two batched bin packing problem ($2 - BBPP$, [9]). in cases when p different sizes are allowed. In our construction this means $p = k + 1$ lists, because of the list L_0.

In this way, if we used the aforementioned equidistant points as the y-system of our construction, we exactly get back the results of Gutin at al. for the two batched bin packing problem when p number of different elements are allowed. The third column of our Table 1 contains the values which are corresponding to their Table on page 77 of [9] for the two batched bin packing problem.

Note that the construction in [9] was similar, but the main point is different: they used only equidistant y_j-values (in our terminology), while we can consider any y-systems (allowing not only equidistant points). In this way our construction can be considered as a generalization of the one of [9]. The construction of [9] placed some elements of size s in the list L_0, $0 < s < \frac{1}{2}$, while L_j contained n/j elements, each of size $1 - js$ ($j = 1, \ldots, m$, where $m = \lfloor \frac{1}{s} \rfloor$). In our construction $x_j = 1 - js$, and $y_j = js$, respectively. (Note, that since our first list has an index 0, their p value corresponds to $k + 1$ in our construction; see Table 1, columns 1–2.)

The essence of the results of the LP approach (either considering equidistant y-systems or not) is that the obtained bounds are valid for the two above mentioned semi-on-line bin packing problems, allowing restricted repacking.

In the following we answer a question raised in [9], namely, whether their bound given for the cases when only p ($p \geq 2$) different elements are allowed, can be improved? In our context, the corresponding problem is whether we can produce better lower bounds from our construction, as compared to the ones obtained from the equidistant y-points. This question leads us to a nonlinear optimization problem which can be described in a short form using the objective function and constraints of Theorem 1:

$$\begin{array}{ll} \max & b^*(y_1, \ldots, y_k), \\ \text{subject to} & 0.25 \leq y_k < \ldots < y_1 \leq 0.5, \end{array} \tag{9}$$

where $b^*(y_1, \ldots, y_k)$ denotes the optimal solution of the minimization problem (3) subject to the constraints (4-7) for a fixed system of $y_1, \ldots, y_k$. Recall that the $0.25 \leq y_k$ restriction is allowed due to (8). The obtained problem (9) is a so called max-min problem. Nevertheless, if we fix the y-s, then we get back the LP (3-7), and again any y-values deliver a lower bound. Now, the question is: which y-system should be considered to maximize (9).

3. The analysis and solution of the nonlinear optimization problem – asymptotic behavior and special cases

Problem (9) seems to be very hard to solve from a global optimization point of view in its presented form. Following the transformations introduced in details in [1], we obtain an equivalent, but more convenient form (with one single objective and easily manageable bound constraints for the variables):

$$\begin{array}{ll} \max & f(y) = 1 + \dfrac{1 - y_k}{y_k + \frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} - k}, \\ \text{subject to} & 0.25 \leq y_k < \ldots < y_1 \leq 0.5. \end{array} \tag{10}$$

Lemma 2. *Let $k \geq 2$ be an arbitrary, but fixed integer. Assume that the one-dimensional optimization problem*

$$\begin{aligned} \max \quad & t(y_k) = 1 + \frac{1 - y_k}{y_k + 2 - k - (k-1)(2y_k)^{-1/(k-1)}}, \\ \text{subject to} \quad & 0.25 \leq y_k \leq 0.5 \end{aligned} \tag{11}$$

has a unique optimal solution $y_k = y^* \in [0.25, 0.5]$ with $t(y^*) = t^*$. Then the k–dimensional problem (10) also has a unique optimal solution, and it is given by

$$y_1 = \frac{1}{2}, \ \ y_i = \frac{1}{2}(2y^*)^{\frac{i-1}{k-1}}, \ i = 2, \ldots, k.$$

Moreover, the optimum of the original problem also equals to t^*.

For this modified form of the original problem the following statements hold:

Lemma 3. *[1] The optimal solution of (11) converges to the maximum of the function*

$$f\,(x) := 1 + \frac{1 - x}{x + 1 + \ln\left(\frac{1}{x}\right) - \ln\left(2\right)} \tag{12}$$

in the interval $[\frac{1}{4}, \frac{1}{2}]$ if $k \to \infty$.

Lemma 4. *[1] The maximum of the function $f(x)$ is $1 - \frac{1}{W_{-1}\left(\frac{-2}{e^3}\right)+1} \approx 1.3871$ in the interval $[\frac{1}{4}, \frac{1}{2}]$, where $W_{-1}\,(x)$ is a branch of the Lambert function.*

We solved (11) with a branch–and–bound global optimization algorithm using interval arithmetic calculations ([10, 13–15]) for the parameter values of $k = 2, \ldots, 10, 20, 50, 100, 1000$. In each case, we have managed to verify the existence and uniqueness of y^*, and we have obtained the guaranteed interval enclosures of both y^* and t^* with high precision. The enclosure of y^* allowed us to verify the additional prerequisite $y^* \leq 0.5$ as well, and thus, to construct the interval enclosures of the unique solution of the original problem. The fourth column of Table 1 contains the the optimum values for the solved problem instances up to 12 digits. Due to the precision of the interval calculations there is a guarantee that the results are accurate in the displayed digits.

Table 1. Improved lower bounds for a variation of the 2-batched bin packing problem when only $k+1$ item sizes are allowed known in advance.

k	p	*Equidistant points (y-s)*	*Arbitrary points (y-s)*
1	2	1.3333...	—
2	3	1.3658...	1.36602540378...
3	4	1.3738...	1.37393876913...
4	5	1.3773...	1.37753136189...
5	6	1.3793...	1.37958528769...
6	7	1.38051	1.38091512540...
7	8	1.38136	1.38184652163...
8	9	1.38198	1.38253525895...
9	10	1.38246	1.38306525702...
10	11	1.38296	1.38348573275...
20	21	1.38509	1.38534022765...
50	51	1.38631	1.38642436208...
100	101	1.38673	1.38678113846...
1000	1001	1.38709	1.38710030535...

4. Summary

The paper dealt with a nice connection between a discrete (combinatorial) optimization problem and global optimization. The lower bound constructions given for special bin packing problems led to a nonlinear optimization problem, and after some transformations the solutions were delivered. Our construction is more general than the ones of [11] and [9], and its special cases improved the results of [9] by answering the questions raised in the paper.

Acknowledgments

The research was supported by the Hungarian National Research Fund (projects T 048377 and T 046822) an by the MÖB-DAAD Hungarian-German Researcher Exchange Program (project No. 21). The authors wish to thank to Tamás Vinkó for his ideas and for the first, previous numerical tests.

References

[1] J. Balogh, J. Békési, and G. Galambos. Lower Bound for Bin Packing Problem with Restricted Repacking. Manuscript, 2005. Available at `http://www.jgytf.u-szeged.hu/~bekesi/crestbinp.ps`.

[2] E.G. Coffman, G. Galambos, S. Martello, and D. Vigo. Bin Packing Approximation Algorithms: Combinatorial Analysis. In *Handbook of Combinatorial Optimization* (Eds. D.-Z. Du and P.M. Pardalos), pages 151-208. Kluwer Academic Publishers, 1999.

[3] J. Csirik and G.J. Woeginger. Online packing and covering problems. In: *Online Algorithms: The State of the Art*, Eds. A. Fiat and G.J. Woeginger, pages 147-177, Berlin, 1998. Springer-Verlag, LNCS 1442.

[4] G. Galambos. A new heuristic for the classical bin packing problem. Technical Report 82, Institute fuer Mathematik, Augsburg, 1985.

[5] G. Galambos and G.J. Woeginger. Repacking helps in bounded space on-line bin packing. *Computing*, 49: 329-338, 1993.

[6] G. Gambosi, A. Postiglione, and M. Talamo. Algorithms for the Relaxed Online Bin-Packing Model. *SIAM J. Computing*, 30(5): 1532-1551, 2000.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability (A Guide to the theory of NP-Completeness).* W.H. Freeman and Company, San Francisco, 1979.

[8] E.F. Grove. Online bin packing with lookahead. *SODA* 1995: 430–436.

[9] G. Gutin, T. Jensen, and A. Yeo. Batched bin packing. *Discrete Optimization*, 2(1): 71-82, 2005.

[10] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[11] Z. Ivkovič and E.L. Lloyd. A fundamental restriction on fully dynamic maintenance of bin packing. *Information Processing Letters*, 59(4): 229-232, 1996.

[12] Z. Ivkovič and E.L. Lloyd. Fully Dynamic Algorithms for Bin Packing: Being (Mostly) Myopic Helps. *SIAM J. Computing*, 28(2): 574-611, 1998.

[13] M.Cs. Markót, T. Csendes, and A.E. Csallner. Multisection in interval branch-and-bound methods for global optimization. II. Numerical tests. *Journal of Global Optimization*, 16(3): 219-228, 2000.

[14] R.E. Moore. *Interval Analysis*. Prentice–Hall, Englewood Cliffs, 1966.

[15] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester, 1988.

[16] S.S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5): 640-671, 2002.

[17] A. Van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43(5): 277-284, 1992.

A global optimization model for locating chaos: numerical results*

Balázs Bánhelyi[1], Tibor Csendes[1], and Barnabás Garay[2]

[1] *University of Szeged, Szeged, Hungary,* banhelyi|csendes@inf.u-szeged.hu

[2] *Budapest University of Technology, Budapest, Hungary,* garay@math.bme.hu

Abstract We present a computer assisted proofs for the existence of so-called horseshoes of the different iterates of the classical Hénon map ($\mathcal{H}(x, y) = (1 + y - \alpha x^2, \beta x)$). An associated abstract provides algorithms and the theoretical basis for the checking of three geometrical conditions to be fulfilled by all points of the solution region. The method applies interval arithmetic and recursive subdivision. This verified technique proved to be fast on the investigated problem instances. So we were able to solve some unsolved problems, the present talk will summarize these computational results.

Keywords: Chaos, Hénon-mapping, Computer-aided proof.

After having introduced our computational methodology to locate chaotic regions (see the associated abstract and the papers [4] and [1]), now we concentrate on the obtained numerical results. First we have checked the reported chaotic region [6] by our checking routine.

We have investigated the seventh iterate of the Hénon mapping with the parameters of $A = 1.4$ and $B = 0.3$. The checked region consists of two parallelograms with sides parallel to the x-axis, the first coordinates of the lower corner points were 0.460, 0.556, 0.588, and 0.620, while the second coordinates were the same, 0.01. The common y coordinate for the upper corner points was 0.28. The tangent of the sides was 2. We have set the ε threshold value for the checking routine to be 10^{-10}.

First the algorithm determined the starting interval, that contains the region to be checked:

$$[0.46000000000, 0.75500000000] \times [0.01000000000, 0.28000000000].$$

Then the three conditions were checked one after the other. All of these proved to be valid — as expected. The amount of function evaluations (for the transformation, i.e. for the seventh iterate of the Hénon mapping in each case) were 273, 523, and 1613, respectively. The algorithm stores those subintervals for which it was impossible to prove whether the given condition holds, these required further subdivision to achieve a conclusion. The depth of the stack necessary for the checking was 11, 13, and 14, respectively. The CPU time used proved to be negligible, only a few seconds.

Then, We have applied the global optimization model for the 5th iterate Hénon mapping. Note that the less the iteration number, the more difficult the related problem: no chaotic regions were reported for the iterates less than 7 till now. We have solved the optimization problem with a clustering method [2]. After some experimentation, the search domain set for

*This work has been partially supported by the Bilateral Austrian-Hungarian project öu56011 as well as by the Hungarian National Science Foundation Grants OTKA No. T 037491, T 032118, T 034350, T 048377, and T 046822.

the parameters to be optimized was:

$$A \in [1.00, 2.00],$$

$$B \in [0.10, 1.00],$$

$$x_a, x_b, x_c, x_d \in [0.40, 0.64].$$

Table 1 presents the numerical results of the ten search runs.

Table 1. Numerical results of the search runs

LO	ZO	FE	PE	T
12	4	13,197	4,086	17
12	1	12,913	3,365	16
12	1	13,569	4,303	19
12	2	12,918	3,394	16
12	1	14,117	5,083	18
12	3	21,391	7,400	25
12	2	12,623	3,296	16
12	0	15,388	6,221	30
12	3	13,458	3,858	15
12	2	14,643	5,002	16

Here LO stands for number of local optima found, ZO for the number of zero optimum values, FE for the number of function evaluations, PE for the number of penalty function evaluations, and finally T for the CPU time used in minutes.

One example of the obtained optimized parameter values is as follows:

$$A = 1.7484856, B = 0.3784193,$$

$$x_a = 0.4379310, x_b = 0.5143267,$$

$$x_c = 0.5661056, x_d = 0.6339521.$$

Finally we consider the 3rd iterate of Hénon-mapping. The first successful run of our global optimization algorithm involved beyond the earlier six parameters also the angle α, and 3 coordinates of the set E. The numerical results with the ten parameters (see also Figure 1):

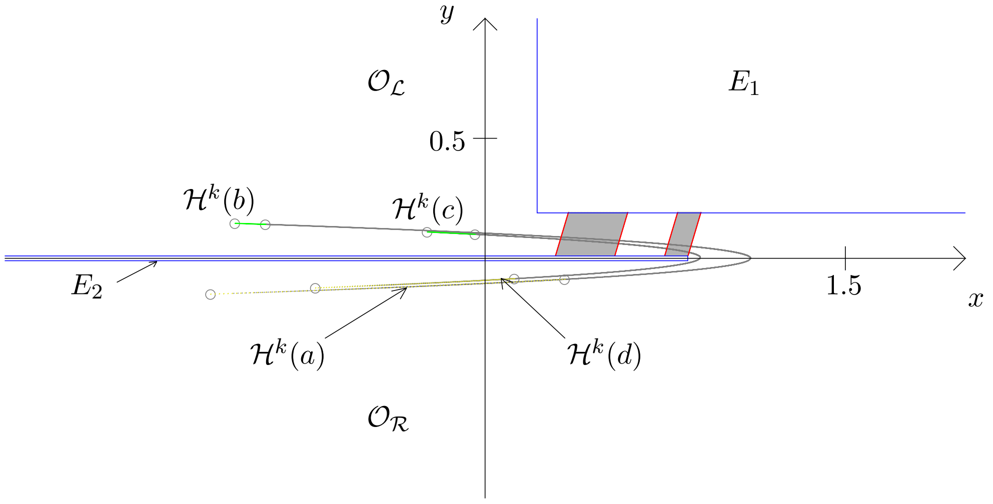


Figure 1. Illustration of the H^3 transformation with the obtained chaotic region of two parallelograms.

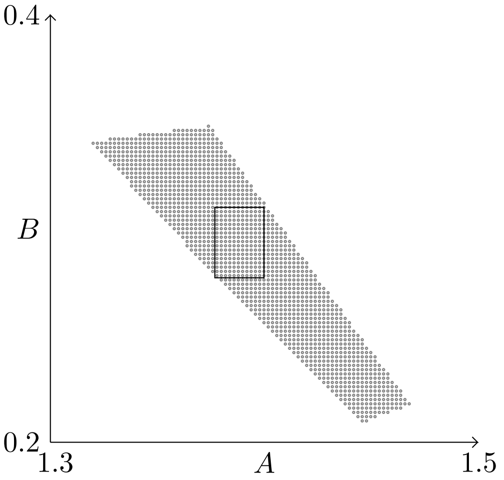


Figure 2. Illustration of the obtained interval containing only such A and B values that ensure a chaotic region in the classic two parallelograms. Those grid points that also fulfill the conditions are denoted by small circles.

$$A = 2.5569088, B = 0.15963498,$$

$$\tan \alpha = 3.3579163,$$

$$x_a = 0.29188440, x_b = 0.53887296,$$

$$x_c = 0.74663494, x_d = 0.84359572,$$

$$E_{1,bottom} = 0.18937143, E_{1,left} = 0.21673342, E_{2,right} = 0.84386042.$$

Beyond the above mentioned results, we have achieved intervals of positive measure containing exclusively feasible points for our constraint satisfaction problem with a tolerance optimization method [3]. As an illustration of the results see Figure 2.

References

[1] Bánhelyi, B., T. Csendes, and B.M. Garay: Optimization and the Miranda approach in detecting horseshoe-type chaos by computer. Manuscript, submitted for publication. Available at
`www.inf.u-szeged.hu/∼csendes/henon2.pdf`

[2] Csendes, T.: "Nonlinear parameter estimation by global optimization – efficiency and reliability", *Acta Cybernetica* **8**, 361–370, 1988.

[3] Csendes, T., Zabinsky, Z. B., and Kristinsdottir, B. P.: "Constructing large feasible suboptimal intervals for constrained nonlinear optimization", *Annals of Operations Research* **58**, 279–293, 1995.

[4] Csendes, T., Garay, B. M., and Bánhelyi, B.: "A verified optimization technique to locate chaotic regions of a Hénon system", Manuscript, submitted for publication. Available at
`www.inf.u-szeged.hu/∼csendes/henon.pdf`

[5] *C-XSC Languages home page:* `http://www.math.uni-wuppertal.de/org/WRST/index_en.html`

[6] Zgliczynski, P.: *Computer assisted proof of the horseshoe dynamics in the Hénon map.* Random & Computational Dynamics 5:1-17, 1997.

An Efficient Pruning Technique for the Global Optimisation of Multiple Gravity Assist Trajectories

V.M. Becerra[1], D.R. Myatt[1], S.J. Nasuto[1], J.M. Bishop[2] and D. Izzo[3]

[1]*Department of Cybernetics, The University of Reading, Reading RG6 6AY, United Kingdom*

[2]*Department of Computing, Goldsmiths College, London SE14 6NW, United Kingdom*

[3]*D. Izzo, Advanced Concepts Team, European Space Agency, Noordwijk, The Netherlands*

Abstract With application to the specific problem of multiple gravity assist trajectory design, a deterministic search space pruning algorithm is developed that displays both polynomial time and space complexity. This is shown empirically to achieve search space reductions of greater than six orders of magnitude, thus reducing significantly the complexity of the subsequent optimisation.

Keywords: mission design, multiple gravity assist, global optimisation, constraint propagation, heuristic search.

1. Introduction

A gravity assist manoeuvre uses a celestial object's gravity in order to change a spacecraft's trajectory. When a spacecraft approaches a celestial object, a small amount of the object's orbital momentum is transferred to the spacecraft. This manoeuvre was used for the first time in the 1970's, when the spacecraft Voyager used multiple gravity assist flybys of Jupiter, Saturn, Uranus and Neptune, to propel itself beyond these planets. Gravity assist manoeuvres (GAs) are frequently used to reduce fuel requirements and mission duration [1]. Most interplanetary trajectory design problems can be stated as optimisation problems, where one of the fundamental goals is the minimisation of fuel requirements, with consideration also given to intermediate planetary flybys, mission duration, type of arrival, launch and arrival windows, and velocity constraints. Traditionally, local optimisation has been used to attempt to solve these design problems [2,3]. However, because of nonlinearities and the periodic motion of the planets, multiple local minima exist and, as a result, local optimisation only helps to find local minima which are heavily dependent on the initial guesses employed and are not necessarily good solutions. The use of global optimisation techniques has been proposed for tackling these problems, as these methods have better chances of finding good solutions approaching the global optimum [4]. Genetic algorithms and similar techniques have been employed, but these techniques may face difficulties in tackling realistic missions due to the large size of the search space associated with these problems. This paper considers the problem of multiple gravity assist (MGA) trajectories with a known planetary sequence and no deep space manoeuvres. In such cases, it can be shown that the vast majority of the search space consists of infeasible, or very undesirable, solutions. This observation motivated the development of a method for producing reduced search spaces by pruning, thus allowing standard global optimisation techniques to be applied more successfully to the reduced box bounds [5]. The technique presented in this paper has been named Gravity Assist Space Pruning (GASP).

2. Gravity assist: space pruning algorithm

This section describes the motivation behind and functionality of the GASP algorithm. Consider the MGA problem with a defined planetary sequence (e.g. Earth-Venus-Venus-Earth-Jupiter-Saturn) and no deep space maneouvres. The decision vector for this problem is as follows

$$\mathbf{x} = \{t_0, t_1, t_2, t_3...\}, \tag{1}$$

where t_0 is the launch date, t_1 is the phase time from the first to second planet, t_2 from the second to third planet etc. An efficient Lambert solver [6] is used to calculate appropriate Keplerian orbits between the planetary positions in the given time, and then a powered swingby model is applied, such as that designed by Gobetz [7].

2.1 Single interplanetary transfer

Consider the simplest case of a single interplanetary transfer with a braking manoeuvre at the target planet. The objective function assumed is a simple minimisation of total thrust (the sum of the initial hyperbolic excess velocity, v_i, and braking manoeuvre, v_f), so

$$f = v_i + v_f. \tag{2}$$

The decision vector in the single transfer case will be $\mathbf{x} = \{t_0, t_1\}$. An important observation is that this search space will contain a line for each time t, that a probe can arrive at the final planet, such that $t_0 + t_1 = t$. Obviously, at a given time t, regardless of the launch time or departure time, the target planet will be *in the same position and have the same velocity*. Therefore, it is beneficial to consider the search space as t_0, $t_0 + t_1$, this is departure time at the first planet compared to arrival time at the second. The optimisation method to be investigated is grid sampling. Grid sampling is usually considered a very inefficient optimiser, particularly in high dimensionalities. For example, using the enumerative search in the Swingby Calculator application [8] yields optimisation times approaching an hour for relatively small search spaces (on a 600Mhz Pentium Processor). However, for only 1 and 2 dimensions grid sampling is computationally tractable, as long as the objective function is reasonably smooth and the exact optimum is not required. Therefore, the objective function for a single interplanetary transfer may be grid sampled at an appropriate resolution in the departure time vs arrival time domain efficiently, although in this case most other optimisation methods would yield better results in terms of objective function evaluations. However, the grid sampled version will require many less Ephemeris calculations, as the same positions/velocities need not be recalculated for a given departure or arrival time. If the 2D search space was discretised into k cells in each dimension, only $2k$ Ephemeris calculations are required for the entire sampling, and k^2 Lambert problem solutions. By comparison, two Ephemeris calculations would be required by each objective function evaluation in a standard optimiser. Even in the single interplanetary transfer case, a large proportion of the search space corresponds to undesirable solutions i.e. those with impractical C3[1]. To illustrate this, the optimisation of an Earth-Mars transfer was considered between the dates (-1200 to 600 MJD2000) and phase times of 25 to 515 days. A sampling resolution of 10 days was used in both axes. Only 12.5% of this search space had a C3 of less than 25km^2/s^2. Figure 1 shows this search space plotted as departure time vs arrival time - the diagonal lines delineate the sampled portion of the search space, and the dark regions within the lines indicate trajectories with a fasible C3 value lower than 25km^2/s^2.

As a consequence, in gravity assist and multiple gravity assist cases starting with an Earth-Mars transfer in these bounds, at least 87.5% (100%-12.5%) of the overall search space *must*

[1] C3 (units km^2/s^2) is the square of the hyperbolic excess velocity.

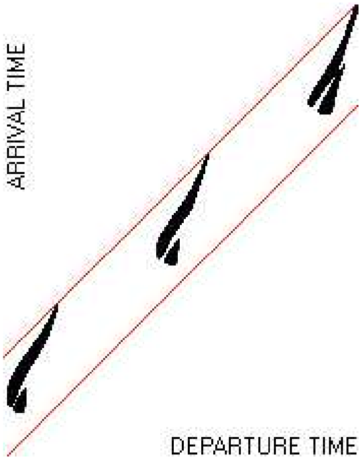


Figure 1. A grid sampled Earth-Mars transfer. The white regions within the delineating diagonal lines indicate solutions with a C3 of greater than 25km^2/s^2

correspond to undesirable solutions. Even allowing an enormous C3 of 100km^2/s^2, only 33% of the search space becomes valid. The GASP algorithm was design to efficiently detect and prune infeasible parts of the space, leaving several sets of box bounds with vastly smaller contents. These reduced box bounds may then be optimised efficiently using a standard optimisation method.

Hyperbolic Excess Velocity Constraint: The maximum allowable hyperbolic excess velocity is the first main constraint of the GASP algorithm, as it determines possible launch dates to the first target planet.

Braking Manoeuvre Constraint: As well as the C3 constraint, it is logical to add a constraint on the maximum braking manoeuvre that the spacecraft can perform. Applying a C3 constraint of 25km^2/s^2 and a braking manoeuvre constraint of 5km/s it can be estimated that less than 5% of the search space yields feasible trajectories. By applying two very simple constraints to the interplanetary case it has been shown that a very significant reduction in search space can been achieved, leaving clear launch windows and arrival time windows.

2.2 Forward Constraining

It has been shown that the C3 and braking manoeuvre constraints alone significantly reduce the search space content for an interplanetary transfer. From Figure 1, it can be seen that for many values of the arrival time there are no feasible departure times. This observation is the key principle on which the GASP algorithm is based: if no feasible trajectories arrive at a planet on a given date then there can be no departures from the planet on that date (assuming the change in velocity from the swingby is instantaneous). Now consider a trajectory with a single gravity assist. Using grid sampling on this function would usually involve sampling in three dimensions, and hence as additional planets were added the number of objective function evaluations would increase exponentially. Instead, with GASP, the search space is sampled as a cascade of two dimensional search spaces, each with possible departure dates (in the horizontal axis) and prospective arrival dates (in the vertical axis). Because of this, the number of Lambert problem evaluations is vastly reduced.

2.3 Gravity assist thrust constraint

Two constraints are added in order to maximise the probability of gravity assists being feasible. The first such constraint is the gravity assist thrust constraint, which limits the maximum absolute difference between incoming and outgoing velocities during a gravity assist to some threshold, T_v. This threshold is set separately for each gravity assist. The following is then performed for *every* arrival time at a planet:

1. Calculate the bounds on incoming velocity, v^i_{min} and v^i_{max}.
2. Invalidate any outgoing trajectories that do not have outgoing velocities in the range $[v^i_{\text{min}} - T_v - L_v, v^i_{\text{max}} + T_v + L_v]$, where L_v is an appropriate tolerance based on the Lipschitzian constant of the current phase plot.
3. Calculate the modified bounds on outgoing velocity, v^f_{min} and v^f_{max}.
4. Invalidate any incoming trajectories with velocities outside the range $[v^f_{\text{min}} - T_v - L_v, v^f_{\text{max}} + T_v + L_v]$.

2.4 Gravity assist angular constraint

The gravity assist angular constraint removes infeasible swingbys from the search space on the basis of them being associated with a hyperbolic periapse under the minimum safe distance for the given gravity assist body. This is determined over every arrival date at a planet as follows, assuming i valid incoming trajectories and j valid outgoing trajectories:

```
1. For all $i$ incoming trajectories
2.     For all $j$ incoming trajectories
3.         If the swingby is valid for the current incoming
           and outgoing trajectory, mark both incoming and
           outgoing trajectory as valid.
4.     End
5. End
6. Invalidate all trajectories not marked as valid
```

The swingby angle is decreased by an appropriate Lipschitzian tolerance θ_L, in order to compensate for the effects of the grid sampling of the search space.

3. Time and space complexity

This section determines the time and space complexity of the GASP algorithm. It will be shown that GASP scales quadratically in space and quartically in time with respect to the number of gravity assist manoeuvres considered. For simplicity, the following analysis assumed that the initial launch window and *all* phase times are the same.

3.1 Space Complexity

Consider a launch window discretised into k bins and a mission phase time also discretised into k bins. For the first phase k^2 Lambert problems must be sampled. The next phase will need to sample $(k+k)k = 2k^2$, as the number of possible times that the planet may be arrived at is doubled (minimum launch date, minimum phase time to maximum launch date, maximum phase time). The third phase will require $3k^2$ Lambert function evaluations, and the n^{th} phase nk^2. This gives the series

$$O(n) = k^2 + 2k^2 + 3k^2 + \ldots + nk^2 \tag{3}$$

$$O(n) = k^2(1 + 2 + 3 + \ldots + n) \tag{4}$$

$$O(n) = k^2 \frac{n(1+n)}{2}. \tag{5}$$

Therefore, the amount of space required for n phases is only of the order $O(n^2)$, rather than $O(k^n)$ for full grid sampling.

Similarly, it is clear that the space complexity with respect to the resolution k, is also of the order $O(k^2)$.

3.2 Time Complexity

The memory space required is directly proportional to the maximum number of Lambert problems that must be solved, and hence the time complexity of the sampling portion of the GASP algorithm must also be of the order $O(n^2)$.

Launch energy constraint complexity: The launch energy constraint is only applied in the first phase, and hence is independent of the number of swingbys. The time complexity is $O(k^2)$ with respect to resolution.

Gravity assist thrust constraint complexity: The time complexity of applying the gravity assist thrust constraint is $O(n^2)$ with respect to dimensionality (number of phases), due to the inevitable increase in size of later phase plots to encompass all possible arrival dates. The first phase requires of the order of $2k \times (k + 3k)$ operations in order to perform the constraining of outgoing velocity from incoming velocity (the back constraining may be ignored at this point). The second phase requires of the order of $3k \times (2k + 4k)$ operations. In general, the n^{th} phase requires of the order of $2n^2k^3$ operations. Therefore, the total number of operations over all phases is

$$2k^2[2^2 + 3^2 + 4^2 + \ldots + n^2] = 2k^2 \frac{n(n+1)(2n+1)}{3} \tag{6}$$

Therefore, applying this constraint yields cubic time complexity in dimensionality and quadratic complexity in resolution.

Gravity assist angular constraint complexity: The maximum number of swingby models that must be calculated for the first phase is close to $k \times 2k \times 3k = 6k^3$. For the second swingby, this is $2k \times 3k \times 4k = 24k^3$. In general, for n phases, the upper bound on the number of swingby calculations, α, is

$$\begin{aligned} \alpha = \quad & 3 \times 2 \times 1 \times k^3 + 4 \times 3 \times 2 \times k^3 \\ & +5 \times 4 \times 3 \times k^3 + \ldots + (n+2)(n+1)nk^3 \end{aligned} \tag{7}$$

From [9], it can be shown that the total number of these operations must be

$$\alpha = k^3 \sum_{j=1}^{n} (j+2)(j+1)j = k^3 \frac{n(n+1)(n+2)(n+3)}{4}. \tag{8}$$

Therefore, the overall time complexity with respect to resolution is $O(k^3)$, while the time complexity with respect to dimensionality is $O(n^4)$. Therefore, the gravity assist angular constraint is the most computationally expensive and hence is applied after GA thrust constraint in order to minimise the number of swingby models that must be calculated.

Overall time complexity: The overall time complexity, taken from the most complex part of the algorithm (the gravity assist angular constraint), is cubic with respect to resolution and quartic with respect to dimensionality.

4. Differential evolution

Differential Evolution (DE) [10] is a novel incomplete probabilistic global optimiser based on Genetic Algorithms [11], and was the highest ranked GA-type algorithm in the First International Contest on Evolutionary Computation. Following [10], scheme DE1 is used in this work as the crossover operator as it has been shown to perform the best on the most complex test function examined.

5. Results

This section demonstrates the improvements that GASP can make over Differential Evolution alone in one test case. Consider the optimisation of an EVVEJS transfer with an orbital insertion, where the objective function is the minimisation of the sum of the launcher and probe thrust. The bounds on the decision vector were as follows:

- $t_0 \in [-1200, 600]$ MJD2000
- $t_1 \in [14, 284]$ days
- $t_2 \in [22, 442]$ days
- $t_3 \in [14, 284]$ days
- $t_4 \in [99, 1989]$ days
- $t_5 \in [366, 7316]$ days

GASP was applied to this problem with a sampling resolution of 10 days. In order to complete the sampling, 144498 Lambert problem solutions and 3749 Ephemeris calculations were required. The following constraints were defined in the GASP algorithm:

- $T_{HEV} = 8000m/s$
- $T_{GA_{1....4}} = 1000m/s$
- $T_{\text{Brake}} = 5000m/s$

This configuration yields two major solution families, one with a launch window of -920 to -660MJD2000, and the other 280 to 490MJD2000. Differential Evolution was applied to the accumulation of each solution family (the tightest decision vector bounds that all solution family nodes exist within). A population of 40 individuals was used and a terminal number of 2000 iterations were allowed. Note that this corresponds to $40 \times 2000 \times 5 = 400000$ Lambert problem solutions and 480000 Ephemeris calculations. The later launch window was eliminated immediately as applying Differential Evolution did not yield any valid solutions. Further optimisation on this launch window has consistently optimised to the same invalid minima. The earlier launch window proved much more promising and, as a consequence, 20 optimisation trials were performed. Of these trials, 19 found the second best known optima to this problem (5225m/s) to within 1m/s, and one came close to the best known optimum, (4870m/s). Experimentation has shown this minimum has a very small basin of attraction with respect to this objective function, and is exceptionally hard to find even in the reduced search space. When applying Differential Evolution alone to the entire search space, only 7 out of the 20 trials found the second best minimum, and none the best known. Using GASP, it is apparent that there is an extremely high probability that at least the second best solution will be found. In subsequent trials, the objective function was altered to penalise any trajectory with hyperbolic excess velocity of greater than 3000m/s. Only two of 20 of the GASP constrained trials failed to find the best known solution to within 10m/s in this case (instead finding the second best one), while again 7 out of 20 optimisation of the entire domain found the second best solution, and none located the best. Again, these results highlight the significant advantages of using the GASP algorithm. Not only does it allow effective visualisation of the search space, but it drastically reduces the requirement for optimisation restarts in order to find good solutions, and at a fraction of the computational expense of an optimisation restart.

6. Conclusions

This paper has described the Gravity Assist Space Pruning algorithm, proved that it has both polynomial time and space complexity, and furthermore demonstrated that it produces significant benefits over optimising the entire domain with relatively little computational expense. Additionally, the GASP algorithm allows intuitive visualisation of a high dimensional search space, and facilitates the identification of launch windows and alternative mission options.

Acknowledgments

This work has been funded by the European Space Agency under Contract 18138/04/NL/MV, Project Ariadna 03/4101.

References

[1] Debban, T.J., McConaghy, T.T., Longuski, J.M., "Design and Optimization of Low-Thrust Gravity-Assist Trajectories to Selected Planets." AIAA 2002-4729, Astrodynamic Specialist Conference and Exhibit, 2002.

[2] Hargraves, C., and Paris, S., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation", Journal of Guidance Control and Dynamic Vol 10, pp. 338-342, 1987.

[3] Betts, J.T., "Practical Methods for Optimal Control Using Nonlinear Programming", Philadelphia: SIAM, 2001.

[4] Vasile, M., "Systematic-Heuristic Approach for Space Trajectory Design" Annals of the New York Academy of Science, Vol 1017, pp. 234-254, 2004.

[5] Myatt, D.R., Becerra, V.M., Nasuto, S.J., and Bishop, J.M., "Advanced Global Optimisation for Mission Analysis and Design." Final Report. Ariadna id: 03/4101. Contract Number: 18138/04/NL/MV, 2004. Available: http://www.esa.int/gsp/ACT/doc/ACT-RPT-ARIADNA-03-4101-Reading.pdf.

[6] Izzo, D., "Efficient Computation of Lambert Interplanetary Arcs", internal report ACT-RPT-4100-DI-ECLI01

[7] Gobetz, F.B., "Optimal transfer between hyperbolic asymptotes", AIAA Journal Vol. 1, No. 9, 1963.

[8] Biesbroek, R., Ancarola, B. "Optimisation of Launcher Performance and Interplanetary Trajectories for Pre-Assessment Studies", IAF abstracts, 34th COSPAR Scientific Assembly, The Second World Space Congress, Houston, TX, USA., p.A-6-07, 10-19 October, 2002.

[9] Riordan, J., Combinatorial Identities, Wiley, 1968.

[10] Storn, R. and Price, K., "Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11, pp.341-359, 1997.

[11] Holland, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

Proceedings of GO 2005, pp. 47 – 52.

Calibration of a Gravity–Opportunity Model of Trip Distribution by a Hybrid Procedure of Global Optimization

Edson Tadeu Bez,[1] Mirian Buss Gonçalves,[2] and José Eduardo Souza de Cursi[3]

[1] *Universidade do Vale do Itajaí - UNIVALI, São José, Brasil,* edsonbez@univali.br

[2] *Universidade Federal de Santa Catarina - UFSC, Florianópolis, Brasil,* mirian@mtm.ufsc.br

[3] *Laboratoire de Mécanique de Rouen (INSA - Rouen), Saint-Etienne du Rouvray, France,* souza@insa-rouen.fr

Abstract In this work, we present a hybrid global optimization method of evolutionary type. The procedure uses random perturbations of descent methods in the mutation step and generates the initial population by using a Representation Formula of the global optimum. The method is applied to a relevant inverse problem: the calibration of a gravity–opportunity model for trip distribution in transportation theory.

Keywords: Global Optimization, Hybrid Methods, Gravity–Opportunity Model

1. Introduction

Transportation planning is a field where the calibration of models is often used. In typical situations, the mathematical model to be used contains parameters to be determined from empiric data by identification procedures. Generally, these procedures lead to non convex optimization problems and robust numerical methods are needed in the calibration process. We consider in this work the calibration of a gravity–opportunity model which is derived by applying the Maximum Likelihood Principle to the experimental data, such as, for instance, measurements of O-D (origin-destination) trips (see, for instance, [5]). Such a model leads to function exhibiting the typical behaviour shown in Figure 1. Previous works [3, 6, 7] properly addressed the numerical difficulties in the calibration procedure, which are essentially connected, on the one hand, to the nonconvexity and, on the other hand, to the wide sensibility of the objective function with respect to the parameters to be determined: large desequilibrated gradients are involved. We present here a hybrid global optimization procedure for solving the problem. The method has been tested on classical functions such as Rastringin's one (Figure 2) [4] and it is applied here to the calibration of a gravity–opportunity model from O-D sources [5]. In this field, the standard procedures are difficulty to use, request high computational effort and uses optimization parameters specially defined for a given data set. Among our objectives, we look for the definition of a calibration procedure less dependent on the data set and saving computational cost.

2. The Trip Distribution Model

By limitation of the room, we do not give more detailed derivation of the model: the reader interested in this aspect is invited to refer to [5]. Let us denote by T_{ij} the number of trips going

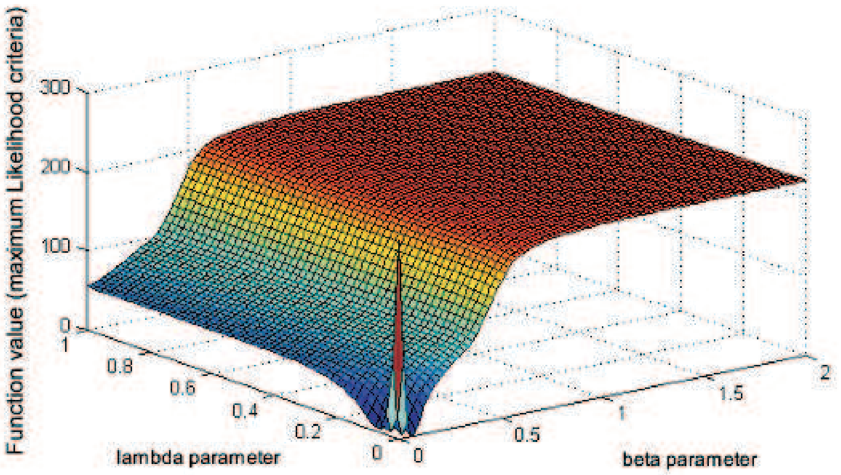


Figure 1. Function defined by the Maximum Likelihood criteria ([2]).

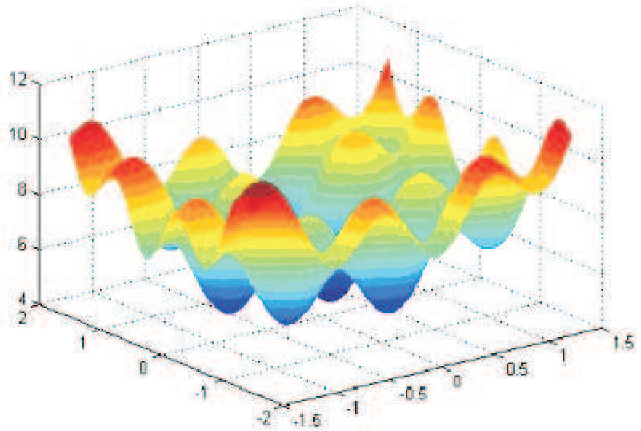


Figure 2. Rastringin's test function.

from origin number i to destination number j, O_i the number of departures from origin number i, D_j the number of arrivals at destination j, W_{ij} a quantity measuring the opportunities found between origin i and destination j, c_{ij} a quantity connected to the cost of the trip from origin i to destination j. The gravity – opportunity model doubly restricted reads as

$$T_{ij} = A_i O_i B_j D_j e^{-(\lambda W_{ij} + \beta c_{ij})} \quad , i, j = 1, 2, ..., n. \tag{1}$$

Parameters A_i, B_j, λ, β are usually unknown and must be determined. Usually, the main parameters are λ and β, since the coefficients A_i and B_j must verify nonlinear relations connected to the equilibrium of the fluxes between destinations and origins. For given λ and β, A_i and B_j may be determined by solving the nonlinear equations establishing the equilibrium of the fluxes. One of the most popular methods for this determination is Furness balancing method, which will not be developed here.

Let us assume that a set of measurements T^*_{ij} is given. Then, we have (see, for instance, [5]):

$$A_i = (\sum_j B_j D_j e^{-\lambda W_{ij}} e^{-\beta c_{ij}})^{-1} \tag{2}$$

$$B_j = (\sum_i A_i O_i e^{-\lambda W_{ij}} e^{-\beta c_{ij}})^{-1} \tag{3}$$

$$\sum_{i\,j} T^*_{ij} c_{ij} = \sum_{i\,j} T_{ij} c_{ij} \tag{4}$$

$$\sum_{i\,j} T^*_{ij} w_{ij} = \sum_{i\,j} T_{ij} w_{ij}. \tag{5}$$

These relations may be derived from the principle of the Maximum Likelihood applied to (1). The parameters β and λ are determined into a way that the results reproduce the average cost of the observed trips and the average number observed in the intervenient opportunities per trip: (β, λ) minimizes the function

$$f(\beta, \lambda) = \left(\overline{c} - \sum_{ij} \frac{T_{ij}(\beta, \lambda)}{T} \cdot c_{ij}\right)^2 + \left(\overline{w} - \sum_{ij} \frac{T_{ij}(\beta, \lambda)}{T} \cdot w_{ij}\right)^2 \tag{6}$$

where

$$\overline{c} = \frac{\sum\limits_{ij} T^*_{ij} c_{ij}}{T^*} \quad and \quad \overline{w} = \frac{\sum\limits_{ij} T^*_{ij} w_{ij}}{T^*}. \tag{7}$$

If Equations 4 – 5 are satisfied $f(\beta, \lambda) = 0$.

3. A Numerical Method

[7] has introduced an evolutionary version of an random perturbation algorithm based on the gradient method. We present here a modification of these method, improved by the introduction of a Representation Formula established by [9]. The Representation is used in order to generate the initial population. The experiments have established improvements in the robustness and speed of convergence. A complete set of experiments with classical test functions is performed in [4].

The Representation Formula reads as follows: let us consider a regular function $f : \Re^n \rightarrow \Re$, defined on a closed bounded not empty set $S \subset \Re^n, \ S \neq \phi$. Assume that P is a probability on S having a strictly positive regular density. Then, we have:

$$x^* = \lim_{\lambda \to +\infty} \frac{E(xg(\lambda, f(x)))}{E(g(\lambda, f(x)))}, \tag{8}$$

where $\lambda \in \Re_+$, g is a function conveniently chosen and $E(\bullet)$ denotes the mean. Equation 8 may be interpreted as a weighted mean of x on S. The weights are connected to the value of $f(x)$ and, for g having suitable decreasing properties, the weights are smaller for points corresponding to higher values of the objective function. At the limit, the weights concentrate on the points corresponding to the global optimum. A possible choice for g, suggested by M. Pincus (see, for instance, [9]) is:

$$g(\lambda, f(x)) = e^{-\lambda f(x)}. \tag{9}$$

In practice, we must choose a value of λ large enough, we generate a finite sample $(x_1, ..., x_{ns})$ of elements of S, according to the distribution P and we approximate

$$x^* \approx x^*_{ns} = \frac{\sum_{i=1}^{ns} x_i g(\lambda, f(x_i))}{\sum_{i=1}^{ns} g(\lambda, f(x_i))}. \tag{10}$$

Analogously to [7], we shall use an evolutionary algorithm where the mutation step is performed by using random perturbations of a descent method which iterates as $x_{n+1} = Q_n(x_n)$: the perturbed method iterates as $x_{n+1} = Q_n(x_n) + P_n$, where P_n is a convenient random variable such as, for instance,

$$P_n = \frac{a}{\sqrt{log(n+b)}} Z \quad ; \quad Z\tilde{N}(0, \sigma Id). \tag{11}$$

Equation 10 will be used in order to generate the initial population. The procedure reads as follows ([4]):

1. Let be given $H > 0$; three non negative integers *NP*, *NF* and *NR*;
2. We generate the initial population $S_0 = \left\{x_0^1, x_0^2, ..., x_0^{NP}\right\}$, by using (Equation 10).
3. At step n, S_{n+1} is derived from S_n as follows:
   - *Crossover*: Let

     $$F_n = \left\{ y_n^i = \alpha_n^i x_n^j + \beta_n^i x_n^k + \gamma_n^i : x_n^j, x_n^k \in S_n : \; i = 1, ..., NF \right\},$$

     where α^i_n, β^i_n, γ^i_n are random values from the uniform distribution on $[-H, H]$ and *j* and *k* are randomly chosen;
     F_n contains $NF = (2 * NP)$ elements;
     We note $B_n = S_n \cup F_n$. B_n contains $NB = (NP + NF)$ elements;
   - *Mutation*: Let

     $$M_n = \{xp_n^j = Arg \min f(Q_n(x_n^j) + P_n^{j,i}) : x_n^j \in B_n : \; i = 0, ..., NR; j = 1, ..., NB\},$$

     M_n contains $NM = (2 * NB)$ elements; $P_n^{j,0} = 0$ e $P_n^{j,1}$, ..., $P_n^{j,NR}$ is a sample of *NR* values from P_n;
   - *Selection*: The elements of $A_n = B_n \cup M_n = S_n \cup F_n \cup M_n$ are increasing ordered according to the value of the objective function. The population S_{n+1} is formed by the first *NP* elements of A_n, *i. e.*, the best NP elements of S_n.

In our experiments, the deterministic descent method Q_n is the gradient method with optimal step (steepest descent. See, for instance, [9]). The maximum step is $t_{\max} = \frac{\alpha}{\|\nabla f(x_n)\|}$, where $\alpha > 0$ is a fixed parameter. Thus, the step t_n is defined by the relation $f(x_n - t_n \nabla f(x_n)) \leq f(x_n - t\,\nabla f(x_n)), \;\; \forall t \in (0, t_{\max})$.

4. Data Sets

We have tested the method on 4 data sets: three sets of real data and one set of simulated data. The first set of real data comes from the region of Londrina, Brazil,which is divided into 12 traffic zones [1]. The second one corresponds to a region that includes part of the State of Santa Catarina, Brazil, divided into 77 zones [5]; The third ones is a subset of the previous region, corresponding to 44 zones of particular interest [5]. The set of simulated data is defined on a region divided into 30 zones [8].

5. Results

All the experiments performed use the following set of parameters: $NP = 2$ (population of two elements); $ns = 100$ (size of the sample for application of the Representation Formula), generated from $N(0, 0.5)$; $NR = 5$ (Number of random perturbations of the deterministic descent method), with $a = 0.1$, $b = 1$ and Z generated from $N(0, 0.2)$; initial step size $\alpha = 0.7$. The determination of the optimal step has been performed by analysis the values of the objective function on a grid of 21 points uniformly distributed on the interval $(0, t_{max})$. Furness iterations have been stopped when either numbers of iterations has reached 100 or the value of the objective function has attained 10^{-3}: $f(\beta,\lambda)$ <10^{-3}. For each situation considered, we have performed 100 runs. In Table 1, we show the mean value of the objective function attained, and we underline the effect of the use of Equation 8: we observe a significant improvement.

Table 1. Mean of the final value of the objective function (Equation 6) for 100 runs.

	Gonçalves Data (1992) 77 zones	*Gonçalves Data (1992) 44 zones*	*Almeida Data (1999) 12 zones*	*Kühlkamp Data (2003) 30 zones*
with/ Equation 8	70745.27	11178.77	257.27	52648.36
without/ Equation 8	22.94	7.96	1.13	43.32

In Table 2, we underline the number of iterations.

Table 2. Mean of the final value of the objective function (Equation 6) for 100 runs.

	Gonçalves Data (1992) 77 zones	*Gonçalves Data (1992) 44 zones*	*Almeida Data (1999) 12 zones*	*Kühlkamp Data (2003) 30 zones*
with/ Equation 8	8800	6720	1960	3720
without/ Equation 8	5960	2120	1160	2440

In Figure 3, we compare the performance of the numerical procedure with or without Equation 8, and also with or without the optimal step procedure.

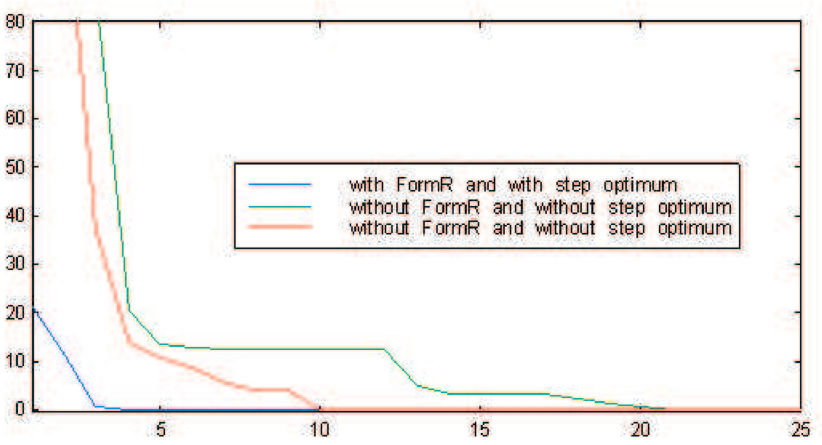


Figure 3. Influence of Equation 8 and of use of the optimum step in the gradient method - Set of data with 44 zones.

6. Concluding Remarks

A stochastic global optimization procedure of evolutionary type has been presented. The procedure uses random perturbations of descent methods in the mutation step and generates the initial population by using a Representation Formula of the global optimum. The Representation Formula, applied in the bettering of the generation of the initial population, placed the points closer to the solution, causing an increase of convergence speed, and consequently, a significant reduction in the number of evaluations of the function.

The method was applied to a relevant inverse problem: the calibration of a gravity-opportunity model for trip distribution in transportation theory. The convergence in the calibration process using a parameter set as default was verified.

The procedure is viewed as an important tool for estimation of parameters of spatial-interaction models.

References

[1] Almeida, L. M. W. *Desenvolvimento de uma metodologia para Análise Locacional de Sistemas Educacionais Usando Modelos de Interação Espacial e Indicadores de Acessibilidade*. Florianópolis, 1999. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina.

[2] Bez, E. T. *Um estudo sobre os procedimentos de calibração de alguns modelos de Distribuição de Viagens*. Florianópolis, 2000. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina.

[3] Bez, E. T. and Gonçalves, M. B. (2001) Uma análise do comportamento de algumas medidas de ajuste usadas na determinação dos parâmetros de modelos de distribuição de viagens. *TEMA*, São Carlos, v.3, n.1, 51-60.

[4] Bez, E. T., Souza de Cursi, J. E. and Gonçalves, M. B. (2003) Procedimento de Representação de Soluções em Otimização Global.Tendências em Matemática Aplicada e Computacional, São José do Rio Preto. *Resumos das Comunicações do XXVI CNMAC*. Rio de Janeiro : SBMAC, v. 1, p. 545-545.

[5] Gonçalves, M. B. *Desenvolvimento e Teste de um Novo Modelo Gravitacional – de Oportunidades de Distribuição de Viagens*. Florianópolis, 1992. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina.

[6] Gonçalves, M. B. and Souza de Cursi, J. E. (1997) Métodos Robustos para a Calibração de Modelos de Interação Espacial em Transportes. Associação Nacional de Pesquisa e Ensino em Transportes, 11., 1997, Rio de Janeiro. *Anais...* Rio de Janeiro: v. 2, 303-313.

[7] Gonçalves, M. B. and Souza de Cursi, J. E. (2001) Parameter Estimation in a Trip Distribution Model by Random Perturbation of a Descent Method**.** *Transportation Research B*, v. 35, 137-161.

[8] Kühlkamp, N. *Modelo de oportunidades intervenientes, de distribuição de viagens, com ponderação das posições espaciais relativas das oportunidades*. Florianópolis, 2003. Tese (Doutorado em Engenharia Civil) - Programa de Pós-Graduação em Engenharia Civil, Universidade Federal de Santa Catarina.

[9] Souza de Cursi J. E. (2003) Representation and Numerical Determination of the Global Optimizer of a Continuous Function on a Bounded Domain, to appear in *Frontiers in Global Optimization,* (C. A. Floudas and P. M. Pardalos, Eds.), Kluwer Academic Publishers.

On weights estimation in Multiple Criteria Decision Analysis*

R. Blanquero[1], E. Carrizosa[1], E. Conde[1], and F. Messine[2]

[1] *Facultad de Matemáticas, Universidad de Sevilla, Spain*
{rblanquero,ecarrizosa,educon}@us.es

[2] *ENSEEIHT-IRIT, 2 rue C Camichel, 31071 Toulouse Cedex, France*
Frederic.Messine@n7.fr

Abstract Several Multiple-Criteria Decision Making methods require, as starting point, weights measuring the relative importance of the criteria. A common approach to obtain such weights is to derive them from a pairwise comparison matrix A.

There is a vast literature on proposals of mathematical-programming methods to infer weights from A, such as the eigenvector method or the least (logarithmic) squares. Since distinct procedures yield distinct results (weights) we pose the problem of describing the set of weights obtained by "sensible" methods: those which are Pareto-optimal for the nonconvex (vector-) optimization problem of simultaneous minimization of discrepancies.

A characterization of the set of Pareto-optimal solutions is given. Moreover, although the above-mentioned optimization problems may be multimodal, standard Global Optimization methods can come up with a globally optimal vector of weights in reasonable time.

Keywords: Pairwise comparison matrices, weights, Interval Analysis.

1. Introduction

Several strategies have been suggested in the literature to associate with a set $\mathcal{D} = \{d_1, \dots, d_n\}$ of decisions weights $x_1, x_2, \dots, x_n$ reflecting decision-maker's preferences. In the Analytic Hierarchy Process (AHP), [11,12,14,15], an $n \times n$ matrix A,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

is obtained after asking the decision-maker (DM) to quantify the ratio of his/her preferences of one decision over another. In other words, for every pair of decisions d_i, d_j, the term $a_{ij} > 0$ is requested satisfying

$$a_{ij} \approx \frac{x_i}{x_j} \tag{1}$$

The matrix A so obtained must be a *positive reciprocal* matrix, i.e.,

$$a_{ji} = \frac{1}{a_{ij}} > 0 \qquad \text{for all } i, j = 1, 2, \dots, n.$$

*This research has been partially supported by grants BFM2002-04525-C02-02 and BFM2002-11282-E MCYT, Spain

For a given positive reciprocal matrix A, different procedures can be followed in order to obtain weights $x_1, \dots, x_n$ according to (1), see e.g. [1,5,9,10,13]. In particular, Saaty proposes the so-called *Eigenvector method* (EM): x is a column vector satisfying the equation

$$Ax = \lambda_{\max} x,$$

where $\lambda_{\max}$ is the dominant eigenvalue of the positive reciprocal matrix A. See e.g. [12] for further details, and [6] for commercial software with it.

Many other choices have been proposed in the literature to derive x according to (1), mostly given as optimal solutions of optimization problems such as

$$\min_{x \in \mathbb{R}^n_{++}} \gamma \left((|x_i/x_j - a_{ij}|)_{i,j=1}^n \right) \tag{2}$$

or

$$\min_{x \in \mathbb{R}^n_{++}} \gamma \left((|\log(x_i/x_j) - \log(a_{ij})|)_{i,j=1}^n \right) \tag{3}$$

where $\mathbb{R}_{++}$ denotes the set of strictly positive reals, and γ is a monotonic norm in the non-negative orthant $\mathbb{R}^{n \times n}_{+}$, i.e.,

$$\gamma(u) \leq \gamma(v) \quad \forall u, v, \, 0 \leq u \leq v \tag{4}$$

such as any (weighted) ℓ_p norm, $1 \leq p \leq +\infty$ (usually $p = 2$ in the literature). See [3] for a very recent and thorough discussion on different procedures for deriving weights.

It should become evident that different procedures, – (EM) or those derived from (2) or (3)–, although following (1), may yield different weights, and even different ranking of decisions may happen, as already shown e.g. in [13].

In the next section we deal with the problem of inferring weights under a multi-objective point of view, focus our attention in the key property of the solutions of vector-optimization problems, namely *efficiency*. In particular, we explore whether the usual weighting methodologies, provide (or do not) solutions with this desirable property.

In the last section we consider the generation of weights from (2), a problem that may be multimodal. We show that standard tools of Global Optimization based on Interval Analysis can be applied to its resolution with reasonable computational times.

2. Inferring weights: A multi-objective approach

The problem of deriving weights from a pairwise comparison matrix naturally leads to the Nonconvex Vector-Optimization problem

$$\min_{x \in \mathbb{R}^n_{++}} (|\frac{x_i}{x_j} - a_{ij}|)_{i \neq j} \tag{X}$$

We recall the reader, e.g. [2], that, given an optimization problem (P),

$$\min_{x \in S} \left(f_1(x), \dots, f_k(x) \right),$$

$y \in S$ is said to *dominate* $x \in S$ if $f_i(y) \leq f_i(x)$ for all $i = 1, \dots, k$, with $f_i(y) < f_i(x)$ for some i. Moreover, $x \in S$ is said to be *efficient* for (P) if no $y \in S$ dominates x, and x is said to be locally efficient for (P) if there exists a neighborhood V of x in S such that no $y \in V$ dominates x.

We provide a test based on Linear Programming that let us to check if any vector $x \in \mathbb{R}^n_{++}$ is efficient for (X), which is completed with a full description of the set of locally efficient and efficient solutions for this problem. Using these results, we prove that the geometric-mean method for deriving weights, i.e.

$$x_i = \left(\prod_{j=1}^{n} a_{ij} \right)^{\frac{1}{n}}, \qquad i = 1, \ldots, n \tag{5}$$

provides efficient solutions for (X).

A geometric characterization of efficiency is also been obtained, based on the graph introduced next.

Definition 1. *Given $y \in \mathbb{R}^N$, let $G(y)$ be the digraph $G(y) := (\{1, 2, \ldots, N\}, E(y))$,*

$$(i,j) \in E(y) \textit{ iff } i \neq j \textit{ and } y_i - y_j \geq \log(a_{ij})$$

The above-mentioned geometric characterization is provided in the following result (we recall that a directed graph is said to be strongly connected if for all pairs of nodes $(i, j), i \neq j$, there exist directed paths from i to j and from j to i).

Theorem 2. *Vector $x \in \mathbb{R}^n_{++}$ is efficient for (X) if and only if $G(\log(x))$ is strongly connected.*

A similar characterization can be obtained for weakly efficient solutions. We recall that $x^* \in \mathbb{R}^n_{++}$ is said to be weakly efficient for (X) if and only if no $x \in \mathbb{R}^n_{++}$ exists with

$$\left| \frac{x_i}{x_j} - a_{ij} \right| < \left| \frac{x_i^*}{x_j^*} - a_{ij} \right| \qquad \text{for all } i, j,\ i \neq j$$

Theorem 3. *Vector $x \in \mathbb{R}^n_{++}$ is weakly efficient for (X) if and only if $G(\log(x))$ contains at least one cycle.*

Using Theorem 2 we show, by means of an example, that the Eigenvector Method may provide non-efficient solutions. This makes preferable, under the viewpoint here considered, the solution provided by the geometric-mean method or any other derived from (2), which are also efficient.

3. Inferring weights: A global optimization approach

The resolution of (2) constitutes an alternative procedure to the Eigenvector Method of Saaty for deriving the priority vector x. In spite of the fact that this problem may be multimodal, local-search methods seem to be the only proposal so far, the exception being the recent work [7], where different Global-Optimization strategies for solving (2) for the Euclidean norm γ are analyzed.

From a practical viewpoint, it has been observed that it is not so easy to provide precise values for the scalar a_{ij}, which are thus replaced by intervals $A_{ij} = [a^L_{ij}, a^U_{ij}]$, see e.g. [4,8,16] and the references therein. In this case, the aspiration $x_i/x_j = a_{ij}$ is replaced by the more general one

$$\frac{x_i}{x_j} \in A_{ij},\ i,j = 1, 2, \ldots, n, \tag{6}$$

and (2) is replaced by the problem of finding strictly positive weights x_j minimizing a norm γ of the distances between the ratios x_i/x_j and the corresponding intervals A_{ij},

$$\min_{x \in \mathbb{R}^n_{++}} f(x) := \gamma\left(\left(\varepsilon\left(x_i/x_j, A_{ij}\right)\right)_{i,j=1}^{n} \right) \tag{7}$$

with $\varepsilon(s,[a^L,a^U])$ defined as the distance between s and the closest point in $[a^L,a^U]$,

$$\begin{aligned}\varepsilon(s,[a^L,a^U]) &= \begin{cases} 0, & \text{if } s \in [a^L,a^U] \\ a^L - s, & \text{if } s \leq a^L \\ s - a^U, & \text{if } s \geq a^U \end{cases} & (8) \\ &= \max\left\{0, \left|s - \frac{a^U + a^L}{2}\right| - \frac{a^U - a^L}{2}\right\} & (9)\end{aligned}$$

Obviously (7) coincides with (2) in what we call the degenerate case in which all intervals A_{ij} are degenerate, $A_{ij} = \{a_{ij}\}$.

In this context, theoretical results are given without needing the definition of a norm. We show that well-known strategies of Deterministic Global Optimization, namely, Branch and Bound algorithms with bounds based on Interval Analysis, provide in reasonable time globally optimal solutions to problem (7), and thus, as particular case, to model (2), for different choices of γ. The standard interval Branch and Bound algorithm has been modified here in order to determine the accuracy of the so-obtained optimal solution and, using this, the numerical experiments show perfectly that some problems can efficiently be solved even if the size is rather important.

References

[1] Bryson, N. (1995). "A goal programming method for generating priority vectors". *Journal of the Operational Research Society,* 46:641–648.

[2] Chankong, V., and Y. Haimes (1983). *Multiobjective Decision Making*, North–Holland, New York.

[3] Choo, E.U. and Wedley, W.C. (2004). "A common framework for deriving preference values from pairwise comparison matrices". *Computers & Operations Research*, 31:893–908.

[4] Conde, E. (2002). "Mean utility in the assurance region model". *European Journal of Operational Research* 140:93–103.

[5] Cook, W.D., and Kress, M. (1988). "Deriving weights from pairwise comparison ratio matrices: An axiomatic approach". *European Journal of Operational Research*, 37:320–355.

[6] *Expert Choice*, at `http://www.expertchoice.com`

[7] Flp, J. "Global Optimization methods for approximation by consistent matrices". Presented at the ISMP 2003, Copenhaguen.

[8] Haines, L.M. (1998). "A statistical approach to the analytic hierarchy process with interval judgements. (I). Distributions on feasible regions". *Journal of the Operational Research Society* 110:112–125.

[9] Lootsma, F.A. (1996). "A model for the relative importance of the criteria in the Multiplicative AHP and SMART". *European Journal Of Operational Research*, 94:467–476.

[10] Ramanathan, R.(1997). "A note on the use of goal programming for the Multiplicative AHP". *Journal of Multi-criteria Decision Analysis*, 6:296–307.

[11] Saaty, T.L. (1977). "A scaling method for priorities in hierarchical structures". *Journal of Mathematical Psychology* 15:234–281.

[12] Saaty, T.L. (1980). *Multicriteria Decision Making: The Analytic Hierarchy Process.* McGraw-Hill, New York.

[13] Saaty, T.L. (1990). "Eigenvector and logaritmic least squares". *European Journal of Operational Research* 48:156–160.

[14] Saaty, T.L. (1994). *Fundamentals of Decision Making.* RSW Publications, Pittsburg, PA.

[15] Saaty, T.L. (1994). "How to make a decision: The Analytic Hierarchy Process". *Interfaces* 24:19–43.

[16] Salo, A. and Hamalainen, R.P. (1995). "Preference programming through approximate ratio comparisons". *European Journal of Operational Research*, 82:458–475.

Least Squares approximation of pairwise comparison matrices[*]

Sándor Bozóki

presently a PhD student at Corvinus University of Budapest, pursuing research at the Laboratory and the Department of Operations Research and Decision Systems, Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), bozoki@oplab.sztaki.hu

Abstract One of the most important steps in solving Multi-Attribute Decision Making (MADM) problems is to derive the weights of importance of the attributes. The decision maker is requested to compare the importance for each pair of attributes. The result expressed in numbers is written in a pairwise comparison matrix. The aim is to determine a weight vector $\mathbf{w} = (w_1, w_2, , w_n)$, which reflects the preferences of the decision maker, in the positive orthant of the n-dimensional Euclidean space. We examine a distance minimizing method, the Least Squares Method (LSM). The LSM objective function is nonlinear and, usually, non-convex, thus its minima is not unique, in general. It is shown that the LSM-minimization problem can be transformed into a multivariate polynomial system. We consider the resultant and the generalized resultant methods, which can be applied in the case of small-size matrices, and the homotopy continuation proposed by Tien-Yien Li and Tangan Gao. Numerical experience show that the homotopy method finds all the roots of polynomial systems, hence all the minima of the LSM objective functions. At present the maximum size of matrices regarding which the LSM approximation problem can be solved by using the homotopy method is 8×8. We show that LSM works even if some elements are missing from the pairwise comparison matrix. The paper ends with few numerical examples.

Keywords: Pairwise comparison matrix, Least Squares Method, Polynomial systems.

1. Introduction

One of the most studied methodology in Multi-Attribute Decision Making is the Analytic Hierarchy Process developed by Thomas L. Saaty [21]. Using AHP, difficult decision problems can be broken into smaller parts by the hierarchical criterion-tree, one level of the tree can be handled by pairwise comparison matrices. The idea of using pairwise comparison matrices is that decision makers may not tell us the explicit weights of the criteria or the cardinal preferences of the alternatives but they can make pairwise comparisons.

A pairwise comparison matrix $\mathbf{A} = [a_{ij}]_{i,j=1..n}$ is defined as

$$\mathbf{A} = \begin{pmatrix} 1 & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & 1 & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & 1 & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}_{+},$$

[*] This research was supported by the Hungarian National Research Foundation, Grant No. OTKA-T043241.

where for any $i, j = 1, \ldots, n$,

$$\begin{aligned} a_{ij} &> 0, \\ a_{ij} &= \frac{1}{a_{ji}}. \end{aligned}$$

The matrix element a_{ij} expresses the relative importance or preference of i-th object compared to j-th object given by the decision maker ($i, j = 1, 2, \ldots, n$). For example, the first object is a_{12} times more important/preferred than the second one.

A pairwise comparison matrix $\mathbf{A} = [a_{ij}]_{i,j=1..n}$ is called consistent, if it satisfies the following properties for all indices $i, j, k = 1, \ldots, n$:

$$\begin{aligned} a_{ij} &= \frac{1}{a_{ji}}, \\ a_{ij}a_{jk} &= a_{ik}. \end{aligned}$$

In practical decision problems, pairwise comparison matrices given by the decision maker are not consistent. Based on the elements of the matrix, we want to find a weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T \in \mathbb{R}^n_+$ representing the priorities of the objects where $\mathbb{R}^n_+$ is the positive orthant. The Eigenvector Method [21] and some distance minimizing methods such as the Least Squares Method [7, 18], Logarithmic Least Squares Method [1, 9–11], Weighted Least Squares Method [2,7], Chi Squares Method [18] and Logarithmic Least Absolute Values Method [8,17], Singular Value Decomposition [15] are of the tools for computing the priorities of the alternatives.

After some comparative analyses [6,9,22,23] Golany and Kress [16] have compared most of the scaling methods above by seven criteria and concluded that every method has advantages and weaknesses, none of them is prime.
Since LSM problem has not been solved fully, comparisons to other methods are restricted to a few specific examples.

The aim of the paper is to present a method for solving LSM for matrices up to the size 8×8 in order to ground for further research of comparisons to other methods and examining its real life application possibilities.

In the paper we study the Least Squares Method (LSM) which is a minimization problem of the Frobenius norm of $(\mathbf{A} - \mathbf{w}\frac{1}{\mathbf{w}}^T)$, where $\frac{1}{\mathbf{w}}^T$ denotes the row vector $(\frac{1}{w_1}, \frac{1}{w_2}, \ldots, \frac{1}{w_n})$.

2. Least Squares Method (LSM)

The aim is to solve the following optimization problem for a given matrix $\mathbf{A} = [a_{ij}]_{i,j=1..n}$.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{w_i}{w_j} \right)^2 \tag{1}$$

$$\sum_{i=1}^{n} w_i = 1,$$

$$w_i > 0, \qquad i = 1, 2, \ldots, n.$$

LSM is rather difficult to solve because the objective function is nonlinear and usually nonconvex, moreover, no unique solution exists [13, 18, 19] and the solutions are not easily computable. Farkas, Lancaster and Rózsa [12] applied Newton's method of successive approximation. Their method requires a good initial point to find the solution.

It is shown in the paper that the LSM minimization problem can be transformed into solving a multivariate polynomial system. For a given $n \times n$ pairwise comparison matrix, the number of equations and variables in the corresponding polynomial system is $n - 1$.

3. Polynomial systems

Polynomial systems are not easy to solve in general. The method based on Gröbner bases [5] in Maple works for the 3×3 matrices but runs out of memory as $n > 3$. Bozóki [3] gave a method based on resultants for solving the LSM problem for 3×3 matrices, and an other one with Lewis [4] based on generalized resultants for 4×4 matrices.

Homotopy method is a general technique for solving nonlinear systems. In the paper, I used the code written by Tien-Yien Li and Tangan Gao. Present CPU and memory capacity allow us to solve the LSM problem up to 8×8 matrices.

4. Pairwise comparison matrix with missing elements

If a decision maker has n objects to compare, he/she needs to fill $\frac{n(n-1)}{2}$ elements of the upper triangular submatrix of the $n \times n$ pairwise comparison matrix. The number of comparisons quickly increases by n.

Eigenvector Method needs all the elements of the matrix, otherwise the eigenvalues and eigenvectors can not be computed.

The Least Squares approximation of a pairwise comparison matrix has the advantage that it can be used in cases of missing elements, too. If we do not have a value in the (i, j)-th position of the matrix, we simply skip the corresponding term $\left(a_{ij} - \frac{w_i}{w_j}\right)^2$ from the objective function (1).

5. Summary

A method for solving the LSM problem for pairwise comparison matrices is given in the paper. The LSM optimization problem is transformed to a polynomial system which can be solved by resultant method, generalized resultant method using the computer algebra system Fermat implemented by Robert H. Lewis, or homotopy method, implemented by Tangan Gao and Tien-Yien Li. One of the advantages of LSM weighting method is that it can be used even if the pairwise comparison matrix is not completely filled.

Acknowledgments

The author wishes to thank Robert H. Lewis and Tangan Gao for making their algorithms available.

References

[1] Barzilai, J., Cook, W.D, Golany, B. (1987): Consistent weights for judgements matrices of the relative importance of alternatives, *Operations Research Letters,* **6,** pp. 131-134.

[2] Blankmeyer, E., (1987): Approaches to consistency adjustments, *Journal of Optimization Theory and Applications,* **54,** pp. 479-488.

[3] Bozóki, S. (2003): A method for solving LSM problems of small size in the AHP, *Central European Journal of Operations Research,* **11** pp. 17-33.

[4] Bozóki, S., Lewis, R. (2005): Solving the Least Squares Method problem for in the AHP 3×3 and 4×4 matrices, *Central European Journal of Operations Research,* (accepted).

[5] Buchberger, B. (1965): An Algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal (in German) Doctoral Dissertation Math. Inst. University of Innsbruck, Austria.

[6] Budescu, D.V., Zwick, R., Rapoport, A. (1986): A comparison of the Eigenvector Method and the Geometric Mean procedure for ratio scaling, *Applied Psychological Measurement*, **10** pp. 69-78.

[7] Chu, A.T.W., Kalaba, R.E., Spingarn, K. (1979): A comparison of two methods for determining the weight belonging to fuzzy sets, *Journal of Optimization Theory and Applications* **4,** pp. 531-538.

[8] Cook, W.D., Kress, M. (1988): Deriving weights from pairwise comparison ratio matrices: An ax0iomatic approach, *European Journal of Operations Research,* **37** pp. 355-362.

[9] Crawford, G., Williams, C. (1985): A note on the analysis of subjective judgment matrices, *Journal of Mathematical Psychology* **29,** pp. 387-405.

[10] De Jong, P. (1984): A statistical approach to Saaty's scaling methods for priorities, *Journal of Mathematical Psychology* **28,** pp. 467-478.

[11] DeGraan, J.G. (1980): Extensions of the multiple criteria analysis method of T.L. Saaty (Technical Report m.f.a. 80-3) Leischendam, the Netherlands: National Institute for Water Supply. Presented at EURO IV, Cambridge, England, July 22-25.

[12] Farkas, A., Lancaster, P., Rózsa, P. (2003): Consistency adjustment for pairwise comparison matrices, *Numerical Linear Algebra with Applications,* **10,** pp. 689-700.

[13] Farkas, A., Rózsa, P. (2004): On the Non-Uniqueness of the Solution to the Least-Squares Optimization of Pairwsie Comparison Matrices, *Acta Polytechnica Hungarica, Journal of Applied Sciences at Budapest Polytechnic Hungary,* **1**, pp. 1-20.

[14] Gao, T., Li, T.Y., Wang, X. (1999): Finding isolated zeros of polynomial systems in $\mathbb{C}^n$ with stable mixed volumes, *J. Symbolic Comput.*, **28**, pp. 187-211.

[15] Gass, S.I., Rapcsák, T. (2004): Singular value decomposition in AHP, *European Journal of Operations Research* **154** pp. 573-584.

[16] Golany, B., Kress, M. (1993): A multicriteria evaluation of methods for obtaining weights from ratio-scale matrices, *European Journal of Operations Research,* **69** pp. 210-220.

[17] Hashimoto, A. (1994): A note on deriving weights from pairwise comparison ratio matrices, *European Journal of Operations Research,* **73** pp. 144-149.

[18] Jensen, R.E. (1983): Comparison of Eigenvector, Least squares, Chi square and Logarithmic least square methods of scaling a reciprocal matrix, *Working Paper 153 http://www.trinity.edu/rjensen/127wp/127wp.htm*

[19] Jensen, R.E. (1984): An Alternative Scaling Method for Priorities in Hierarchical Structures, *Journal of Mathematical Psychology* **28,** pp. 317-332.

[20] Li, T.Y. (1997): Numerical solution of multivariate polynomial systems by homotopy continuation methods, *Acta Numerica,* **6** pp. 399-436.

[21] Saaty, T.L. (1980): The analytic hierarchy process, *McGraw-Hill,* New York.

[22] Saaty, T.L., Vargas, L.G. (1984): Comparison of eigenvalues, logarithmic least squares and least squares methods in estimating ratios, *Mathematical Modeling,* **5** pp. 309-324.

[23] Zahedi, F. (1986): A simulation study of estimation methods in the Analytic Hierarchy Process, *Socio-Economic Planning Sciences,* **20** pp. 347-354.

Locating Competitive Facilities via a VNS Algorithm*

Emilio Carrizosa[1], José Gordillo[1], and Dolores R. Santos-Peñate[2]

[1] *Universidad de Sevilla, Spain*
{ecarrizosa,jgordillo}@us.es

[2] *Universidad de Las Palmas de Gran Canaria, Spain*
drsantos@dmc.ulpgc.es

Abstract We consider the problem of locating in the time period $[0, T]$ q facilities in a market in which competing firms already operate with p facilities. Locations and the entering times maximizing profits are sought.

Assuming that demands and costs are time-dependent, optimality conditions are obtained, and different demand patterns are analyzed. The problem is posed as a mixed integer nonlinear problem, heuristically solved via a VNS algorithm.

Keywords: Competitive location, Variable Neighborhood Search, Dynamic Demand, Covering Models, Maximal Profit.

1. Introduction

The problem of locating facilities in a competitive environment has been addressed, both at the modelling and computational level, in a number of papers in the field of Operations Research and Management Science, see e.g. [2, 3, 5–9, 12, 14] and the references therein.

The simplest models accommodating competition are those related with *covering*: a firm is planning to locate a series of facilities to compete against a set of already operating facilities, in order to maximize its profit, usually equivalent to maximizing the market share. Different attraction rules may be (and have been) considered to model consumer behavior, and thus to evaluate market share. For instance, with the *binary* attraction rule, one assumes that consumers demand is fully captured by the closest facility, or more generally by the most attractive facility, where attractiveness is measured by a function decreasing in distance and price, e.g. [1, 10, 11, 13].

Serious attempts have been made to model different metric spaces (a discrete set, a transportation network or the plane), or different attraction rules (e.g. the above-mentioned binary rule, as well as rules relying upon the assumption that consumer demand is split into the different facilities, each capturing a fraction of the demand, this demand being decreasing in distance, ...). However, most models assume that consumer demand remains constant through the full planning horizon, which may be a rather unrealistic assumption for new goods or high seasonality products. See e.g. [4, 15] for facility location models which do accommodate time-dependent demand.

In this talk we introduce a covering model for locating facilities in a competitive environment in which demand is time-dependent. This implies that, as e.g. in [4], not only the facility

*Partially supported by projects BFM2002-04525, Ministerio de Ciencia y Tecnología, Spain, and FQM-329, Junta de Andalucía, Spain

sites, but also the times at which facilities become operative, are decision variables, to be chosen to optimize a certain performance measure: the total profit in the planning period.

2. The model and its formulation

We consider a market where demand is concentrated at a finite set of points $V = \{v_i\}_{i=1}^n$. Firm A wants to enter into the market by locating at most q facilities within the set $F = \{f_j\}_{j=1}^m$ of candidate sites. Location is assumed to be sequential, in the sense that there exists a time interval $[0, T]$, $0 < T < +\infty$, within which the facilities will start to be operating. In other words, both the sites for the facilities and the times at which they will be located must be determined.

Prior to entry of firm A, a series E of facilities (from firms different to A) are already operating in the market. We also assume that, once A opens a facility at time t, it will remain active in the whole interval $[t, T]$. Moreover, for simplicity we assume that the competing firms will not open new facilities, so the set of competing facilities E is kept constant in $[0, T]$.

Demand is assumed to be inelastic, and consumer preferences are modelled via a binary rule: for each consumer $v \in V$, there exists a threshold value $\underline{d}_v$, satisfying that, if, at time $t \in [0, T]$, firm A has a facility open at some $f \in F$ such that the travel distance $d(v, f)$ from consumer v to facility at f is smaller than such threshold value $\underline{d}_v$, then the demand of v at instant t will be fully captured by firm A. Else, such demand will be fully captured by some of the existing facilities of E, and will be lost for A. In other words, if, for each $v \in V$ we denote by N_v the set of candidate sites for A which *cover* v, i.e., which will capture the demand from v, we have that

$$N_v = \{f \in F : d(v, f) < \underline{d}_v\}.$$

In any given infinitesimal time interval $[t, t + \Delta t]$, demand of consumer $v \in V$ is of the form $\omega_v(t)\Delta t$, where $\omega_v(t)$ is called hereafter demand function.

The net profit margin at time t per unit of revenue is $\rho(t)$, and thus the total incomes generated by v within the infinitesimal time interval $[t, t + \Delta t]$ are given by $\rho(t)\omega_v(t)\Delta t$, if some facility from A covering v is operating at t, and zero otherwise.

Operating costs of a facility at $f \in F$ within the infinitesimal time interval $[t, t + \Delta t]$ have the form $c_f(t)\Delta t$.

Hereafter we assume that functions ω_v, ρ and c_f are continuous on the interval $[0, T]$.

We seek the sites and opening times for a set of at most q facilities for firm A in such a way that the *total profit*, i.e. total incomes minus operating costs within $[0, T]$, are maximized.

To express this as a mathematical program, we will first consider the much easier case in which the opening times are fixed, thus yielding a dynamic location problem, similar to those addressed e.g. by [16,17], and later these will also be considered to be decision variables.

Suppose then that facilities are scheduled to start operating at *fixed* instants $\tau_1, \ldots, \tau_r$, with

$$0 = \tau_0 \le \tau_1 \le \ldots \le \tau_r \le \tau_{r+1} = T,$$

and $0 \le r \le q$.

For $f \in F$, $v \in V$ and k with $1 \le k \le r$, define the variables

$$y_f^k = \begin{cases} 1 & \text{if a facility from } A \text{ located at } f \text{ is operating in the interval } [\tau_k, T] \\ 0 & \text{otherwise} \end{cases}$$

$$x_v^k = \begin{cases} 1 & \text{if } v \text{ is covered by a facility from } A \text{ in the interval } [\tau_k, T] \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, once the values of the variables y_f^k are fixed, those for the variables x_v^k become fixed. However, in order to come up with a linear program, these are also be considered to be decision variables.

With this notation, for entry instants $\tau_1, \dots, \tau_r$ fixed, the covering location problem to be solved is the following linear integer program

$$\begin{aligned} \Pi_r(\tau_1, \dots, \tau_r) &= \max_{\mathbf{x},\mathbf{y}} \sum_{k=1}^{r} \{ \sum_{v \in V} x_v^k \int_{\tau_k}^{\tau_{k+1}} \rho(t) w_v(t) dt - \sum_{f \in F} y_f^k \int_{\tau_k}^{\tau_{k+1}} c_f(t) dt \} \\ & \sum_{f \in F} y_f^r \le q \qquad (1) \\ & x_v^k \le \sum_{f \in N_v} y_f^k, \ \ \forall v \in V, \ 1 \le k \le r \qquad (2) \\ & y_f^{k-1} \le y_f^k, \ \ \forall f \in F, \ 2 \le k \le r \qquad (3) \\ & y_f^k, \ x_v^k \in \{0,1\}, \ \forall f \in F, \forall v \in V, \ 1 \le k \le r. \qquad (4) \end{aligned}$$

We briefly discuss the correctness of the formulation. For the objective, within the time interval $[\tau_0, \tau_1]$, no benefit or cost is incurred, since no plants from A are operating. The interval $[\tau_1, \tau_{r+1}]$ is split into the subintervals $[\tau_k, \tau_{k+1}]$, $k = 1, \dots, r$. Within an interval $[\tau_k, \tau_{k+1}]$, the total incomes obtained from consumer v are $x_v^k \int_{\tau_k}^{\tau_{k+1}} \rho(t) w_v(t) dt$, whereas the total operating cost incurred by facility at f is given by $y_f^k \int_{\tau_k}^{\tau_{k+1}} c_f(t) dt$.

Constraint (1) imposes that the number of open facilities from A cannot exceed q. Constraints (2) impose that, if $x_v^k = 1$, i.e., if v is counted as captured by A in time interval $[\tau_k, T]$, then there must exist at least one facility f from A operating within such interval covering v.

With constraints (3) we express that, if plant f is operating within $[\tau_{k-1}, T]$, then it must also be so in $[\tau_k, T]$.

Finally, constraints (4) express the binary character of the variables x_v^k and y_f^k for $v \in V$, $f \in F$ and $1 \le k \le r$.

Since the variables x_v^k and y_f^k are binary and the functions $\rho(t) w_v(t)$ and $c_f(t)$ are continuous, the optimization problem above is well defined, and its optimal value $\Pi_r(\tau_1, \dots, \tau_r)$ is attained.

The continuity of $\rho(t) w_v(t)$ and $c_f(t)$ enables us also to define their primitives,

$$\begin{aligned} g_v(t) &= \int_0^t \rho(s) w_v(s) ds \\ h_f(t) &= \int_0^t c_f(s) ds, \end{aligned}$$

which are differentiable functions.

Moreover, for each k, $1 \le k \le r$, one has

$$\begin{aligned} \int_{\tau_k}^{\tau_{k+1}} \rho(t) \omega_v(t) dt &= g_v(\tau_{k+1}) - g_v(\tau_k) \\ \int_{\tau_k}^{\tau_{k+1}} c_f(t) dt &= h_f(\tau_{k+1}) - h_f(\tau_k). \end{aligned}$$

With this notation, we can express the optimal profit $\Pi_r(\tau_1, \dots, \tau_r)$ for opening times $\tau_1, \dots, \tau_r$ as

$$\Pi_r(\tau_1, \dots, \tau_r) = \max_{(\mathbf{x},\mathbf{y}) \in S_r} \sum_{k=1}^{r} \{ \sum_{v \in V} x_v^k [g_v(\tau_{k+1}) - g_v(\tau_k)] - \sum_{f \in F} y_f^k [h_f(\tau_{k+1})) - h_f(\tau_k)] \}, \qquad (5)$$

where S_r denotes the set of pairs $(\mathbf{x}, \mathbf{y})$ satisfying constraints (1)-(4).

Considering the instant times $\tau_1, \dots, \tau_r$ as decision variables to be optimized yields the optimal planning for locating at most q facilities in *at most* r different instant times. Indeed, for $\tau_1 < \ldots < \tau_r$ the facilities will become operative in exactly r different instants. Allowing different τ_i to coincide collapses the number of different instants considered. In other words, the problem of determining sites for at most q facilities, to become operative in at most r different instants, can be written as

$$\begin{array}{ll} \max & \sum_{k=1}^{r} \Big\{ \sum_{v \in V} x_v^k [g_v(\tau_{k+1}) - g_v(\tau_k)] - \sum_{f \in F} y_f^k [h_f(\tau_{k+1}) - h_f(\tau_k)] \Big\} \\ & (\mathbf{x}, \mathbf{y}) \in S_r \\ & 0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_r \leq T, \end{array} \tag{6}$$

or equivalently as the bilevel problem

$$\begin{array}{ll} \max & \Pi_r(\tau_1, \ldots, \tau_r) \\ & 0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_r \leq T, \end{array} \tag{7}$$

or equivalently as the bilevel problem

$$\begin{array}{lll} \max_{(\mathbf{x},\mathbf{y}) \in S_r} & \max & \sum_{k=1}^{r} \Big\{ \sum_{v \in V} x_v^k [g_v(\tau_{k+1}) - g_v(\tau_k)] - \sum_{f \in F} y_f^k [h_f(\tau_{k+1}) - h_f(\tau_k)] \Big\} \\ & & 0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_r \leq T. \end{array} \tag{8}$$

Since (6), (7) and (8) are simply different writings of the same problem, we will use one or other at our convenience.

Our problem, namely, the determination of optimal locations and opening times for at most q facilities corresponds to (6), (7) or (8) with $r = q$.

General properties concerning the optimal sites and times are derived. These results are strengthened for different particular instances of demand patterns, namely, the cases in which the demand of a given consumer is constant, varies linearly in time, is increasing or is decreasing. A heuristic procedure is proposed to determine the optimal policy, and numerical experiences are provided.

References

[1] Benati, S. and P. Hansen (2002): The maximum capture problem with random utilities: Problem formulation and algorithms, *European Journal of Operational Research* 143, 518-530.

[2] Craig, S., A. Ghosh and S. McLafferty (1984): Models of the retail location process: a review, *Journal of Retailing*, 60, 1, pp. 5-36.

[3] Drezner, T. (1995): Competitive facility location in the plane, in *Facility Location. A survey of applications and methods*, Z. Drezner (Ed.), Springer, Berlin, 285-300.

[4] Drezner Z. and G.O. Wesolowsky (1991): Facility Location When Demand is Time Dependent, *Naval Research Logistics*, 38, 763-777.

[5] Eiselt, H.A. and G. Laporte (1989): Competitive spatial models, *European Journal of Operational Research*, 39, pp.231-242.

[6] Eiselt, H.A., G. Laporte and J-F. Thisse (1993): Competitive location models: A framework and bibliography, *Transportation Science*, 27, nž 1, pp.44-54.

[7] Friesz, T.L., T. Miller and R.L. Tobin (1988): Competitive network facility location models: A survey, *Papers of the Reginal Science Association*, 65, pp. 47-57.

[8] Hakimi, S.L. (1983): On locating new facilities in a competitive environment, *European Journal of Operational Research*, 12, pp. 29-35.

[9] Hakimi, S.L. (1990): Location with spatial interactions: Competitive locations and games, in Mirchandani, P.B. and R.L. Francis (editors), *Discrete Location Theory*, John Wiley & Sons, Nueva York, pp. 439-478.

[10] Hansen, P., Peeters, D. and Thisse, J.-F., (1995): The profit-maximizing Weber Problem, *Location Science* 3, 2, 67-85.

[11] Plastria, F. (1997): Profit maximising single competitive facility location in the plane, *Studies in Locational Analysis*, 11, 115-126.

[12] Plastria, F. (2001): Static competitive facility location: an overview of optimisation approaches, *European Journal of Operational Research*, 129, 461-470.

[13] Plastria, F. and E. Carrizosa (2004): Location and design of a competitive facility for profit maximisation, *Mathematical Programming* 100, 345-353.

[14] Santos-Peñate, D.R. (2004): Competencia espacial en redes. In Pelegrín, B. (editor), *Avances en localización de servicios y sus aplicaciones*, Servicio de Publicaciones de la Universidad de Murcia, pp. 219-248.

[15] Wey, W.-M. and C.-T. Liao (2001): A study of parking facility location when demand is time dependent. Presented at INFORMS Anual Meeting 2001, Maui, Hawaii, USA.

[16] Wesolowsky, G.O. (1973): Dynamic facility location, *Management Science* 19, 1241-1247.

[17] Wesolowsky, G.O. and W.G. Truscott (1975): The multiperiod location-allocation problem with relocation of facilities, *Management Science* 22, 57-65.

Globally optimal prototypes in kNN classification methods*

E. Carrizosa[1], B. Martín-Barragán[1], F. Plastria[2] and D. Romero-Morales[3]

[1] *Universidad de Sevilla, Spain*
{ecarrizosa,belmart}@us.es

[2] *Vrije Universiteit Brussel, Belgium*
Frank.Plastria@vub.ac.be

[3] *University of Oxford, United Kingdom*
dolores.romero-morales@sbs.ox.ac.uk

Abstract The Nearest Neighbor classifier has shown to be a powerful tool for multiclass classification. In order to alleviate its main drawbacks (high storage requirements and time-consuming queries), a series of variants, such as the Condensed or the Reduced Nearest Neighbor, have been suggested in the last four decades.

In this note we explore both theoretical properties and empirical behavior of another such variant, in which the Nearest Neighbor rule is applied after selecting a set of so-called prototypes, whose cardinality is fixed in advance, by minimizing the empirical misclassification cost.

The problem is shown to be $\mathcal{NP}$-Hard. Mixed Integer Programming (MIP) programs are formulated, theoretically compared and solved by a standard MIP solver for problems of small size. Large sized problem instances are solved by a variable neighborhood metaheuristic yielding good classification rules in reasonable time.

Keywords: Data Mining, Classification, Optimal Prototype Subset, Nearest Neighbor, Integer Programming.

1. Introduction

In a Classification problem, one has a database with individuals of $|C|$ different classes, and one wants to derive a *classification rule*, i.e., a procedure which labels every future entry v as member of one of the $|C|$ existing classes.

Roughly speaking, classification procedures can be divided into two types: *parametric* and *non-parametric*. Parametric procedures assume that each individual from class $c \in C$ is associated with a random vector with known distribution, perhaps up to some parameters, to be estimated, (e.g. data are multivariate normal vectors, with unknown mean μ_c and covariance matrix Σ_c), and use the machinery of Statistics as main technique, see e.g. [21].

For complex databases, with no evident distributional assumptions on the data (typically the case of databases with a mixture of quantitative and qualitative variables), non-parametric methods, as the one described in this talk, are needed.

In recent years there has been an increasing interest in deriving (non-parametric) classification rules via Mathematical Programming. Most of such methods require, for each individual i, a vector v^i of n numerical variables. In particular this assumes variables to be ratio-scaled,

*Partially supported by projects BFM2002-04525-CO2-02, Ministerio de Ciencia y Tecnología, Spain, and FQM-329, Junta de Andalucía, Spain

and not nominal or ordinal. Moreover, no blanks are allowed, which excludes its direct use for cases in which some measures are missing or simply do not apply. See e.g. Cristianini and Shawe-Taylor [5], Freed and Glover [9], Gehrlein [10], Gochet et al. [12] and Mangasarian [20].

A more flexible methodology, which just requires the knowledge of a metric (or, with more generality, a dissimilarity), is the Nearest Neighbor (NN) method [4,6,7,14], which provides, as documented e.g. in [16], excellent results.

In Nearest Neighbor methods, for each new entry i, the distances (or dissimilarities) $d(i,j)$ to some individuals j in the database (called *prototypes*) are computed, and i is classified according to such set of distances. In particular, in the classical NN, [4], all individuals are prototypes, and i is classified as member of class c^* to which its closest prototype j^* (satisfying $d(i,j^*) \leq d(i,j)\,\forall j$) belongs.

A generalization of the NN is the k-NN, e.g. [7], which classifies each i in the class most frequently found in the set of k prototypes closest to i. In particular, the NN is the k-NN for $k = 1$.

These classification rules, however, require distances to be calculated to all data in the database for each new entry, involving high storage and time resources, making it impractical to perform on-line queries.

For these reasons, several variants have been proposed in the last three decades, see e.g. [1,2,6,7,11,13,17,18] and the references therein. For instance, Hart [13] suggests the Condensed Nearest Neighbor (CNN) rule, in which the full database J is replaced by a certain subset I, namely, a so-called minimal consistent subset: a subset of records such that, if the NN is used with I (instead of J) as set of prototypes, all points in J are classified in the correct classes.

Since such minimal consistent subset can still be too large, several procedures have been suggested to reduce its size. Although such procedures do not necessarily classify correctly all the items in the database, (i.e., they are not consistent), they may have a similar or even better behavior to predict class membership on future entries because they may reduce the possible overfitting suffered by the CNN rule, see e.g. [3,19].

In this talk we propose a new model, in which a set of prototypes $I \subset J$, of prespecified cardinality p is sought, minimizing an empirical misclassification cost. Hence, if p is taken greater or equal than the cardinality p^* of a minimal consistent subset, then all individuals in the training sample will be correctly classified, yielding an empirical misclassification cost of zero. On the other hand, for $p < p^*$ we allow some data in the training sample to be incorrectly classified with the hope of reducing the possible overfitting which the classifier based on a minimal consistent subset might cause. Additionally, the effort needed to classify a new entry is directly proportional to p, which may therefore serve in practice to guide the choice of an upper bound on p.

For simplicity we restrict ourselves to the classification rule based on the closest distance, and hence can be seen as a variant of the NN rule. However, the results developed here extend directly to the case in which the k closest distances, $k \geq 1$, are considered in the classification procedure, leading to a variant of the k-NN method.

The talk is structured as follows. First, the mathematical model is introduced, showing that it is $\mathcal{NP}$-Hard. Two Integer Programming formulations are proposed and theoretically compared. Numerical results are given, showing that, when the optimization problems are solved exactly (with a standard MIP solver) the classification rules for $p < p^*$ behaves better than or equal to the CNN rule, but with enormous preprocessing times. For this reason, a heuristic procedure is also proposed, its quality and speed being also explored. It is shown that the rules obtained with this heuristic procedure have similar behavior on testing samples as the optimal ones.

References

[1] M.V. Bennett and T.R. Willemain. The Filtered Nearest Neighbor Method for Generating Low-Discrepancy Sequences. INFORMS Journal on Computing 16:68–72, 2004.

[2] J.C. Bezdek and L.I. Kuncheva. Nearest Prototype Classifier Designs: an Experimental Study. *International Journal of Intelligent Systems*, 16:1445–1473, 2001.

[3] H. Brighton and C. Mellish. Advances in Instance Selection for Instance-Based Learning algorithms. *Data Mining and Knowledge Discovery*, 6:153–172, 2002.

[4] T.M. Cover and P.E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[6] B.V. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[7] L. Devroye, L. Györfi and G. Lugosi. *A probabilistic Theory of Pattern Recognition*. Springer, 1996.

[8] R.A. Fisher. The Use of Multiple Measurements in Taxonomy Problems. *Annals of Eugenics*, 7:179–188, 1936.

[9] N. Freed and F. Glover. Simple but Powerful Goal Programming Models for Discriminant Problems. *European Journal of Operational Research*, 7:44–60, 1981.

[10] W.V. Gehrlein. General Mathematical Programming Formulations for the Statistical Classification Problem. *Operations Research Letters*, 5(6):299–304, 1986.

[11] S. Geva and J. Sitte. Adaptive Nearest Neighbor Pattern Classifier. *IEEE Trans Neural Networks*, 2(2):318–322, 1991.

[12] W. Gochet, A. Stam, V. Srinivasan and S.X. Chen. Multigroup Discriminant Analysis Using Linear Programming. *Operations Research*, 45:213–225, 1997.

[13] P.E. Hart. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[14] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[15] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, 1990.

[16] R.D. King, C. Feng and A. Sutherland. Statlog: Comparision of Classification Algorithm in Large Real-word Problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[17] L.I. Kuncheva. Fitness Function in Editing k-NN Reference Set by Genetic Algorithms. *Pattern Recognition*, 30(6):1041–1049, 1997.

[18] L.I. Kuncheva and J.C. Bezdek. Nearest Prototype Classification: Clustering, Genetic Algorithm or Random Search? *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 28(1):160–164, 1998.

[19] U. Lipowezky. Selection of the Optimal Prototype Subset for 1-NN Classification. *Pattern Recognition Letters*, 19:907–918, 1998.

[20] O.L. Mangasarian. Missclasification Minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[21] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

Branch-and-Bound for the semi-continous quadratic mixture design problem (SCQMDP)*

Leocadio G. Casado[1], Eligius M.T. Hendrix[2], and Inmaculada García[1]

[1] *Departmento de Arquitectura de Computadores y Electronica, Universidad de Almería,* {leo,inma}@ace.ual.es

[2] *Operationele Research en Logistiek Groep, Wageningen Universiteit,* eligius.hendrix@wur.nl

Abstract The semi-continuous quadratic mixture design problem (SCQMDP) is described as a problem with linear, quadratic and semi-continuity constraints. Moreover, a linear cost objective and integer valued objective are introduced. The research question is to deal with the SCQMD Problem from a Branch-and-Bound point perspective. Therefore, an algorithm is outlined that contains several variants. In the full paper the variants are tested on several cases derived from industry. Moreover, the technique of grid search is used as a benchmark and new theoretical results on the number of points and number of subsets are derived.

Keywords: Blending, Branch-and-Bound, Linear and Quadratic Constraint, semi-continuity.

1. Introduction

Consider the following formulation of the SCQMD problem which actually consists of identifying mixture products, each represented by a vector $x \in R^n$, which meet certain requirements. The set of possible mixtures is mathematically defined by the unit simplex $S = \{x \in R^n | \sum_j x_j = 1.0;\ x_j \geq 0\}$, where the variables x_j represent the fraction of the components in a product x.

Additionally, there are linear inequality constraints and bounds defining the design space $X \subset S$. The requirements are defined as quadratic inequalities.

$$g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0;\ \ i = 1, \ldots, m \tag{1}$$

in which A_i is a symmetric n by n matrix, b_i is an n-vector and d_i is a scalar. In this way we formulate the problem to be solved as finding elements of the set of "satisfactory" (feasible) products $D = \{x \in X | g_i(x) \leq 0;\ \ i = 1, \ldots, m\}$. This defines the quadratic mixture design problem (QMDP), as studied in [1].

In mixture design (blending) problems, the cost of the material, $f(x) = c^T x$ is minimised, where vector c gives the cost of the raw materials. In practical situations, companies are interested in find a feasible mixture $x \in D$, minimising its cost and also in minimising the number of raw materials in the mixture $t(x) = \sum_{j=1}^{n} \delta_j$, where

$$\delta_j = \begin{cases} 1 & \textbf{if} \quad x_j > 0, \\ 0 & \textbf{if} \quad x_j = 0. \end{cases} \tag{2}$$

*This work has been partially supported by the Ministry of Education and Science of Spain through grant CICYT-TIC2002-00228.

Moreover, usually there is a minimum acceptable dose md. This leads to using semi-continous variables in the problem description, i.e. either $x_j = 0$ or $x_j \geq md$. A common way to formulate semi-continuity is as follows [3]: $x_j \leq \delta_j;\ j = 1, \ldots, n$ and $x_j \geq \delta_j \cdot md;\ j = 1, \ldots, n$.

The following question is how to solve the SCQMD problem. The industry the problem originates from, applies different approaches to generate good acceptable solutions. Solvers are usually based on concepts such as generating random solutions, clustering, non-linear programming local search and grid search [2]. The research question of our investigation is whether we can construct a specific Branch-and-Bound approach for the SCQMD problem, taken into accont its semi-continuous character, implement it and test it on several practical cases. First in Section 2, we introduce many general ingredients of a specific algorithm for the SCQMD problem. In the Section 3, convergence ideas are discussed from the point of view of grid search.

2. Ingredients of a procedure for the SCQMDP

For setting up the Branch-and-Bound procedure for SCQMDP, many choices have to be made. Not only the shape of the partition sets, the way of branching, the selection of the set to be split further, etc. but also how to deal with all the goals: The linear constraints defining X, the quadratic requirements (1) defining D, the cost objective f and number of raw materials objective t. Moreover, to force theoretical convergence and due to some practical reasons one would like to discard partition sets that become too small. First we describe here the ingredients of the specific algorithm: the partition sets and the way of pruning. In the full paper an algorithmic scheme is described that leaves some choices in the applied rules. Moreover, the performance of variants applied to several cases is reported in there.

2.1 Partition sets and their vertices

In this work, simplicial subsets C_k are applied where the first subset is the unit simplex, $C_1 = S$. As a splitting rule, bisection along the longest edge is applied. Every vertex v generated in this way is evaluated:

- It is checked, whether the vertex fulfils the linear restrictions of X.
- The quadratic properties $g_i(v)$ are evaluated.
- The linear cost function $f(v)$ is evaluated.
- The number of used raw materials ($v_j > 0$) is counted i.e. $t(v)$ is evaluated.
- Finally it is checked whether the generated mixture v fails the semi-continuity restriction $0 < v_j < md$ for one of the raw materials j.

All this information can be used to determine good mixtures and to throw out those regions, where the best design cannot be situated. This is sketched in the following section. First of all, let's discuss the choice of the partition sets. Intuitively, the choice may be strange, but some advantages are outlined here. One of the strange characteristics is that most of the volume of the first set C_1 may be located in the md-interior $\{x \in R^n | \sum_j x_j = 1.0;\ x_j > md\}$, which as such is an unappealing part of the search space. Keep in mind that one tries to minimise t. Another approach would be to fix an upper bound RM on the number of raw materials $t \leq RM$. This would imply that one starts with a list of all facets (lower dimensional simplices here) containing RM raw materials i.e. $\begin{pmatrix} n \\ RM \end{pmatrix}$ subsets. In the current algorithm we will deal with t more as an objective to be minimised than as a predefined goal. An appealing characteristic of the choice of the simplicial shape and bisection branching is that we start with the vertices of the unit simplex where $t = 1$. During the bisection, every splitting will only increase the number of raw materials with at most one. In this way we start sampling in the most interesting areas with respect to the number of raw materials. A side effect was noticed where an implementation of [1] was applied in an industrial environment: the values of all generated feasible designs are a multiple of $(1/2)^K$, where K is an integer representing the depth of the search tree.

Another advantage of the bisection splitting is due to the shape of the partition sets. The length of the longest edge is at most twice the size of the shortest edge. Therefore the sets never get a needle shape. For every stored partition set C_k we keep the following data needed for the pruning based on the information in its vertices v:

- A lower bound zf_k of the cost function: $min_{v \in C_k} f(v)$
- A lower bound zt_k on the number of raw materials: $min_{v \in C_k} t(v)$

2.2 Storing and pruning partition sets

In a Branch-and-Bound algorithm, partition sets C_k where the optimum can be, are stored in memory. The challenge is not to let the algorithm fill all memory of the computer. This is attempted by pruning, i.e. discarding areas where the optimum cannot be located. In the SCQMDP we have several considerations for throwing away the area; it cannot contain a feasible point and/or cannot contain an optimal point. A last consideration is to throw out partition sets that appear to be small. We will discuss this topic separately. First we discuss the reasons why to throw out partition sets.

- First, one can check the linear constraints of X. If for one of the constraints all vertices of C_k are infeasible, we can discard C_k.
- The feasibility check for the quadratic constraints is more complicated. If for one of the constraints i all vertices of C_k are infeasible, we determine a lower bound zg_{ik} are determined as outlined in the full paper. If $zg_{ik} > 0$, C_k cannot contain a point that fulfills $g_i(x) \leq 0$. Therefore C_k can be discarded.
- Besides linear and quadratic feasibility, an evaluated vertex v is only labeled feasible, when its individual doses v_j are heaving a value that is either zero or greater or equal than the minimum dose md. A reason for a subset to be discarded would be that all its vertices are having doses $0 < v_j < md$, so it would be completely in the interior of the md-boundary of the unit simplex. Given the bisection way of splitting the occurence will be deep in the search tree.
- We are dealing with two objectives, linear f and integer t we want to minimise i.e. a so-called bi-objective problem. Finally we are looking for Pareto optimal products that, as illustrated in Figure 1 have a minimum cost given a number of raw materials. This means we can throw out subsets only when they are dominated, i.e. there exists a feasible product somewhere else with as well a better value for the cost f as for the number of raw materials t. To check this, we introduce a vector of upper bounds $zf^U(q)$: best cost found thus far for a feasible product with at most q raw materials, $q = 1, \ldots n$. Initially its values are set on ∞ and, as soon as a feasible vertex is evaluated, values are updated according to the definition, resulting finally in the situation as sketched in Figure 1. Given the lower bounds zf_k and zt_k, now a subset C_k is known to be dominated and can be discarded if $zf_k > zf^U(zt_k)$. Notice that if $zf^u(q)$ was found, we can set the values of $zf^u(q+i),\ i = 1, \ldots n-q$ to $zf^u(q)$ if $zf^u(q+i) > zf^u(q)$.

Besides the pruning rules mentioned, we would like to throw out small partition sets, because they occupy memory space and they will not generate further information. Morover, throwing out subsets with $size(C_k) < \epsilon$ theoretically will lead to convergence. Two types of candidates for removal may still be on the list:

1. Subsets may have passed all tests before, despite that non of the vertices appeared feasible: $\forall v \in C_k \ \exists i, g_i(v) > 0$ and moreover all lower bounds are smaller than zero; $zg_i < 0$. One reason to throw them out would be the fact that such a set does not contain an ϵ-robust point, i.e. an element of $\{x \in D | (x+h) \in D, \forall h, \|h\| \leq \epsilon\}$.

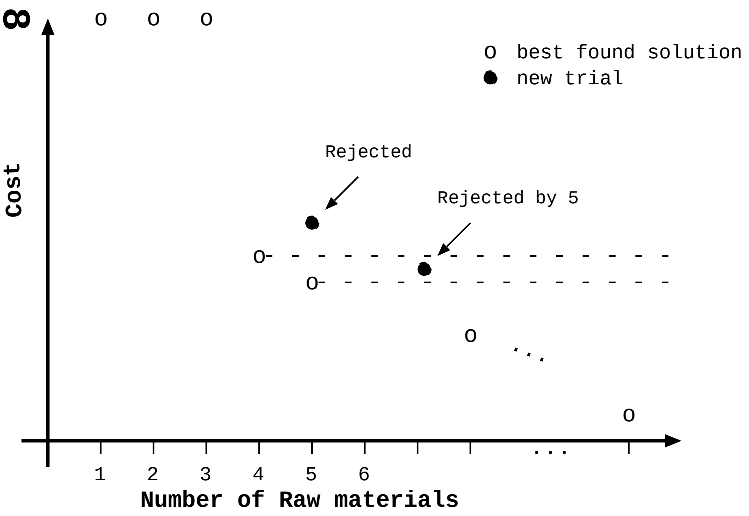


Figure 1. Rejection due to domination, Pareto optimality

2. Subsets that contain a feasible vertex. The reason to discard small subsets is the idea that exploring further would only lead to a marginal gain in objective function value.

The sampling part of the algorithm consists of generating and evaluating candidate points that are vertices. The Branch-and-Bound part with the pruning takes care of the guarantee of infeasibility or optimality of the points found. For every value of the number of raw materials $q = 1, \ldots, n$, we open a list $Q(q)$ to save the best points, also called incumbent minimum points. Usually, for every (maximum) number of raw materials q, only one optimal solution will exist, so we expect $Q(q)$ to contain either no point, because q is so low that no feasible point exists, or it contains one point with an objective function value of $zf^U(q)$.

3. Convergence

As sketched above, the algorithm has a certain guarantee to reach optima. Why the ϵ-accuracy gives an effectiveness of the algorithm is outlined here. Another question is of course the efficiency; how many function evaluations are necessary and how many simplices should be stored in a worst case? To understand the convergence concepts, it is useful to think in an α-grid over the unit simplex. It is also convenient to apply the ∞-norm as a distance measure, where the size of the unit simplex, i.e. the largest edge is 1 (in euclidean distance it is $\sqrt{2}$). An algorithm applying bisection does not automatically generate points on a regular grid. A rule is required for choosing the edge to be split, when there are several edges of the same size.

Consider a regular grid with M equal distant values for every axis, resulting in mesh size of $\alpha = 1/(M-1)$. The concept of evaluating points over a grid, is that when $\alpha \leq \epsilon$, we are certain to be closer than ϵ to the best solution, or alternatively, know that an ϵ-robust solution does not exist. A strategy to evaluate all grid points is not popular, as it is not very efficient. When performed on a unit box, the number of function evaluations grows exponentially with the dimension: M^n. This is not that bad on the unit simplex, as we are dealing with the mixture equality $\sum_j x_j = 1.0$. This is illustrated in Figure 2.

It can be verified that the total number of points on the grid is given by

$$\sum_{k=1}^{n-1} \begin{pmatrix} M \\ k \end{pmatrix} \begin{pmatrix} n-2 \\ k-1 \end{pmatrix} \tag{3}$$

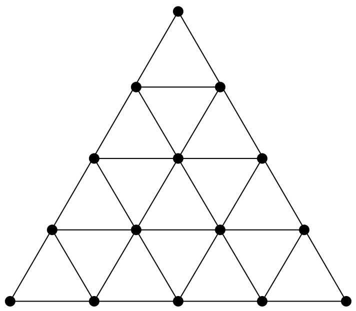


Figure 2. 2-dim projection of regular grid over the unit simplex, M=5

as sketched in Table 1. This means that the number of points, although very high, is not even exponential in the dimension. For the example in Figure 2, $n = 3$ and $m = 5$, we have 15 grid points and an "accuracy" of $\alpha = 0.25$.

Table 1. Number of regular grid points on the unit simplex

	n=2	n=3	n=4	n=5	n=6	n=7
M=11	11	66	286	1001	3003	8008
M=101	101	5151	176851	$4.6\ 10^6$	$9.7\ 10^7$	$1.7\ 10^9$

Notice that the concept of Branch-and-Bound is not to generate all these points, as we can throw out parts of the area where the optima cannot be found. However, if one studies the bisection process, we arrive for the same example at more points as illustrated by Figure 3.

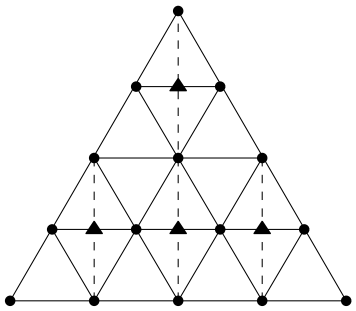


Figure 3. Bisection process 2-dim projection, $\epsilon \geq 0.25$

As such this is not a disaster. The main question is what happens with the storage of subsets; the number of stored simplices should not grow out of hand. Before dealing with that, let us remark with respect to Figure 3, that the picture does not appear straightforward. In the implementation of the bisection, a choice rule is required that determines in case of equal sized largest edges, which one is to be bisected first. In our implementations we did so on the basis of the co-ordinates. To ensure reproducibility of results, it should not be chosen at random. The number of simplices that is generated (and stored) in the worst case, depends on many aspects. We could derive an upper bound and a lower bound on the worst case performance. In the worst case, rules lead to splitting and storing simplices that have a size slightly larger than ϵ. At every bisection over a midpoint, one edge is halved and several are getting shorter. This means that after going $n(n-1)/2$ deeper in the search tree, at least all edges have been halved and the size of the simplex is less than half its original size. The maximum number K of halving the simplices is given by $1/2^K \leq \epsilon$, such that $K = \lceil(-ln\epsilon/ln(2))\rceil$, where $\lceil x\rceil$ is the lowest integer greater than or equal to x. Given the number of edges per simplex $n(n-1)/2$, the maximum depth of the search tree is $K \times n(n-1)/2$. The final level is not stored, as the simplices don't pass the size test. An overestimate for the worst case number of simplices on the list is $2^{K \times n(n-1)/2-1}$, where $K = \lceil(-ln\epsilon/ln(2))\rceil$. This analysis provides a guarantee that the algorithm is finite given an accuracy. Looking more realistically, this upper bound of an upper bound also leads to dispare; for $\epsilon = 1\%$, it gives a bound of the order of 10^6 for $n = 3$ and 10^{44} for $n = 7$. This does not sound very encouraging nor realistic.

Let us now consider a lower bound on the worst case behaviour. Consider again the regular grid in Figure 2. Suppose an algorithm would generate with an accuracy of $\epsilon = \frac{1}{M-1}$ all corresponding simplices of the regular mesh. We know that bisection has to generate more in the worst case. How many simplices would the algorithm generate? The number of simplices in the regular grid is

$$(M-1)^{n-1} \tag{4}$$

Formula (4) can be derived from volume considerations. The unit simplex represents a volume in $n-1$ dimensional space proportional to $Size^{n-1}$. As the size of a simplices within the regular grid has a size of $\frac{1}{M-1}$ times the unit simplex size, its volume is $\left(\frac{1}{M-1}\right)^{n-1}$ that of the unit simplex. Also the number (4) turns out big; for $\epsilon = 1\%$, it gives 10^4 for $n = 3$ and 10^{12} for $n = 7$. We will observe in the experiments that practically the number is much lower. The real success of Branch-and-Bound depends on how good parts of the tree can be pruned. Search strategies on deep or wide search determine the final result.

4. Summary

This extended abstract describes the quadratic blending problem with additional complications of semi-continuity and bi-objectivity. The ingredients are sketched of a Branch-and-Bound approach under investigation. Some convergence analyses based on the benchmark of grid search are given. In the full paper empirical results on cases derived from industry are reported.

References

[1] Eligius M. T. Hendrix and Janos D. Pinter. An application of Lipschitzian global optimization to product design. *J. Global Optim.*, 1:389–401, 1991.

[2] R. Horst, P.M. Pardalos, and N.V. Thoai, editors. *Introduction to Global Optimization*, volume 3 of *Noncovex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, Holland, 1995.

[3] H. P. Williams. *Model Building in Mathematical Programming*. Wiley & Sons, Chichester, 1993.

Reliable Optimization in Civil Engineering — Structural and Numerical Optimization of Civil Engineering Problems[*]

András Erik Csallner,[1] Tibor Csendes,[1] and András Balázs Kocsis[2]

[1] *University of Szeged, Szeged, Hungary,* csallner@jgytf.u-szeged.hu

[2] *KÉSZ Ltd., Szeged, Hungary*

Abstract It is shown in this work, how engineering problems, especially some kinds of civil engineering problems can be solved using verified techniques exploiting the knowledge of structure and engineers' habits. Some numerical results are also mentioned.

Keywords: Civil engineering, patents, reliable methods.

1. Introduction

Civil engineering is a field — as many of other engineering sciences (e.g. see [1]) — where the most of methods used for solving optimization problems are based on experience and experiments, and models using local information, however drawn from global models [6]. The present work outlines interesting classes of problems from this field, and initiates some possible ways to solve those problems utilizing the wide tool capabilities of interval arithmetic for error handling and interval branch-and-bound algorithms [4, 5, 7, 8] to solve the original or modified industrial models automating civil engineers' work. The investigations are in an early stage but are promising both in a theoretical and in a practical sense. The final aim is to contribute to the refinement of emerging industrial standards and patents such as [9–12].

Reliable numerical techniques have substantial advantages and a bright future in civil engineering computational procedures, still there is a long way to go. The talk highlights a simple example problem with the traditional engineering solution and with the interval arithmetic based result of tolerance optimization. We report on the solution and numerical experiences collected investigating civil engineering design problems. A subsequent careful study must clear the consequences and advantages of such approaches with special attention to the related application fields.

2. A Simple Problem

The problems arising in civil engineering are MINLP problems. The integer part describes the decision of different materials and properties which are scaled (e.g. the thickness of different enforcement steel bars), whilst the nonlinear part realizes the physical model in most of the cases.

[*] This work was supported by the KÉSZ Ltd. (www.kesz.hu), and by the Grants OTKA T 048377, and T 046822.

A simple example for a civil engineering problem consisting only of the nonlinear part is the simplified footing design problem [2]. The problem was to find the minimal volume dimensions of a concrete footing in such a way that given constraints hold. In detail:

$$\min h\mu B^2 \tag{1}$$

subject to

$$\sigma_{Sd} \le \sigma_{Eff},$$

$$1 \le \mu \le 5,$$

$$0.5\mu B \le h \le -a_s - 0.5,$$

$$B > 0,$$

$$-\frac{5}{6}B \le a_x \le \frac{5}{6}B,$$

$$-\frac{5}{6}\mu B \le a_y \le \frac{5}{6}\mu B.$$

where a_x, a_y, h, μ and B are the variables to be optimized. The functional dependencies are the following:

$$R = P_z + \mu h B^2 \overline{\rho}_{rc},$$

$$\sigma_{Sd} = \frac{R}{\left(B - 2|\frac{P_z a_x + P_x h - M_y}{R}|\right)\left(\mu B - 2|\frac{P_z a_y + P_y h + M_x}{R}|\right)},$$

$$N_t = e^{\pi \tan\phi} \tan^2\left(45^o + \frac{\phi}{2}\right),$$

$$f = \frac{P_x}{P_z},$$

$$i_t = (1 - 0.7f)^3$$

$$\sigma_1 = \left(1 - \frac{1}{3\mu}\right)\gamma_1 B (N_t + 1)\tan\phi(1-f)^3 j_B,$$

$$\sigma_2 = \left(1 + \frac{1}{2\mu}\right)\left(\gamma_2(|a_s| - 0.1)N_t i_t j_t + c(N_t - 1)\cot\phi\left(i_t - \frac{1 - i_t}{N_t - 1}\right)j_c\right),$$

$$\sigma_{Eff} = \alpha_1\alpha_2\alpha_3(\sigma_1 + \sigma_2).$$

The variable h contains the height of the footing, B is its width, and the respective length is μB. The building's support point is shifted by the vector (a_x, a_y) from the center of the footing's projection to the x-y plane.

An interval method could supply a good inclusion for a suboptimal feasible set, while the traditional engineers' approach could only point out a single solution of this set near to the border of it. Note that we were not able to find the global minimum even in this simple case.

3. Structural Optimization and Verified Iteration

It is obvious that involving integer variables into models like the one above makes things even more difficult. Our aim is to find simplified models first, which can be solved by verified methods so that we can learn more about the structure of civil engineering problems.

Unfortunately our first endeavors are not very successful but the oral presentation enlightens the methodology of this field to be able to achieve results of practical use. For this purpose it is considered, how in a special but widely used class of problems, i.e., the reinforced concrete beam design problem the engineers' experiences can be exploited. First, the handling of the integer parameters has to be cleared up. The steel diameters, number of reinforcement bars, steel quality, and concrete quality can be predicted based on some expertise. This leads to an NLP, where the stirrup design is still an integer problem in reality, but appears in the model as the weight of the used stirrup steel mass. After all, the model which is not to be shown here for the sake of simplicity is a highly nonlinear optimization problem. Since the objective of the problem is the cost, but several stability parameters reach the optimality at the border of the feasible set determined by the design rules, it is better to use a suboptimal solution set to choose from, to prevent the beam from breaking. The problem is solved with the aid of a simple idea, called the Verified Iterating Method [3]:

1. Find an initial feasible interval of parameters exploiting engineers' knowledge (approaching parameters from 'below').

2. Grow a set of feasible intervals around this set with halving growth.

3. Use this feasible set as an input of an advanced global optimization verified interval search.

In step 1, a special verified local search is started to have an initial point or interval we can grow. Then in step 2 intervals are grown around this point with halving width with a given accuracy. After these two steps we have a suboptimal feasible solution set. If we need an optimal result, then this can be tightened by our desire in step 3.

4. Summary

Unfortunately both the structural stage and the Verified Iterating Method set a few further questions.

1. It is a rough estimate to make a structural pre-optimization isolated from the nonlinear second stage. Usually we tried to incorporate the scaled variables in a special way into the second stage to obtain better results.

2. Step 3 of the Verified Iterating Method has not worked even for the much simpler footing problem. That is, our first attempts to solve these optimization problems with ordinary verified interval methods failed. So we cannot expect success for a much more difficult problem over a feasible set of several subintervals.

The presentation will, however, contain some new numerical results to demonstrate how problems like the two above can be eliminated in future.

References

[1] Csallner, A.E.: Global optimization in separation network synthesis. Hungarian J. of Industrial Chemistry 21(1993) 303-308.

[2] Csallner, A.E., T. Csendes, A.B. Kocsis: Reliable numerical computation in civil engineering, Numerical Algorithms, Vol. 37(2004) 85-91.

[3] Csallner A.E., A.B. Kocsis: Handling Interval Constraints in Civil Engineering Problems, IMACS/GAMM SCAN-2004 Conference, Fukuoka, Japan, October 4-8, 2004.

[4] Csendes, T., Z.B. Zabinsky, and B.P. Kristinsdottir: Constructing large feasible suboptimal intervals for constrained nonlinear optimization. Annals of Operations Research, 58(1995) 279-293.

[5] Csendes, T.: Optimization methods for process network synthesis — a case study, In: Christer Carlsson and Inger Eriksson (eds.): Global & multiple criteria optimization and information systems quality. Abo Academy, Turku, 1998, pp. 113-132.

[6] Gyengő, T.: Vasbeton szerkezetek elmélete, méretezése és szerkezeti kialakítása (In Hungarian: The theory, design and structural shaping of reinforced concrete structures), Műszaki Könyvkiadó, Budapest, 1960.

[7] Kristinsdottir, B.P., Z.B. Zabinsky, T. Csendes, and M.E. Tuttle: Methodologies for tolerance intervals. Interval Computations, 3(1993) 133-147.

[8] Kristinsdottir, B.P., Z.B. Zabinsky, M.E. Tuttle, and T. Csendes: Incorporating manufacturing tolerances in optimal design of composite structures, Engineering Optimization 26(1996) 1-23.

[9] MSZ 15002-1987: Építmények alapozásának erőtani tervezése. Általános méretezési előírások (in Hungarian: Mechanic Design of the Foundations of Buildings. General Sizing Requirements).

[10] MSZ ENV 1992-1-1/1999: Eurocode2: Betonszerkezetek tervezése, 1-1 rész: Általános és az épületekre vonatkozó szabályok (in Hungarian: Eurocode2: Design of Concrete Structures, Part 1-1: General Rules and Rules for Buildings).

[11] NAD MSZ ENV 1992-1-1: Magyar Nemzeti Alkalmazási Dokumentum az Eurocode2 szabványhoz. Betonszerkezetek tervezése (in Hungarian: Hungarian National Application Document for Eurocode2. Design of Concrete Structures).

[12] MSZ ENV 1997-1-1: Eurocode7. Geotechnikai tervezés (in Hungarian: Eurocode7: Geotechnical Design).

A global optimization model for locating chaos*

Tibor Csendes[1], Balázs Bánhelyi[1], and Barnabás Garay[2]

[1] *University of Szeged, Szeged, Hungary,* csendes|banhelyi@inf.u-szeged.hu

[2] *Budapest University of Technology, Budapest, Hungary,* garay@math.bme.hu

Abstract We present a global optimization model to find chaotic regions of certain dynamic systems. The technique has two innovations: first an interval arithmetic based guaranteed reliability checking routine to decide whether an inclusion relation holds, and a penalty function based nonlinear optimization problem that enables us to automatize the search for fitting problem instances. We provide the theoretical results proving correctness and convergence properties for the new algorithm. A companion talk discusses the results achieved by the presented method.

Keywords: Chaos, Computer-aided proof, Global optimization.

An important question is while studying approximations of the solutions of differential equations, whether the given problem has a chaotic solution. The problem is usually solved by careful studying of the given problem with much human interaction, followed by an estimation of the Lipschitz constant, bounding the rounding errors to be committed, and finally a number of grid points are checked one by one by a proper computer program [6].

We study verified computational methods to check and locate regions the points of which fulfill the conditions of chaotic behaviour. The investigated Hénon mapping is $\mathcal{H}(x,y) = (1 + y - Ax^2, Bx)$. The paper [6] considered the $A = 1.4$ and $B = 0.3$ values and some regions of the two dimensional Euclidean space: $E = E_1 \cup E_2 = \{(x,y) \mid x \geq 0.4,\ y \geq 0.28\} \cup \{(x,y) \mid x \leq 0.64,\ |y| \leq 0.01\}$, $\mathcal{O}_1 = \{(x,y) \mid x < 0.4,\ y > 0.01\}$, $\mathcal{O}_2 = \{(x,y) \mid y < 0\}$.

According to [6] Theorem 1 below ensures the chaotic behaviour for the points of the parallelograms Q_0 and Q_1 with parallel sides with the x axis (for $y_0 = 0.01$ and $y_1 = 0.28$, respectively), with the common tangent of 2, and x coordinates of the lower vertices are $x_a = 0.460$, $x_b = 0.556$; and $x_c = 0.558$, $x_d = 0.620$, respectively. The mapping and the problem details (such as the transformed sides of the parallelograms, $H^7(a)$, $H^7(b)$, $H^7(c)$, and $H^7(d)$) are illustrated on Figure 1.

Theorem 1. *Assume that the following relations hold for the given particular Hénon mapping:*

$$\mathcal{H}^7(a \cup d) \subset \mathcal{O}_2, \tag{1}$$

$$\mathcal{H}^7(b \cup c) \subset \mathcal{O}_1, \tag{2}$$

$$\mathcal{H}^7(Q_0 \cup Q_1) \subset \mathbb{R}^2 \setminus E, \tag{3}$$

then chaotic trajectories belong to the starting points of the regions Q_0 and Q_1.

To check the inclusion relations required in Theorem 1 we have set up an adaptive subdivision algorithm based on interval arithmetic:

*This work has been partially supported by the Bilateral Austrian-Hungarian project öu56011 as well as by the Hungarian National Science Foundation Grants OTKA No. T 037491, T 032118, T 034350, T 048377, and T 046822.

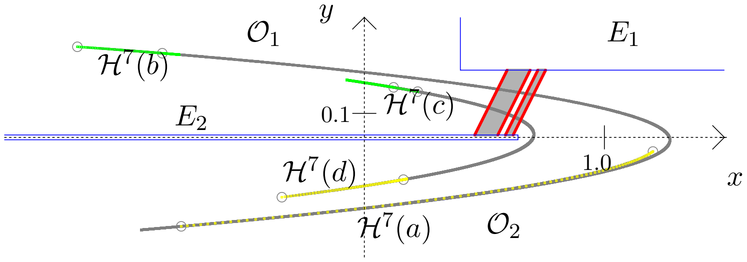


Figure 1. Illustration of the H^7 transformation for the classic Hénon parameters $A = 1.4$ and $B = 0.3$ together with the chaotic region of two parallelograms. The a, b, c, and d sides of the parallelograms are depicted on the upper left picture of Figure 2.

Algorithm 1 : The Checking Routine

Inputs: – ε: the user set limit size of subintervals,
– Q: the argument set to be proved,
– $\mathcal{O}$: the aimed set for which $T(Q) \subset \mathcal{O}$ is to be checked.

```
Calculate the initial interval $I$, that contains the regions of interest
Push the initial interval into the stack
while ( the stack is nonempty )
  Pop an interval $v$ out of the stack
  Calculate the width of $v$
  Determine the widest coordinate direction
  Calculate the transformed interval $w = T(v)$
  if $v \cap Q \neq \emptyset$, and the condition $w \subset \mathcal{O}$ does not hold, then
    if the width of interval $v$ is less than $\varepsilon$ then
      print that the condition is hurt by $v$ and stop
      else bisect $v$ along the widest side: $v = v_1 \cup v_2$
        push the subintervals into the stack
    endif
  endif
end while
print that the condition is proven and stop
```

We have proven that this algorithm is capable to provide the positive answer after a finite number of steps, and also that the given answer is rigorous in the mathematical sense. Once we have a reliable computer procedure to check the conditions of chaotic behavior of a mapping it is straightforward to set up an optimization model that transforms the original chaos location problem to a global optimization problem.

The key question for the successful application of a global optimization algorithm was how to compose the penalty functions. On the basis of earlier experiences collected with similar constrained problems, we have decided to add a nonnegative value proportional to how much the given condition was hurt, plus a fixed penalty term in case at least one of the constraints was not satisfied.

As an example, consider the case when one of the conditions for the transformed region was hurt, e.g. when (2), i.e. the relation

$$\mathcal{H}^k(b \cup c) \subset \mathcal{O}_1$$

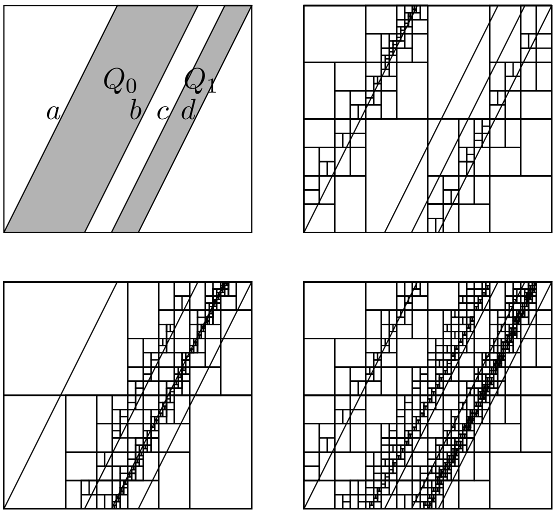


Figure 2. The parallelograms and the starting interval covered by the verified subintervals for which either the given condition holds (in the order of mentioning in Theorem 1), or they do not contain a point of the argument set.

does not hold for a given kth iterate, and for a region of two parallelograms. In such a case the checking routine will provide a subinterval that contains at least one point of the investigated region, and which contradicts the given condition. Then we have calculated the Hausdorff distance of the transformed subinterval $H^k(I)$ to the set $\mathcal{O}_1$ of the right side of the condition,

$$\max_{z \in H^k(I)} \inf_{y \in \mathcal{O}_1} d(z, y),$$

where $d(z, y)$ is a given metric, a distance between a two dimensional interval and a point. Notice that the use of maximum in the expression is crucial, with minimization instead our optimization approach could provide (and has provided) result regions that do not fulfill the given conditions of chaotic behaviour. On the other hand, the minimal distance according to points of the aimed set (this time $\mathcal{O}_1$) is satisfactory, since it enables the technique to push the search into proper directions. In cases when the checking routine answered that the investigated subinterval has fulfilled the given condition, we have not changed the objective function.

Summing it up, we have considered the following bound constrained problem for the T inclusion function of the mapping $\mathcal{T}$:

$$\min_{x \in X} g(x), \tag{4}$$

where

$$g(x) = f(x) + p\left(\sum_{i=1}^{m} \max_{z \in T(I)} \inf_{y \in S_i} d(z, y)\right),$$

X is the n-dimensional interval of admissible values for the parameters x to be optimized, $f(x)$ is the original, nonnegative objective function, and $p(y) = y + C$ if y is positive, and $p(y) = 0$ otherwise. C is a positive constant, larger than $f(x)$ for all the feasible x points, m is the number of conditions to be fulfilled, and S_i is the aimed set for the i-th condition.

The emerging global optimization problem has been solved by the clustering optimization method described in citecst. We have proven the correctness of this global optimization model:

Theorem 2. *For the bound constrained global optimization problem defined in (4) the following properties hold:*

1. In case a global optimization algorithm finds a point for which the objective function g has a value below C, i.e. when each penalty term $\max_{z \in T(I)} \inf_{y \in S_i} d(z, y)$ is zero, then all the conditions of chaos are fulfilled by the found region represented by the corresponding optimal parameters found. At the same time, the checking routine provided a guaranteed reliability computational proof of the respective subset relations.

2. In case the given problem does not have a parameter set within the bounds of the parameters to be optimized such that the corresponding region would fulfill the criteria of chaos, then the optimization cannot result in an approximate optimizer point with an objective function value below C.

Talk will provide an insight into the theoretical statements and their proofs. On this basis we have checked chaos for an earlier investigated 7th iterate Hénon mapping and also other problem instances, some of them have involved tolerance optimization too [3]. The numerical results (see also in [1], [4]) will be covered by an other talk.

Acknowledgments

This work has been partially supported by the Bilateral Austrian-Hungarian project öu56011 as well as by the Hungarian National Science Foundation Grants OTKA No. T 037491, T 032118, T 034350, T 048377, and T 046822.

References

[1] Bánhelyi, B., T. Csendes, and B.M. Garay: Optimization and the Miranda approach in detecting horseshoe-type chaos by computer. Manuscript, submitted for publication. Available at
`www.inf.u-szeged.hu/∼csendes/henon2.pdf`

[2] Csendes, T.: "Nonlinear parameter estimation by global optimization – efficiency and reliability", *Acta Cybernetica* **8**, 361–370, 1988.

[3] Csendes, T., Zabinsky, Z. B. & Kristinsdottir, B. P.: "Constructing large feasible suboptimal intervals for constrained nonlinear optimization", *Annals of Operations Research* **58**, 279–293, 1995.

[4] Csendes, T., Garay, B. M. & Bánhelyi, B.: "A verified optimization technique to locate chaotic regions of a Hénon system", Manuscript, submitted for publication. Available at
`www.inf.u-szeged.hu/∼csendes/henon.pdf`

[5] *C-XSC Languages home page:* `http://www.math.uni-wuppertal.de/org/WRST/index_en.html`

[6] Zgliczynski, P.: *Computer assisted proof of the horseshoe dynamics in the Hénon map.* Random & Computational Dynamics 5:1-17, 1997.

Global Optimization of Low-Thrust Space Missions Using Evolutionary Neurocontrol

Bernd Dachwald

German Aerospace Center (DLR), Institute of Space Simulation, Cologne, Germany, bernd.dachwald@dlr.de

Abstract Low-thrust space propulsion systems enable flexible high-energy deep space missions, but the design and optimization of the interplanetary transfer trajectory is usually difficult. It involves much experience and expert knowledge because the convergence behavior of traditional local trajectory optimization methods depends strongly on an adequate initial guess. Within this extended abstract, evolutionary neurocontrol, a method that fuses artificial neural networks and evolutionary algorithms, is proposed as a smart global method for low-thrust trajectory optimization. It does not require an initial guess. The implementation of evolutionary neurocontrol is detailed and its performance is shown for an exemplary mission.

Keywords: Evolutionary Neurocontrol, Spacecraft Trajectory Optimization, Low-Thrust Propulsion.

1. Introduction to Low-Thrust Missions

Innovative interplanetary deep space missions require ever larger velocity increments (ΔVs) and thus ever more demanding propulsion capabilities. Chemical high-thrust propulsion systems (rocket engines) have a limited ΔV-capability because the chemical energy of the propellant limits its exhaust velocity. Electric low-thrust propulsion systems can significantly enhance or even enable high-energy missions by providing much larger exhaust velocities than chemical high-thrust systems. Consequently, they permit significantly larger ΔVs, while at the same time allowing direct trajectories with simpler mission profiles, flexible launch windows, and mostly even reduced flight times because no gravity assist maneuvers are required. Another innovative low-thrust propulsion system is the solar sail, a large ultra-lightweight reflecting surface made of thin aluminized plastic film. Solar sails utilize solely the freely available solar radiation pressure for propulsion and require therefore no propellant at all. Therefore, their ΔV-capability depends only on their lifetime in the space environment (and their distance from the sun). Electric propulsion systems have already been successfully flown in space while solar sails are under development [10].

A spacecraft trajectory is, in simple terms, the spacecraft's path from A (the initial body or orbit) to B (the target body or orbit). In general, the optimality of a trajectory can be defined according to several objectives, like transfer time or propellant consumption. Because solar sailcraft does not consume any propellant, trajectories are typically optimized with respect to transfer time alone. Trajectory optimization for electric spacecraft is less straightforward because transfer time minimization and propellant minimization are mostly competing objectives. Spacecraft trajectories can also be classified with respect to the final constraints. If, at arrival, the position $\boldsymbol{r}_{\mathrm{SC}}$ and the velocity $\dot{\boldsymbol{r}}_{\mathrm{SC}}$ of the spacecraft must match that of the target, $\boldsymbol{r}_{\mathrm{T}}$ and $\dot{\boldsymbol{r}}_{\mathrm{T}}$ respectively, one has a rendezvous problem. If only the position must match, one has a flyby problem.

For spacecraft with high thrust, optimal interplanetary trajectories can be found relatively easily because few thrust phases are necessary during its "free fall" within the gravitational field of the solar system. These can be approximated by singular events that change the spacecraft's velocity instantaneously while its position remains fixed. Low-thrust propulsion systems, in contrast, are required to operate for a significant part of the transfer to generate the necessary ΔV. A low-thrust trajectory is obtained from the numerical integration of the spacecraft's equations of motion. Besides the inalterable external forces, the spacecraft trajectory $\boldsymbol{x}_{\mathrm{SC}}[t] = (\boldsymbol{r}_{\mathrm{SC}}[t], \dot{\boldsymbol{r}}_{\mathrm{SC}}[t])$ is determined entirely by the thrust vector history $\boldsymbol{F}[t]$ ('$[t]$' denotes the time history of the preceding variable, $\boldsymbol{x}_{\mathrm{SC}}$ is called the spacecraft state). The thrust vector $\boldsymbol{F}(t)$ of low-thrust propulsion systems is a continuous function of time. It is manipulated through the n_u-dimensional spacecraft control function $\boldsymbol{u}(t)$ that is also a continuous function of time. Therefore, the trajectory optimization problem is to find, in infinite-dimensional function space, the optimal spacecraft control function $\boldsymbol{u}^\star(t)$ that yields the optimal trajectory $\boldsymbol{x}^\star_{\mathrm{SC}}[t]$. This renders low-thrust trajectory optimization a very difficult problem that can not be solved except for very simple cases. What can be solved, at least numerically, however, is a discrete approximation of the problem. Dividing the allowed transfer time interval $[t_0, t_{f,\max}]$ into τ finite elements, the discrete trajectory optimization problem is to find the optimal spacecraft control history $\boldsymbol{u}^\star[\bar{t}]$ (the symbol $\bar{t}$ denotes a point in discrete time) that yields the optimal trajectory $\boldsymbol{x}^\star_{\mathrm{SC}}[t]$. Through discretization, the problem of finding the optimal control function $\boldsymbol{u}^\star(t)$ in infinite-dimensional function space is reduced to the problem of finding the optimal control history $\boldsymbol{u}^\star[\bar{t}]$ in a finite but usually still very high-dimensional parameter space. For optimality, some cost function J must be minimized. If the used propellant mass $\Delta m_{\mathrm{P}} = m_{\mathrm{P}}(\bar{t}_0) - m_{\mathrm{P}}(\bar{t}_f)$ is to be minimized, $J = \Delta m_{\mathrm{P}}$ is an appropriate cost function, if the transfer time $T = \bar{t}_f - \bar{t}_0$ is to be minimized, $J = T$ is an appropriate cost function.

2. Evolutionary Neurocontrol as a Smart Global Low-Thrust Trajectory Optimization Method

Traditionally, low-thrust trajectories are optimized by the application of numerical optimal control methods that are based on the calculus of variations. All these methods can generally be classified as local trajectory optimization methods (LTOMs), where the term optimization does not mean to find the best solution but rather to find *a* solution. The convergence behavior of LTOMs is very sensitive to the initial guess, which has to be provided prior to optimization by an expert in astrodynamics and optimal control theory. Because the optimization process requires nearly permanent expert attendance, the search for a good trajectory can become very time-consuming and expensive. Even if the optimizer finally converges to an optimal trajectory, this trajectory is typically close to the initial guess and that is rarely close to the (unknown) global optimum. Another drawback of LTOMs is the fact that the initial conditions (launch date $\bar{t}_0$, initial propellant mass $m_{\mathrm{P}}(\bar{t}_0)$, initial velocity vector $\dot{\boldsymbol{r}}_{\mathrm{SC}}(\bar{t}_0)$, etc.) – although they are crucial for mission performance – are generally chosen according to the expert's judgment and are therefore not explicitly part of the optimization problem.

To evade the drawbacks of LTOMs, a smart global trajectory optimization method (GTOM) was developed by the author [4]. This method – termed evolutionary neurocontrol (ENC) – fuses artificial neural networks (ANNs) and evolutionary algorithms (EAs) into so-called evolutionary neurocontrollers (ENCs). The implementation of ENC for low-thrust trajectory optimization was termed InTrance, which stands for **In**telligent **Tra**jectory optimization using **n**euro**c**ontroller **e**volution. To find a near-globally optimal trajectory, InTrance requires only the target body and intervals for the initial conditions as input. It does not require an initial guess or the attendance of a trajectory optimization expert. During the optimization process, InTrance searches not only the optimal spacecraft control but also the optimal initial conditions within the specified intervals.

A Delayed Reinforcement Learning Problem. In reinforcement learning (RL) problems, the optimal behavior of the learning system (called agent) has to be learned solely through interaction with the environment, which gives an immediate or delayed evaluation[1] J (also called reward or reinforcement) [11]. The agent's behavior is defined by an associative mapping from situations to actions $\mathsf{S} : \mathcal{X} \mapsto \mathcal{A}$. Here, this associative mapping that is typically called policy in the RL-related literature, is termed strategy. The optimal strategy $\mathsf{S}^\star$ of the agent is defined as the one that maximizes the sum of positive reinforcements and minimizes the sum of negative reinforcements over time. If, given a situation $X \in \mathcal{X}$, the agent tries an action $A \in \mathcal{A}$ and the environment immediately returns an evaluation $J(X, A)$ of the (X, A) pair, one has an immediate reinforcement learning problem. More difficult are delayed reinforcement learning problems, where the environment gives only a single evaluation $J(X, A)[t]$, collectively for the sequence of (X, A) pairs occurring in time during the agent's operation. From the perspective of machine learning, a spacecraft steering strategy may be defined as an associative mapping S that gives – at any time along the trajectory – the current spacecraft control $\boldsymbol{u}$ from some input $\boldsymbol{X}$ that comprises the variables that are relevant for the optimal steering of the spacecraft (the current state of the relevant environment). Because the trajectory is the result of the spacecraft steering strategy, the trajectory optimization problem is actually a problem of finding the optimal spacecraft steering strategy $\mathsf{S}^\star$. This is a delayed reinforcement problem because a spacecraft steering strategy can not be evaluated before its trajectory is known under the given environmental conditions (constellation of the initial and the target body etc.) and a reward can be given according to the fulfillment of the optimization objective(s) and constraints. ANNs can be used to implement spacecraft steering strategies.

Evolutionary Neurocontrol. For the work described here, feedforward ANNs with a sigmoid neural transfer function have been used. Such an ANN can be considered as a continuous parameterized function (called network function) $\mathsf{N}_{\boldsymbol{w}} : \mathcal{X} \subseteq \mathbb{R}^{n_i} \rightarrow \mathcal{Y} \subseteq (0,1)^{n_o}$ that maps from an n_i-dimensional input space $\mathcal{X}$ onto an n_o-dimensional output space $\mathcal{Y}$. The parameter set $\boldsymbol{w} = \{w_1, \ldots, w_{n_w}\}$ of the network function comprises the n_w internal parameters of the ANN, i.e., the weights of the neuron connections and the biases of the neurons. ANNs have already been successfully applied as neurocontrollers (NCs) for reinforcement learning problems [7]. The most simple way to apply an ANN for controlling a dynamical system is by letting the ANN provide the control $\boldsymbol{u}(\bar{t}) = \boldsymbol{Y}(\bar{t}) \in \mathcal{Y}$ from some input $\boldsymbol{X}(\bar{t}) \in \mathcal{X}$ that contains the relevant information for the control task. The NC's behavior is completely characterized by its network function $\mathsf{N}_{\boldsymbol{w}}$ (that is – for a given network topology – again completely characterized by its parameter set $\boldsymbol{w}$). Learning algorithms that rely on a training set – like backpropagation – fail when the correct output for a given input is not known, as it is the case for delayed reinforcement learning problems. EAs can be employed for searching $\mathsf{N}^\star$ because $\boldsymbol{w}$ can be mapped onto a real-valued string $\boldsymbol{c}$ (also called chromosome or individual) that provides an equivalent description of a network function. If an EA is already employed for the optimization of the NC parameters, it is manifest to use it also for the co-optimization of the initial conditions. This way, the initial conditions are made explicitly part of the optimization problem.

Neurocontroller Input and Output. Two fundamental questions arise concerning the application of a NC for spacecraft steering, what *input* the NC should get (or what the NC should *know* to steer the spacecraft) and what *output* the NC should give (or what the NC should *do* to steer the spacecraft). To be robust, a spacecraft steering strategy should be time-independent: to determine the currently optimal spacecraft control $\boldsymbol{u}(\bar{t}_i)$, the spacecraft steering strategy should have to know – at *any* time step $\bar{t}_i$ – only the current spacecraft state $\boldsymbol{x}_{\mathrm{SC}}(\bar{t}_i)$ and the

[1] This evaluation is analogous to the cost function in optimal control theory. To emphasize this fact, it will also be denoted by the symbol J.

current target state $\boldsymbol{x}_{\mathrm{T}}(\bar{t}_i)$, hence $\mathsf{S} : \{(\boldsymbol{x}_{\mathrm{SC}}, \boldsymbol{x}_{\mathrm{T}})\} \mapsto \{\boldsymbol{u}\}$. If a propulsion system other than a solar sail is employed, the current propellant mass $m_{\mathrm{P}}(\bar{t}_i)$ might be considered as an additional input. The number of potential input sets, however, is still large because $\boldsymbol{x}_{\mathrm{SC}}$ and $\boldsymbol{x}_{\mathrm{T}}$ may be given in coordinates of any reference frame. At any time step $\bar{t}_i$, each output neuron $j \in \{1, \ldots, n_o\}$ gives a value $Y_j(\bar{t}_i) \in (0, 1)$. The number of potential output sets is also large because there are many alternatives to define $\boldsymbol{u}$, and to calculate $\boldsymbol{u}$ from $\boldsymbol{Y}$. Good results have been obtained by letting the NC provide a three-dimensional output vector $\boldsymbol{d}'' \in (0, 1)^3$ from which a unit vector $\boldsymbol{d}$ is calculated.[2] $\boldsymbol{d}$ is interpreted as the desired thrust direction and is therefore called direction unit vector. For solar sailcraft, $\boldsymbol{u} = \boldsymbol{d}$, hence $\mathsf{S} : \{(\boldsymbol{x}_{\mathrm{SC}}, \boldsymbol{x}_{\mathrm{T}})\} \mapsto \{\boldsymbol{d}\}$. For electric spacecraft, the output must include the engine throttle $0 \leq \chi \leq 1$, so that $\boldsymbol{u} = (\boldsymbol{d}, \chi)$, hence $\mathsf{S} : \{(\boldsymbol{x}_{\mathrm{SC}}, \boldsymbol{x}_{\mathrm{T}}, m_{\mathrm{P}})\} \mapsto \{\boldsymbol{d}, \chi\}$.

Neurocontroller Fitness Assignment. In EAs, the optimality of a chromosome is rated by a fitness function[3] J. The optimality of a trajectory might be defined with respect to various primary objectives (e.g., transfer time or propellant consumption). When an ENC is used for trajectory optimization, the accuracies of the trajectory with respect to the final constraints must also be considered as secondary optimization objectives because they are not enforced otherwise. If, for example, the transfer time for a rendezvous is to be minimized, the fitness function must include the transfer time T, the final distance to the target $\Delta r_f = |\boldsymbol{r}_{\mathrm{T}}(\bar{t}_f) - \boldsymbol{r}_{\mathrm{SC}}(\bar{t}_f)|$, and the final relative velocity to the target $\Delta v_f = |\dot{\boldsymbol{r}}_{\mathrm{T}}(\bar{t}_f) - \dot{\boldsymbol{r}}_{\mathrm{SC}}(\bar{t}_f)|$, hence $J = J(T, \Delta r_f, \Delta v_f)$. If, for example, the propellant mass for a flyby problem is to be minimized, T and Δv_f are not relevant but the consumed propellant Δm_{P} must be included in the fitness function, hence $J = J(\Delta m_{\mathrm{P}}, \Delta r_f)$ in this case. Because the ENC unlikely generates a trajectory that satisfies the final constraints exactly ($\Delta r_f = 0\,\mathrm{m}$, $\Delta v_f = 0\,\mathrm{m/s}$), a maximum allowed distance $\Delta r_{f,\max}$ and a maximum allowed relative velocity $\Delta v_{f,\max}$ have to be defined. Because in the beginning of the search process most individuals do not meet the final constraints with the required accuracy, a maximum transfer time $T_{\max}$ must be defined for the numerical integration of the trajectory. For a detailed description of the NC fitness assignment, the reader is referred to [4].

Evolutionary Neurocontroller Design. This section details how an ENC may be applied for low-thrust trajectory optimization. To find the optimal spacecraft trajectory, the ENC method runs in two loops. Within the (inner) trajectory integration loop, a NC k steers the spacecraft according to its network function $\mathsf{N}_{\boldsymbol{w}_k}$ that is completely defined by its parameter set $\boldsymbol{w}_k$. The EA in the (outer) NC optimization loop holds a population $P = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_q\}$ of NC parameter sets including the additionally encoded initial conditions, and examines them for their suitability to generate an optimal trajectory. Within the trajectory integration loop, the NC takes the current spacecraft state $\boldsymbol{x}_{\mathrm{SC}}(\bar{t}_{i\in\{0,\ldots,\tau-1\}})$ and that of the target $\boldsymbol{x}_{\mathrm{T}}(\bar{t}_i)$ as input, and maps them onto some output. For electric spacecraft, the input includes the current propellant mass $m_{\mathrm{P}}(\bar{t}_i)$ and the output includes the current throttle $\chi(\bar{t}_i)$. The first three output values are interpreted as the components of $\boldsymbol{d}''(\bar{t}_i)$, from which the direction unit vector $\boldsymbol{d}(\bar{t}_i)$ is calculated. This way the spacecraft control $\boldsymbol{u}(\bar{t}_i)$ is calculated from the NC output. Then, $\boldsymbol{x}_{\mathrm{SC}}(\bar{t}_i)$ and $\boldsymbol{u}(\bar{t}_i)$ are inserted into the equations of motion and numerically integrated over one time step to yield $\boldsymbol{x}_{\mathrm{SC}}(\bar{t}_{i+1})$. The new state is fed back into the NC. The trajectory integration loop stops when the final constraints are met with sufficient accuracy or when a given time limit is reached ($\bar{t}_{i+1} = \bar{t}_{f,\max}$). Then, back in the NC optimization loop, the trajectory is rated by the EA's fitness function $J(\boldsymbol{c}_k)$. The fitness of $\boldsymbol{c}_k$ is crucial for its probability to reproduce and to create offspring. Under this selection pressure, the EA breeds more and more suitable steering strategies that generate better and better trajectories. Finally, the EA that is used within this

[2] via $\boldsymbol{d}' = 2\boldsymbol{d}'' - (1, 1, 1)^{\mathrm{T}} \in (-1, 1)^3$ and $\boldsymbol{d} = \boldsymbol{d}'/|\boldsymbol{d}'|$

[3] This fitness function is also analogous to the cost function in optimal control theory. To emphasize this fact, it will be denoted also by the symbol J.

work converges against a single steering strategy (by shrinking the solution space during the search process), which gives in the best case a near-globally optimal trajectory $\boldsymbol{x}^{\star}_{\mathrm{SC}}[t]$.

3. Results: Mercury Rendezvous With a Solar Sail

To assess the performance of of InTrance, it was used to re-calculate trajectories for problems found in the literature (henceforth called reference problems/trajectories). Due to the limitations of this extended abstract, results for only one example can be given, a Mercury rendezvous using solar sail propulsion.[4] For an ideal solar sail with a characteristic acceleration[5] of $0.55\,\mathrm{mm/s^2}$ a (locally) optimal trajectory was reported in [8, 9] using a LTOM. It launches from Earth on 15 Jan 03 and takes 665 days to rendezvous Mercury, inserting the solar sailcraft directly into an interplanetary trajectory with zero hyperbolic excess energy ($C_3 = |\dot{\boldsymbol{r}}_{\mathrm{SC}}(\bar{t}_0) - \dot{\boldsymbol{r}}_{\mathrm{Earth}}(\bar{t}_0)|^2 = 0\,\mathrm{km^2/s^2}$).

InTrance was run five times with different initial populations for the launch date of the reference trajectory (reference launch date). A neurocontroller with 12 input neurons, 1 hidden layer with 30 hidden neurons, and 3 output neurons was used, where the input neurons received the solar sailcraft state $\boldsymbol{x}_{\mathrm{SC}}$ and the target state $\boldsymbol{x}_{\mathrm{T}}$ in cartesian coordinates, and the output neurons defined the direction unit vector $\boldsymbol{d}$ (different input sets, different output sets, and different numbers of hidden neurons/layers have been tested, but the results are beyond the scope of this extended abstract; they can be found in [4]). The maximum transfer time was set to $T_{\max} = 600\,\mathrm{days}$. For discretization, this time interval was divided into $\tau = 600$ periods of equal length. The accuracy limit was set to $\Delta r_{f,\max} = 0.1{\cdot}10^6\,\mathrm{km}$ and $\Delta v_{f,\max} = 0.1\,\mathrm{km/s}$.[6] Based on preliminary experiments, the population size was set to $q = 50$. Fig. 1 shows the resulting trajectories for the five InTrance-runs. The best trajectory is 91 days faster than the reference trajectory, revealing that the latter is far from the global optimum. The achieved accuracy is well better than required. The small variation of the solutions (15 days) gives evidence for a good convergence behavior.

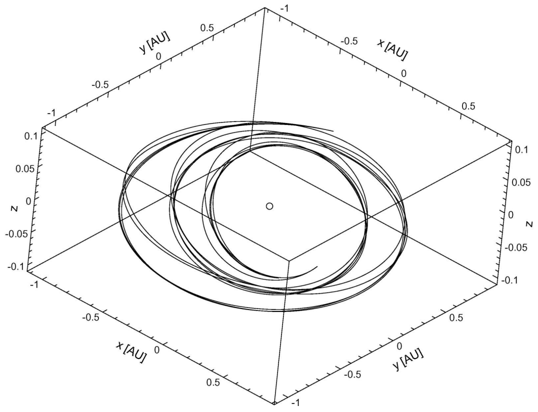


Figure 1. Mercury rendezvous trajectories (reference launch date, 5 different initial populations)

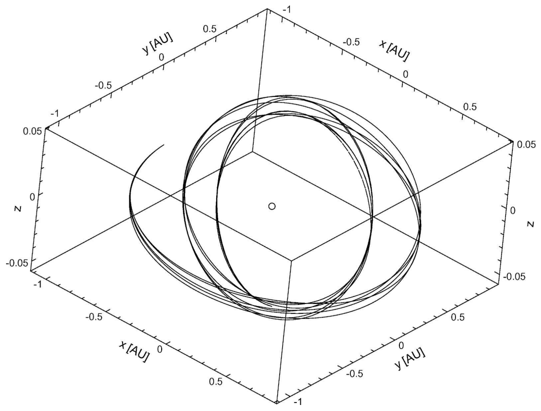


Figure 2. Mercury rendezvous trajectories (optimized launch date, 5 different initial populations)

To find the optimal launch date, InTrance was used to determine the shortest orbit transfer, which can be obtained by requiring only a rendezvous with Mercury's orbit, not with Mercury itself. The shortest orbit transfer found by five InTrance-runs takes $T = 510\,\mathrm{days}$. By scanning the planetary positions, it can be found that within a $1\,\mathrm{year}$-interval around the

[4] The results for further reference problems can be found in [4]. Novel applications of InTrance for low-thrust trajectory optimization can be found in [1–6].

[5] maximum acceleration of solar sailcraft at Earth distance from the sun

[6] $\Delta v_{f,\max} = 0.1\,\mathrm{m/s}$ was also used in [8, 9], whereas $\Delta r_{f,\max}$ is not given there.

reference launch date, the constellation of Earth and Mercury is most similar to that of the optimal orbit transfer solution for a launch on 27 Mar 03. Therefore, InTrance was run five times, being allowed to choose the launch date from the interval [26 Mar 03, 31 Mar 03]. The maximum transfer time was set again to $T_{\max} = 600\,\mathrm{days}$ (with $\tau = 600$). The small variation of the solutions shown in Fig. 2 (4 days) gives again evidence for a good convergence behavior. Taking 502 days to rendezvous Mercury, the best InTrance-trajectory is now 163 days faster than the reference trajectory, the accuracy again being well better than required.

4. Summary and Conclusions

Within the work described here, low-thrust trajectory optimization was attacked from the perspective of machine learning. A novel global method for spacecraft trajectory optimization, termed InTrance, was proposed. It fuses artificial neural networks and evolutionary algorithms into evolutionary neurocontrollers. The re-calculation of an exemplary Mercury rendezvous mission with a solar sail revealed that a reference trajectory, which was generated by a trajectory optimization expert using a local trajectory optimization method, is quite far from the global optimum. Using InTrance, the transfer time could be reduced considerably. InTrance runs without an initial guess and does not require the attendance of an expert in astrodynamics and optimal control theory. Being problem-independent, its application field may be extended to a variety of other optimal control problems.

References

[1] B. Dachwald. Minimum transfer times for nonperfectly reflecting solar sailcraft. *Journal of Spacecraft and Rockets*, 41(4):693–695.

[2] B. Dachwald. Optimal solar sail trajectories for missions to the outer solar system. *Journal of Guidance, Control, and Dynamics*. in press.

[3] B. Dachwald. Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol. *Journal of Guidance, Control, and Dynamics*, 27(1):66–72.

[4] B. Dachwald. *Low-Thrust Trajectory Optimization and Interplanetary Mission Analysis Using Evolutionary Neurocontrol*. Doctoral thesis, Universität der Bundeswehr München; Fakultät für Luft- und Raumfahrttechnik, 2004.

[5] B. Dachwald. Optimization of very-low-thrust trajectories using evolutionary neurocontrol. *Acta Astronautica*, 57(2-8):175–185, 2005.

[6] B. Dachwald and W. Seboldt. Multiple near-earth asteroid rendezvous and sample return using first generation solar sailcraft. *Acta Astronautica*, 2005. in press.

[7] D. C. Dracopoulos. *Evolutionary Learning Algorithms for Neural Adaptive Control*. Perspectives in Neural Computing. Springer, Berlin, Heidelberg, New York, 1997.

[8] M. Leipold. *Solar Sail Mission Design*. Doctoral thesis, Lehrstuhl für Flugmechanik und Flugregelung; Technische Universität München, 1999. DLR-FB-2000-22.

[9] M. Leipold, W. Seboldt, S. Lingner, E. Borg, A. Herrmann, A. Pabsch, O. Wagner, and J. Brückner. Mercury sun-synchronous polar orbiter with a solar sail. *Acta Astronautica*, 39(1-4):143–151, 1996.

[10] W. Seboldt and B. Dachwald. Solar sailcraft of the first generation: Technology development. Bremen, Germany, September/October 2003. 54^{th} International Astronautical Congress. IAC-03-S.6.03.

[11] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, London, 1998.

Neutral Data Fitting from an Optimisation Viewpoint

Mirjam Dür[1] and Chris Tofallis[2]

[1] *Department of Mathematics, Darmstadt University of Technology, Schloßgartenstraße 7, D-64289 Darmstadt, Germany,* duer@mathematik.tu-darmstadt.de

[2] *Department of Management Systems, University of Hertfordshire Business School, Hatfield, Hertfordshire AL10 9AB, United Kingdom,* c.tofallis@herts.ac.uk

Abstract Suppose you are interested in estimating the slope of a line fitted to data points. How should you fit the line if you want to treat each variable on the same basis? Least squares regression is inappropriate here since it's purpose is the prediction of one of the variables. If the variables are switched it provides a different slope estimate. We present a method which gives a unique slope and line, and which is invariant to changes in units of measurement. We attempt to extend the approach to fitting a linear model to multiple variables. In contrast to least squares fitting our method requires the solution of a global optimisation problem.

Keywords: Data fitting, regression models, fractional programming.

1. Introduction

Multiple Neutral Data Fitting is a method to analyse the relationship between a number of variables alternative to the well known least squares estimation.

Least squares is so popular because it is so easily computed. From an optimisation standpoint an unconstrained convex quadratic optimisation problem has to be solved which is done by setting the partial derivatives of the objective equal to zero. This gives the famous normal equations which have a nice analytic solution.

Least squares regression does, however, suffer from several shortcomings: it does not possess several properties as listed in [7] that seem desirable for a data fitting method. For example, in order to apply least squares, the user needs to specify which variables are the independent variables, and which one is the dependent variable. A change in this setting will lead to a completely different least squares estimate. In practice, however, a distinction between explanatory and response variables is not always so easy.

Another deficiency of least squares fitting is an assumption in the underlying model that may be unrealistic in many situations: the model is based on the assumption that only the dependent variable is subject to measurement errors. The independent variables are assumed to be known exactly, a premise that is often not fulfilled.

Multiple Neutral Data Fitting is an approach that avoids these shortcomings. The basic idea is that a different criterion is chosen as objective of the optimisation problem: Instead of minimizing the sum of the squares of the residuals, we consider the deviations for each variable and multiply them. As a result, we get a slope estimate different from the least squares solution. The new estimate possesses nice theoretical properties, but comes with a higher computational cost when fitting a model to multiple variables. In contrast to least squares regression, our method requires the solution of a global optimisation problem.

More precisely, the objective in the Multiple Neutral Data Fitting method is a sum of absolute values of ratios. The number of ratios corresponds to the number of observations. The number of variables equals the number of parameters needed to describe the fitted line (or, more generally, the fitted hyperplane). In practical applications, this number is usually not too big, which is advantageous for our global optimisation problem.

The Multiple Neutral Data Fitting approach has been alluded to by several statisticians throughout the last century. Some properties of the resulting fitted line have been shown, but generally speaking, the method has not been explored as an optimisation.

In the talk, we analyse Multiple Neutral Data Fitting as a global optimisation problem. We discuss in detail the two dimensional case (i.e., the case of only one independent variable). We then generalize to higher dimensions and propose an algorithm to solve the Multiple Neutral Data Fitting optimisation problem.

2. Underlying Models

Let $(X_i, Y_i), i = 1, \dots, n$ be a set of data points and suppose we wish to estimate the slope of a line fitted to these data. We will illustrate our ideas with the following tiny sample set of ten data points $(X_i, Y_i) \in \mathbb{R} \times \mathbb{R}$:

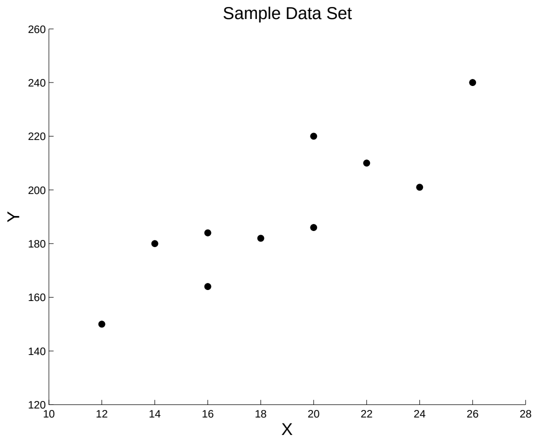


Figure 1. A tiny data set for illustration purposes.

2.1 The Least Squares Model

When using least squares regression, we usually assume the following relationship applies to the data, cf. [3]:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad i = 1, \dots, n, \tag{1}$$

that is, we have an explanatory variable X_i which is assumed to be measured exactly, and we have a response variable Y_i which is a combination of the true $\beta_0 + \beta_1 X_i$ plus a random error ε_i. The distribution of the ε_i does not depend on the β's.

The sum of the squares of deviations from the true line is

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2,$$

and the least squares regression line is the line with minimal S, cf. Fig2.

Since this amounts to solving an unconstrained convex quadratic optimisation problem, the estimates b_0 for β_0 and b_1 for β_1 are easily computed by setting the partial derivatives $\frac{\partial S}{\partial \beta_0}$ and

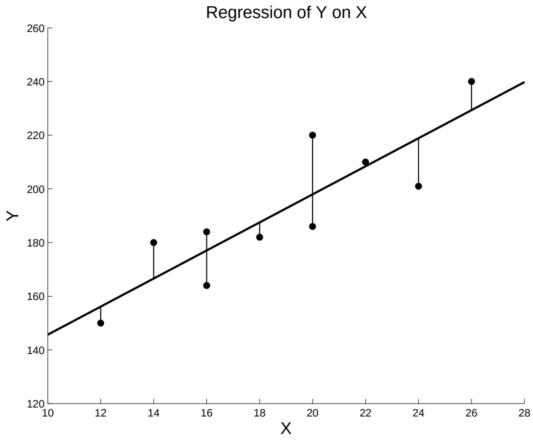


Figure 2. The least squares model minimizes the sum of the squares of the deviations.

$\frac{\partial S}{\partial \beta_1}$ equal to zero. This leads to the well known normal equations

$$b_0 n + b_1 \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} Y_i \qquad \text{and} \qquad b_0 \sum_{i=1}^{n} X_i + b_1 \sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} X_i Y_i$$

from which we get analytical expressions for b_0 and b_1.

Observe that switching the variables will lead to a different solution and therefore to a different regression line. For the sample data, this is illustrated in Fig. 3.

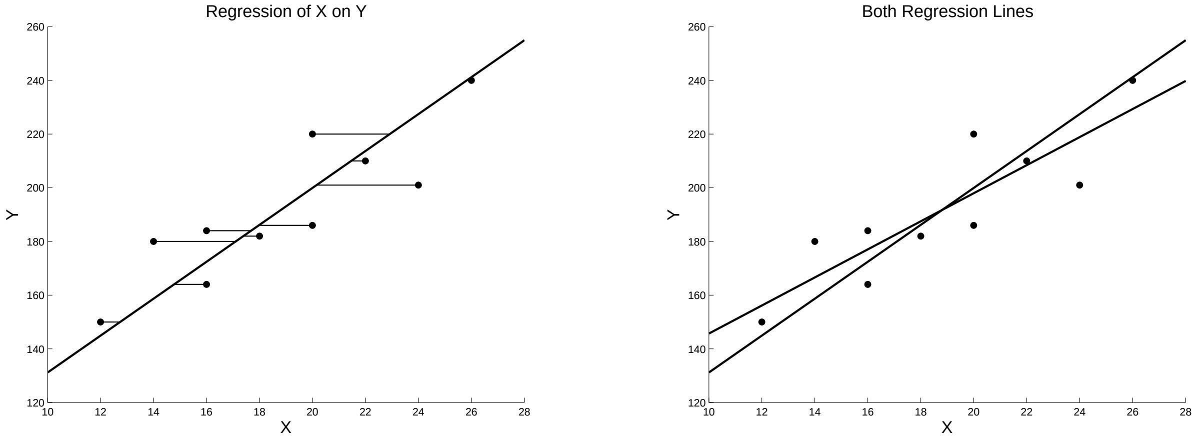


Figure 3. Left: treating X as the response variable and Y as explanatory variable gives deviations in the x-coordinate. Minimizing the sum of the squares of these deviations yields a least squares estimate different from the one in Fig. 2. Right: the two different least squares regression lines.

In higher dimensional settings the situation is similar: Assume we have data points $X_i \in \mathbb{R}^k$ $(i = 1, \ldots, n)$, then there are k ways to choose the response variable, and thus k different least squares regression hyperplanes.

The sensitivity of least squares regression to the choice of the response variable is one reason why it is desirable to consider alternatives.

2.2 All Variables subject to Errors

Next, assume both X_i and Y_i are subject to random errors, δ_i and ε_i, respectively (cf. also [4]):

$$\begin{aligned} Y_i &= \eta_i + \varepsilon_i, \\ X_i &= \xi_i + \delta_i. \end{aligned}$$

In this model, η_i and ξ_i are the true but unobserved values. A linear relationship $\eta_i = \beta_0 + \beta_1 \xi_i$ between them translates to

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i^*, \qquad \text{with} \qquad \varepsilon_i^* = \varepsilon_i - \beta_1 \delta_i. \tag{2}$$

So we have an equation similar to (1), but now the distribution of the error terms ε_i^* depends on β_1 which makes the situation much more complicated from the statistical viewpoint. Therefore, least squares regression is no longer an appropriate model:

Under the assumption of normally distributed errors ($\varepsilon_i \sim N(0, \sigma^2)$), the least squares estimate b_1 of β_1 is the maximum likelihood estimate for model (1), and it provides the minimum variance linear unbiased estimator.

For model (2), the least squares estimate has none of these properties. Since model (2) is often more realistic in practical situations, this is a second reason why we need alternative data fitting methods.

3. Neutral Data Fitting as an Optimisation Problem

As outlined in [11], a possibility to overcome the shortcomings of least squares regression lies in the following approach: Consider the deviations from the fitted line *in each coordinate direction* and multiply them together. In two dimensions, this looks as follows: For every data point, we multiply the vertical (y) deviation with the horizontal (x) deviation. This gives twice the area of the right-angled triangle formed by the fitted line and the two lines passing through the data point parallel to the axes, cf. Fig. 4. Our fitting criterion is then to minimize the sum of these triangular areas, i.e. the sum of the products of the deviations. For this reason the approach is called *method of least triangles* in [1].

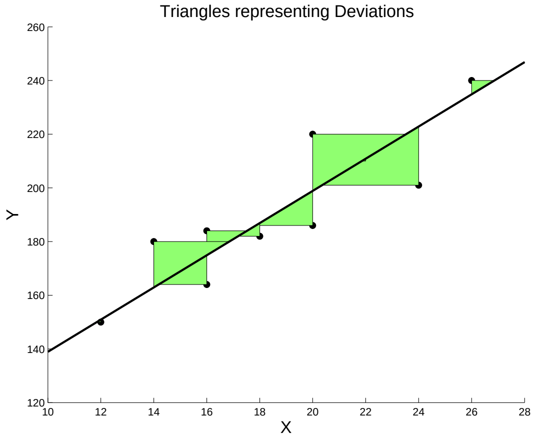


Figure 4. The product of the deviations of each data point is proportional to the area of the shaded triangle.

Consider the two variable case in more detail: Given n data points $(X_i, Y_i) \in \mathbb{R} \times \mathbb{R}$, assume we would like to fit a line of the form

$$ax + by = c.$$

For each data point (X_i, Y_i), the deviation is the x-direction is given by

$$\left|X_i - \frac{c - bY_i}{a}\right| = \left|\frac{aX_i + bY_i - c}{a}\right|,$$

whereas the deviation in the y-direction is

$$\left|Y_i - \frac{c - aX_i}{b}\right| = \left|\frac{aX_i + bY_i - c}{b}\right|.$$

Hence the product of the deviations associated to data point (X_i, Y_i) equals $\left|\frac{(aX_i+bY_i-c)^2}{ab}\right|$. Summing over all data points yields the objective function, and we are faced with the following optimisation problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \left|\frac{(aX_i + bY_i - c)^2}{ab}\right| \\ \text{s.t.} \quad & a, b, c \in \mathbb{R}. \end{aligned} \tag{3}$$

In the higher dimensional case, i.e. in the presence of variables $X_i \in \mathbb{R}^k$ $(i = 1, \dots, n)$, the model generalizes in the obvious way, cf. also [10] and [6]: The hyperplane we want to fit is then of the form

$$\sum_{j=1}^{k} a_j x_j = c,$$

deviation of a data point X_i from the hyperplane in coordinate direction j amounts to

$$\left|\frac{\sum_{j=1}^{k} a_j (X_i)_j - c}{a_j}\right|,$$

and the optimisation problem becomes

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \left|\frac{\left(\sum_{j=1}^{k} a_j (X_i)_j - c\right)^k}{\prod_{j=1}^{k} a_j}\right| \\ \text{s.t.} \quad & a \in \mathbb{R}^k, c \in \mathbb{R}. \end{aligned} \tag{4}$$

Note that the dimension of the optimisation problem is $k + 1$, not the number n of data points. Usually, $k + 1 \ll n$ in practical applications. In some cases the hyperplane is known not to pass through the origin, such that it is possible to set $c = 1$, thus reducing the dimension by one.

Problem (3) falls into a class of nonlinear optimisation problems called fractional programming problems which have received a considerable amount of interest in the last decades (for a survey cf. [8]). While problems involving ratios of linear functions are well understood (see [5] or [9] and references therein), and certain problems involving ratios of linear functions with absolute values have been treated in [2], to our knowledge the special structure of (3) has not yet been studied.

References

[1] Barker, F., Soh, Y.C., and Evans, R.J. (1988). Properties of the Geometric Mean Functional Relationship. *Biometrics* 44, 279–281.

[2] Chadha, S. S. (2002). Fractional Programming with Absolute-value Functions. *European Journal of Operational Research* 141, 233–238.

[3] Draper, Norman R. and Smith, Harry (1998). *Applied Regression Analysis, 3rd. ed.* New York: Wiley.

[4] Draper, Norman R. and Yang, Yonghong (1997). Generalization of the Geometric Mean Functional Relationship. *Computational Statistics and Data Analysis* 23, 355–372.

[5] Dür, Mirjam, Horst, Reiner and Thoai, Nguyen Van (2001). Solving Sum-of-Ratios Fractional Programs using Efficient Points. *Optimization* 49, 447–466.

[6] Goodman, T.N.T. and Tofallis, C. (2003). Neutral Data Fitting in Two and Three Dimensions. *University of Hertfordshire Business School Working Paper.*

[7] Samuelson, Paul A. (1942). A Note on Alternative Regressions. *Econometrica* 10(1), 80–83.

[8] Schaible, Siegfried (1995). Fractional Programming. In: Reiner Horst and Panos M. Pardalos (eds.), *Handbook of Global Optimization* 495–608. Dordrecht: Kluwer Academic Publishers.

[9] Schaible, Siegfried and Shi, Jianming (2003). Fractional Programming: The Sum-of-Ratios Case, *Optimization Methods and Software* 18, 219–229.

[10] Tofallis, Chris (2002). Model Fitting for Multiple Variables by Minimising the Geometric Mean Deviation. In: S. Van Huffel and P. Lemming (eds.), *Total Least Squares and Errors-in-variables Modeling: Algorithms, Analysis and Applications* 261–267. Dordrecht: Kluwer Academic Publishers.

[11] Tofallis, Chris (2003). Multiple Neutral Data Fitting. *Annals of Operations Research* 124, 69–79.

Constraint-like methods for obtaining an outer approximation of the efficient set of nonlinear biobjective optimization problems[*]

José Fernández[1] and Boglárka Tóth[1,†]

[1] *Dpt. Statistics and Operations Research, University of Murcia, Spain,* {josefdez,boglarka}@um.es

Abstract Two methods for obtaining an outer approximation of the efficient set of nonlinear biobjective problems are presented. They are based on the well-known "constraint method" and allow to obtain a superset of the efficient set by solving a finite number of single objective constraint problems. Computational experience is reported.

Keywords: nonlinear biobjective problems, efficient set, outer approximation, interval analysis, constraint method.

1. The problem

Let $f_1, f_2 : \mathbb{R}^n \longrightarrow \mathbb{R}$ be two real functions, and consider the problem

$$\begin{array}{ll} \min & \{f_1(x), f_2(x)\} \\ \text{s.t.} & x \in S \end{array} \tag{1}$$

We will denote the vector of objective functions by $f(x) = (f_1(x), f_2(x))$, and the image of the feasible region by $Z = f(S)$.

Definition 1. *A feasible vector $x^* \in S$ is said to be* efficient *iff there does not exist another feasible vector $x \in S$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, 2$, and $f_j(x) < f_j(x^*)$ for at least one index j. The set S_E of all the efficient points is called the* efficient set.

Efficiency is defined in the decision space. The corresponding definition in the criterion space is as follows.

Definition 2. *An objective vector $z^* = f(x^*) \in Z$ is said to be* nondominated *(or also* efficient*) iff x^* is efficient. The set of all nondominated vectors will be denoted by Z_N.*

Other related and widely used concept is the weak efficiency.

Definition 3. *A feasible vector $x^* \in S$ is said to be* weakly efficient *iff there does not exist another feasible vector $x \in S$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, 2$.*

In many problems it is important to obtain the whole efficient set. However, to the extent of our knowledge, no method has been proposed in the literature with that purpose. In fact, even

[*] This paper has been supported by the Ministry of Science and Technology of Spain under the research project BEC2002-01026, in part financed by the European Regional Development Fund (ERDF).

[†] On leave from the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged, Hungary.

obtaining a single efficient point can be a difficult task. Some authors have proposed to present to the decision-maker a "representative set" of efficient points which suitably represent the whole efficient set, either by modifying the definition of efficiency [2] or by selecting a finite set of efficient points with the criteria of coverage, uniformity and cardinality as quality measures [1, 9]. The approach in this paper is completely different. Instead of offering a subset of the efficient set to the decision-maker, we give him a superset containing it, whose accuracy is fixed before-hand (see also [3]).

2. A constraint-like method

Due to the possible non-convexity (and non-concavity) of the objective functions a global optimization technique is required in order to obtain efficient points. In this section, we present a constraint-like method which uses interval analysis tools. For details about interval analysis the interested reader is referred to [5, 6, 8].

2.1 Preliminaries

The constraint method. The constraint method was introduced in [4]. One of the objective functions, say f_1, is selected to be minimized and the other one, f_2, is converted into a constraint by setting an upper bound f_2^{ub} to it. The single objective problem to be solved, called *constraint problem*, is then

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{ub} \\ & x \in S \end{array} \tag{2}$$

The goodness of the constraint method can be seen in the following results.

Theorem 4. *The solution of the constraint problem (2) is weakly efficient.*

Theorem 5. *A feasible vector $x^* \in S$ is efficient if and only if it is a solution of the constraint problems*

$$\begin{array}{ll} \min & f_1(x) \\ s.t. & f_2(x) \leq f_2(x^*) \\ & x \in S \end{array} \quad \textit{and} \quad \begin{array}{ll} \min & f_2(x) \\ s.t. & f_1(x) \leq f_1(x^*) \\ & x \in S \end{array} . \tag{3}$$

Instead of having to solve those two constraint problems in (3), efficiency can also be guaranteed with only one of them, provided that it has a unique solution, as the following theorem shows.

Theorem 6. *A feasible vector $x^* \in S$ is efficient if it is a unique solution of any of the constraint problems in* (3)*.*

From the previous theorems it follows that it is possible to find all the efficient solutions of any multiobjective optimization problem by the constraint method. However, we need to solve one or two problems to find 'one' efficient solution, which means that if the efficient set is not a discrete set (as it is usually the case in continuous multiobjective optimization) then the method is not practical for finding the complete efficient set.

Obtaining a region of δ-optimality. Consider the single-objective problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^n \end{array} \tag{4}$$

We want to find the region R_δ of δ-optimality of (4), i.e. the set of feasible points whose objective value does not exceed the optimal value of (4) by more than a fraction δ ($\delta > 0$). In formula, if we denote by f^* the optimal value of (4),

$$R_\delta = \{x \in S : f(x) - f^* \leq \delta \cdot |f^*|\}.$$

In [7], a branch-and-bound method for obtaining the region of δ-optimality (of a general continuous location problem) is presented. In what follows we will refer to it as the *δ-opt algorithm*. It consists of two phases. The first one entails the determination of the optimal objective value of (4) up to a prespecified relative precision ε. The second phase consists of the determination of R_δ, up to a prespecified precision η. The output of the algorithm is two lists of subsets, $\mathcal{L}_3$ and $\mathcal{L}_4$. The union of the subsets in the first list gives an inner approximation of R_δ, whereas the union of the subsets in both $\mathcal{L}_3$ and $\mathcal{L}_4$ gives an outer approximation OR_δ of R_δ, guaranteed to lie entirely within $R_{\delta+\eta(1+\delta)}$, i.e., $OR_\delta = \bigcup_{Q\in\mathcal{L}_3\cup\mathcal{L}_4} Q\cap S$ satisfies that $R_\delta \subseteq OR_\delta \subseteq R_{\delta+\eta(1+\delta)}$.

The algorithm can be easily carried out with the help of *Interval Analysis*. In that case, the subsets Q will be boxes (multidimensional intervals), and the required bounds on them can be obtained automatically with the use of inclusion functions.

2.2 The constraint problems

Going back to the determination of the efficient set of (1), we will use constraint problems of the form[1]

$$(P_i)\quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \le f_2^{(i)} \\ & x \in S \end{array} \tag{5}$$

where $f_2^{(i)}$ is a given constant defined below. Let $\hat{x}^{(i)}$ denote an optimal solution of (P_i), and let $R_\delta^{(i)}$ be the region of δ-optimality of (P_i), that is,

$$R_\delta^{(i)} = \{x \in S : f_2(x) \le f_2^{(i)}, f_1(x) - f_1(\hat{x}^{(i)}) \le \delta \cdot |f_1(\hat{x}^{(i)})|\}.$$

In the first problem that we will consider, (P_0), we set $f_2^{(0)} = +\infty$. Thus, problem (P_0) is in fact the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

Once we have solved problem (P_i) and have obtained an outer approximation of $R_\delta^{(i)}$ with the help of the δ-opt algorithm mentioned in the previous subsection, the constant $f_2^{(i+1)}$ for the next problem (P_{i+1}) is given by (see Figure 1)

$$f_2^{(i+1)} = \min\{U_2(Q) : Q \in \mathcal{L}_3 \cup \mathcal{L}_4, Q \cap R_\delta^{(i)} \ne \emptyset\} \tag{6}$$

where $U_2(Q)$ is an upper bound on all objective values of f_2 at Q. However, from a computational point of view, it can be better to set

$$f_2^{(i+1)} = \min\{U_2(Q) : Q \in \mathcal{L}_3, Q \cap S \ne \emptyset\} \tag{7}$$

although this is a worst (higher) value than the one obtained with (6). Using (7) we only have to check whether a box Q in $\mathcal{L}_3$ contains at least one feasible point. If so, we take that box into account for calculating the minimum in (7).

Let us denote by $Q_N^{(i)}$ the subset at which the previous minimum (either (6) or (7)) is attained, i.e, $f_2^{(i+1)} = U_2(Q_N^{(i)})$.

Lemma 7. $f_1(\hat{x}^{(i+1)}) \le f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$.

[1] We always minimize f_1 subject to a constraint on f_2, but a similar process can be developed interchanging the functions.

2.3 The method

The method that we propose to obtain an outer approximation of the efficient set of (1) is the following (see Figure 1):

Algorithm 1.

```
$i \leftarrow 0$, $f_2^{(0)} \leftarrow +\infty$.
While $(P_i)$ is feasible
    Solve $(P_i)$ and obtain an outer approximation $OR_\delta^{(i)}$ of $R_\delta^{(i)}$ using the $\delta$-opt algorithm.
    Calculate $f_2^{(i+1)}$ as given by (6) or (7).
    If $f_2^{(i+1)} \not< f_2^{(i)}$ then
        $j \leftarrow 0$.
        Repeat
            $j \leftarrow j+1$.
            Solve the problem

                $(P_{i+j})$   min   $f_1(x)$
                              s.t.  $f_2(x) \leq f_2^{(i)}$
                                    $x \in S$
                                    $f_1(x) \geq f_1(\hat{x}^{(i)}) + j \cdot \delta |f_1(\hat{x}^{(i)})|$        (8)

            and obtain an outer approximation $OR_\delta^{(i+j)}$ of $R_\delta^{(i+j)}$.
            Calculate $f_2^{(i+j+1)}$ as given by (6) or (7).
        until $f_2^{(i+j+1)} < f_2^{(i)}$ or the problem $(P_{i+j})$ is infeasible.
        If $f_2^{(i+j+1)} < f_2^{(i)}$ then set $i \leftarrow i+j+1$.
        If the problem $(P_{i+j})$ is infeasible then set $i \leftarrow i+j$ and break
    else $i \leftarrow i+1$.
$\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$ contains the efficient set of (1).
```

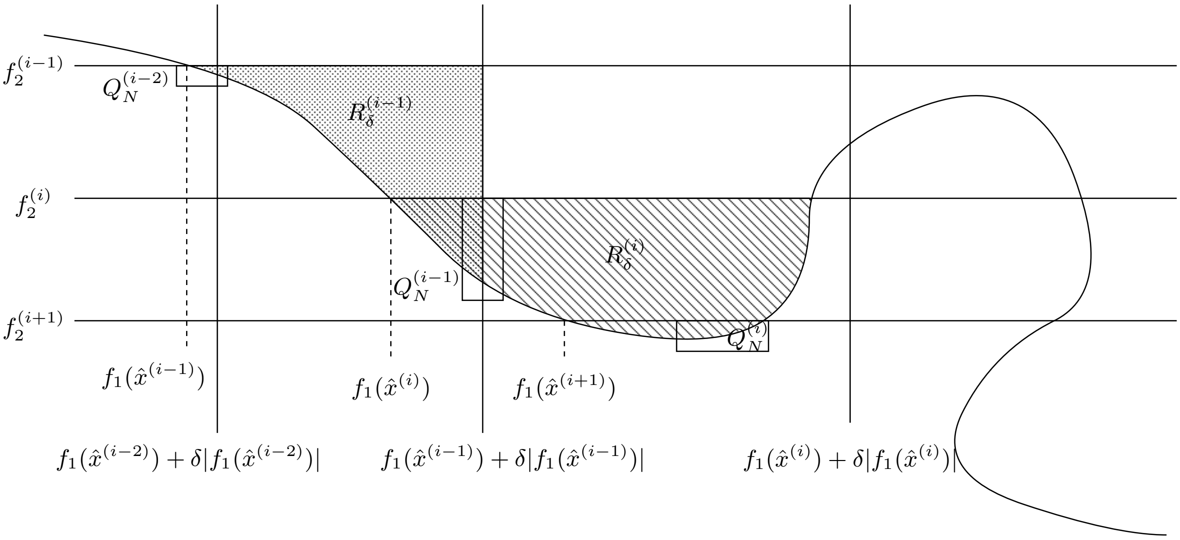


Figure 1. Image space of a biobjective problem using Algorithm 1. For the shake of clarity, for each problem (P_i) we have only drawn one of the boxes forming $OR_\delta^{(i)}$, namely, $Q_N^{(i)}$. The grey region is $R_\delta^{(i-1)}$, and the dotted region is $R_\delta^{(i)}$.

Theorem 8. *Algorithm 1 obtains all the efficient solutions of (1).*

Notice that we need problems of type (8) to avoid that the algorithm gets stuck when $f_2^{(i+1)} > f_2^{(i)}$. This may happen when the efficient set is not connected and the 'jump' (along the abscisas axis) is greater than $\delta|f_1(\hat{x}^{(i)})|$ (see Figure 1; without problems of type (8) Algorithm 1 will get stuck after solving problem (P_{i+1})) or when there is a continuum of weakly efficient points with the same f_2-value (i.e., in the image space the weakly efficient points form a segment parallel to the abscisas axis, being the length of that segment greater than $\delta|f_1(\hat{x}^{(i)})|$.)

3. Another constraint-like method

In the previous constraint-like method, in addition to constraint problems of the form (5) we also needed problems of the form (8) to be able to guarantee the finiteness of the algorithm. In this section we present another constraint-like method which only uses one type of constraint problems, a simplified version of the one in (8).

3.1 The constraint problems

The first problem that we will consider, (P_0), is again the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

Once we have solved problem (P_i) and have obtained a minimizer point $\hat{x}^{(i)}$ and an outer approximation of $R_\delta^{(i)}$ with the help of the δ-opt algorithm, we compute

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{U_2(Q) : Q \in \mathcal{L}_3 \cup \mathcal{L}_4, Q \cap R_\delta^{(i)} \neq \emptyset\}\} \tag{9}$$

or

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{U_2(Q) : Q \in \mathcal{L}_3, Q \cap S \neq \emptyset\}\} \tag{10}$$

Notice that $f_2^{(0)} = +\infty$. The next problem to be solved is

$$(P_{i+1}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i+1)} \\ & f_1(x) \geq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})| \\ & x \in S \end{array} \tag{11}$$

3.2 The method

The method that we propose to obtain an outer approximation of the efficient set of (1) is the following (see Figure 2):

Algorithm 2.

```
$i \leftarrow 0$, $f_2^{(0)} \leftarrow +\infty$.
While $(P_i)$ (as described in (11)) is feasible
    Solve $(P_i)$ and obtain an outer approximation $OR_\delta^{(i)}$ of $R_\delta^{(i)}$. Let $\hat{x}^{(i)}$ denote an
    optimal solution of $(P_i)$.
    Calculate $f_2^{(i+1)}$ as given by (9) or (10).
    $i \leftarrow i + 1$.
$\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$ contains the efficient set of (1).
```

Theorem 9. *Algorithm 2 obtains all the efficient solutions of (1).*

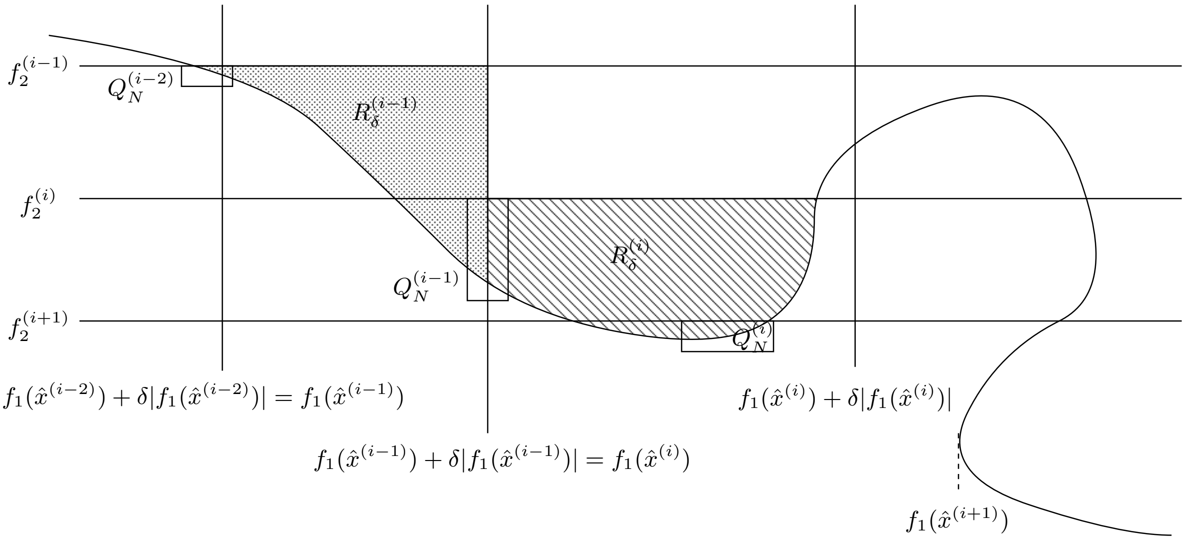

$f_2^{(i-1)}$
$Q_N^{(i-2)}$
$R_\delta^{(i-1)}$
$f_2^{(i)}$
$R_\delta^{(i)}$
$Q_N^{(i-1)}$
$f_2^{(i+1)}$
$Q_N^{(i)}$
$f_1(\hat{x}^{(i-2)}) + \delta|f_1(\hat{x}^{(i-2)})| = f_1(\hat{x}^{(i-1)})$
$f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$
$f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| = f_1(\hat{x}^{(i)})$
$f_1(\hat{x}^{(i+1)})$

Figure 2. Image space of a biobjective problem using Algorithm 2. The grey region is $R_\delta^{(i-1)}$, and the dotted region is $R_\delta^{(i)}$.

References

[1] Benson H. and Sayin S. (1997) Towards finding global representations of the efficient set in multiple objective mathematical programming, *Naval Research Logistics* 44: 47–67.

[2] Carrizosa E., Conde E. and Romero-Morales D. (1997), Location of a semiobnoxious facility. A biobjective approach, in *Advances in multiple objective and goal programming 1996*, Springer, 338–346.

[3] Fernández J., Tóth B., Plastria F. and Pelegrín B. (2004), Reconciling franchisor and franchisee: a planar biobjective competitive location and design model, submitted for publication.

[4] Haimes Y.Y., Lasdon L.S. and Wismer D.A. (1971) On a bicriterion formulation of the problems of integrated system identification ans system optimization, *IEEE Transactions on System, Man and Cybernetics* 1: 296-297.

[5] Hansen E. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.

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

[7] Plastria F. (1992), GBSSS: The Generalized Big Square Small Square Method for Planar Single-Facility Location, *European Journal of Operational Research* 62: 163–174.

[8] Ratschek H. and Rokne J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester.

[9] Sayin S. (2000) Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming, *Mathematical Programming, Ser. A*, 87: 543–560.

Application of interval arithmetics for exploring feasibility of extractive distillation variants

Erika R. Frits,[1,2] Ali Baharev,[1] Zoltán Lelkes,[1] Mihály Markót,[3] Zsolt Fonyó,[1,2] Endre Rév,[1,2] and Tibor Csendes[4]

[1] *Budapest Univ. Techn. Econ., Dept. Chem. Eng., Budapest, H-1521, Hungary,* ufo@mail.bme.hu

[2] *Bud. Univ. Techn. Econ. - Hung. Acad. Sci. Res. Group Techn. Chem., Budapest, H-1521, Hungary,* efrits@mail.bme.hu

[3] *European Space Agency, Advanced Concepts Team, Keplerlaan 1, 2201 AZ Noordwijk, the Netherlands,* Mihaly.Csaba.Markot@esa.int

[4] *Univ. Szeged, Inst. Appl. Informatics, Pf. 652., Szeged, H-6701, Hungary,* csendes@inf.u-szeged.hu

Abstract Feasibility study of extractive distillation variants is a successful application area of interval arithmetics based reliable computation of real function zeroes and global extrema. Feasibility of these processes can be well estimated by studying a differential equation system coupled with algebraic equations. Existence and location of the singular points play a crutial role in feasibility of these processes.

From mathematical point of view, the main problem consists of reliably finding the existence and loci of all the singular points in a given domain, and determining the bifurcations of the phase map. Both the singular points and the bifurcations have been reliably found by applying an interval-arithmetic based branch and bound optimization algorithm.

The mathematical model is shown, and the capability of the method is demonstrated on example results related to the separation of acetone – methanol mixture (forming minimum boiling azeotrope) with water as a heavy boiling entrainer, in both batch and continuous extractive distillation processes.

Keywords: interval arithmetics, branch and bound, phase map, singular point, bifurcation, extractive distillation, feasibility.

1. Introduction

Distillation as a process for separating liquid components is based on the different volatility of the components to be separated. Separation of close-volatility mixtures by conventional distillation is difficult and expensive. Separation of azeotrope-forming mixtures by conventional distillation is impossible. A mixture is called azeotrope-forming if the volatility of the components are identical at some composition, the so-called azeotrope.

Extractive distillation processes apply a third component, the so-called entrainer, to make the separation feasible. There are several variants of these processes according to the volatility relations between the components, and according to the several technical opportunities to be chosen.

Production of components in specified purity and acceptable recovery is possible only if the process pameters are kept inside a domain whose form and extension is not known in advance. This is called 'the feasible domain'. The process is called feasible if such a domain exists. The main process parameters are the so-called reflux ratio and feed ratio.

Feasibility study is engineering activity applied for finding a feasible domain of the process parameters. If such a domain is not found, the process is considered infeasible. However, finding a feasible set of process parameters is a difficult task, and a feasible process can easily be qualified to infeasible if inappropriate methodology is applied. Determining the border of the feasible domain is an even more difficult task.

As it has earlier been shown [1–3], feasibility of the many different process variants can be well estimated, and the feasible domain can be explored by applying approximating ordinary differential equations (initial value problems) for modelling the process. Feasibility of the studied process variants can be judged by studying the phase plots of these differential equations; here the phase curves are called 'profiles', namely rectifying profiles, stripping profiles, and extractive profiles. (These profiles describe the behaviour of different sections of the distillation column). Profile maps (i.e., phase maps) can take radically different shapes, depending on the process parameters.

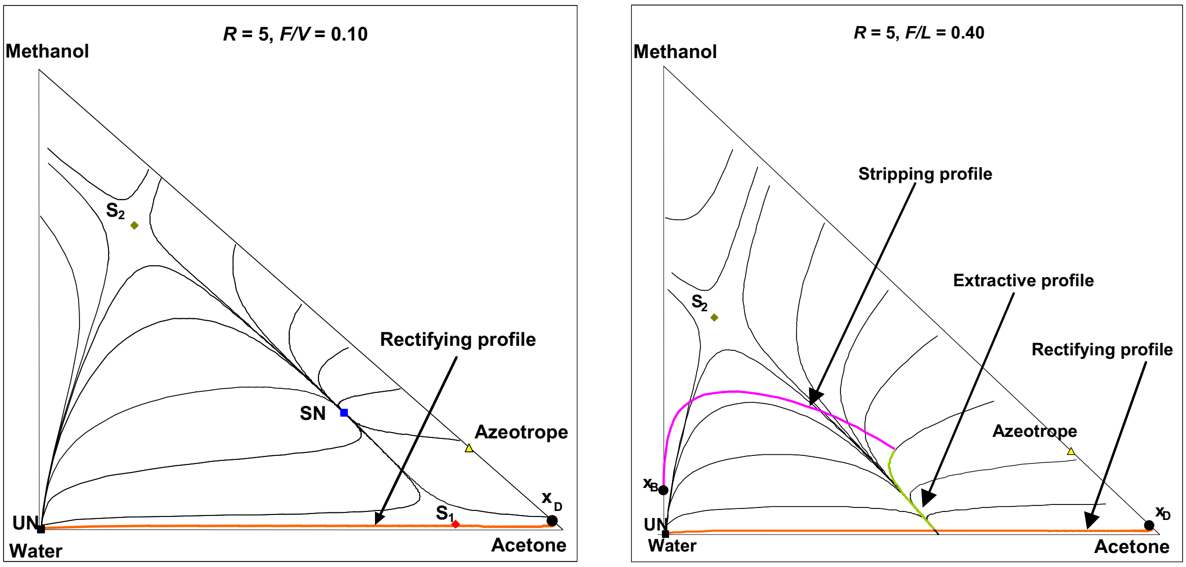


Figure 1. Extractive profile maps of continuous extractive distillation, minimum boiling azeotrope, heavy entrainer. a/ Left hand side: infeasible. b/ Right hand side: feasible

Existence and location of the singular points of the different profiles take a key role in judging feasiblity and exploring the feasible domain. For example, two different maps are shown in **Figures 1a** and **1b**, belonging to the same continuous extractive distillation process with different parameters. There is a saddle point S$_2$ in both figures. Two of the four separatrices of S$_2$, namely the pair in the direction of vertex Water - S$_2$ - edge of Acetone and Methanol, divides the triangle to a right hand side and a left hand side. Let us call this separatrix pair as 'boundary'. There is a stable node (SN) of the extractive profiles inside the composition domain, in **Figure 1a**. Extractive profiles started from the right of the boundary approach SN; they do not cross the rectifying profile drawn from $\mathbf{x}_D$, running along the Acetone-Water edge. As a result, this process is infeasible at $R = 5$ and $F/L = 0.1$. Stable node SN and saddle S$_1$ approach each other if the feed ratio F/L is increased. They meet over a critical feed ratio, and suddenly disappear. The map changes shape, and looks like that in **Figure 1b**. The extractive profiles are attracted to a point somewhere outside the triangle. All these extractive profiles cross the rectifying profile. As a result, the process is feasible at $R = 5$ and $F/L = 0.4$.

Finding stable nodes is relative easy by computing phase curves; finding the unstable nodes and saddle points is rather difficult. Even more difficult is to find the parameter values of bifurcations, i.e. the points at which the phase map suddenly changes shape. It is this bifurcation, however, what finally is looked for because this sudden change designates the border of feasible domain.

2. Mathematical formulation

In all the studied variants, the differential equations have the form of

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}h} = \frac{V}{L}\left(\boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{y}^*(\boldsymbol{x})\right) \tag{1}$$

where $\boldsymbol{x}$ is the array of independent variables, h is a running variable irrelevant for us, ratio V/L is a simple function of R and F/V or F/L, $\boldsymbol{y}^*(\boldsymbol{x})$ is a non-explicit function described below, and $\boldsymbol{y}(\boldsymbol{x})$ is an explicit function expressed in a general form as

$$\boldsymbol{y} = \left(\frac{R}{R+1} + \frac{F}{V}\right)\boldsymbol{x} + \frac{1}{R+1}\boldsymbol{x}_D - \frac{F}{V}\boldsymbol{x}_F \tag{2}$$

Here R is one of the main parameters of the process, and F/V is another one. $\boldsymbol{x}_D$ and $\boldsymbol{x}_F$ are known process specifications. Array $\boldsymbol{y}^*$ is determined according to the following system of equations (3 to 7):

$$y_i^* P = \gamma_i x_i p_i^\circ \quad (i = 1, 2, 3) \tag{3}$$

$$\lg p_i^\circ = A_i - \frac{B_i}{T - 273.14 + C_i} \quad (i = 1, 2, 3) \tag{4}$$

$$\ln \gamma_i = \frac{\sum_j \tau_{ji} G_{ji} x_j}{\sum_l G_{li} x_l} + \sum_j \frac{x_j G_{ij}}{\sum_l G_{lj} x_j}\left(\tau_{ij} - \frac{\sum_n \tau_{nj} G_{nj} x_n}{\sum_l G_{lj} x_j}\right) \quad (i = 1, 2, 3) \tag{5}$$

$$\tau_{ij} = \frac{U_{ij}}{R_G T} \quad ; \quad G_{ij} = \exp\left(-\alpha_{ij}\,\tau ij\right) \quad (i = 1, 2, 3) \tag{6}$$

$$\sum_{i=1}^{3} y_i^* = 1 \tag{7}$$

Here P is the systems pressure, R_G is a universal physical constant, A_i, B_i, C_i, U_{ij}, and α_{ij} are model parameters. p_i° and γ_i can be considered as functions of $\boldsymbol{x}$ and T; such a T is to be found for a given x that satisfies equation (7). The problem of finding the singular points can be formulated as solving the system of equations (2) to (8).

$$\boldsymbol{0} = \boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{y}^*(\boldsymbol{x}) \tag{8}$$

The problem of finding bifurcations can be formulated as follows. Differencial equation (1) is linearized around the singular point in a form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}h} = \boldsymbol{A}\boldsymbol{x} \tag{9}$$

The task is to find that parameters at which at least one of the eigenvalues of $\boldsymbol{A}$ has zero real part. In this problem class we encounter node and saddle singularities only; thus, bifurcations are simply signalled by zero determinant of $\boldsymbol{A}$. Unfortunately, entries of $\boldsymbol{A}$ (the *Jacobian*) cannot by directly determined because of the non-explicite nature of function $\boldsymbol{y}^*(\boldsymbol{x})$. This difficulty is solved by applying implicit function theorem. Computation of the partial derivatives of γ are also rather tedious.

3. Solution by interval optimization

In order to reliably find information about the existence of the characterisctic singular points and, in case of existence, finding their exact loci, we have implemented an interval arithmetic based optimization algorithm [4, 5]. Interval algebra provides us with a tool for determining real intervals that certainly include the range of the functions studied over a given domain. Utilizing this technique, all the roots of those functions, as well as all the minima of a real function, can be determined.

Criteria for singular points and bifurcation points can be described by algebraic equations. Instead of searching for zeroes of these equations, we formulated these problems as minimizing the sum of squares of the equation residues. The steps of the applied algorithm are the following:

```
Step 1. Let L be an empty list, the leading box B:=X (the total studied domain), and the
        iteration counter $k := 1$. Set the upper bound $F^u$ of the global minimum to be the
        upper bound of $F(\mathbf{X})$.
Step 2. Subdivide B into $s$ subsets. Evaluate the inclusion function $F(\mathbf{X})$ for all the new
        subintervals, and update the upper bound $F^u$ of the global minimum as the mini-
        mum of the old value and the upper bounds on the new subintervals.
Step 3. Add the new subintervals to the list L.
Step 4. Delete parts of the subintervals stored in L that cannot contain a global minimizer
        point.
Step 5. Set B to be that subinterval from list L which has the smallest lower bound on $F$, and
        remove the related item from the list.
Step 6. While termination criteria do not hold, let $k := k + 1$, and go to St. 2.
```

This is a branch-and-bound algorithm. Interval arithmetic and the interval extension of the used standard functions were realized by the PROFIL library [6]. The algorithm itself is a customized version of the global optimization procedure published in [7], and improved in several steps. The computational environment was a Pentium IV PC (1 Gbyte RAM and 1.4 MHz) with Linux operation system.

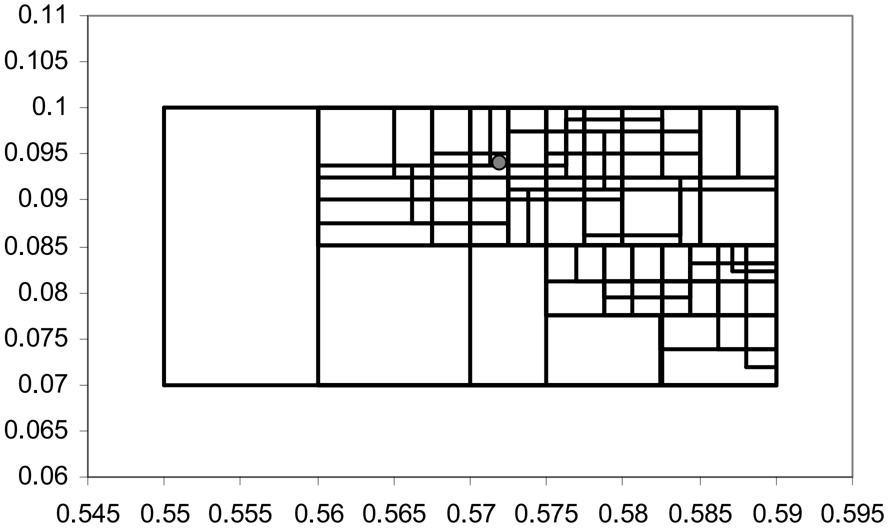


Figure 2. Subdivision of a subdomain in course of the search process

How a subregion is subdivided into smaller boxes is demonstrated in **Figure 2**. This picture taken at a moment, shows the yet living boxes, and how they are subdivided. The box containing a wide dot contains a solution.

4. Results

Feasible domain of extractive distillation of particular material systems have been explored. Here we show the results related to separation of the acetone – methanol mixture forming minimum boiling azeotrop, with the help of high boiling entrainer water, in batch extractive distillation process (batch rectifier device, charge in the boiling vessel, entrainer feed to the column) and continuous extractive distillation process (entrainer feed above the main feed).

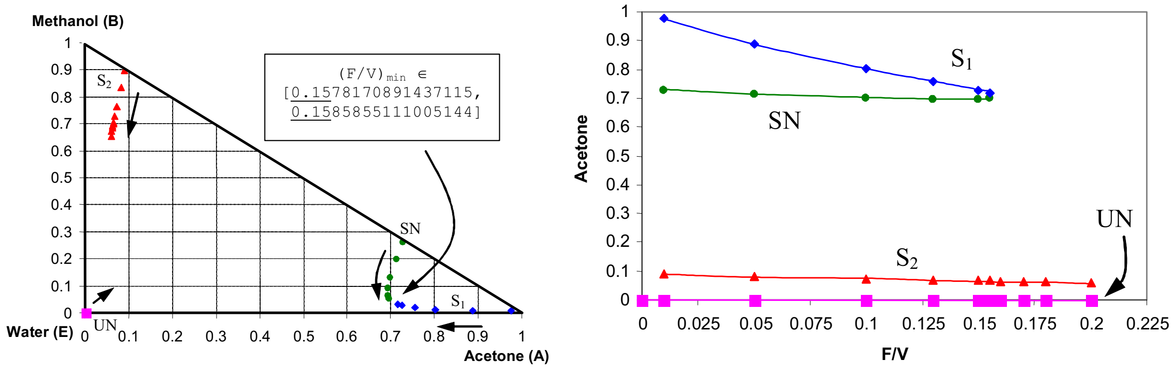


Figure 3. Evolution of the singularities at $R = 10$. a/ Left hand side: Singular point paths. b/ Right hand side: Bifurcation diagram

Example results for the case of batch extractive distillation are shown in **Figures 3a** and **3b**. The exact F/V at which the bifurcation occurs could not be determined by repeatedly finding the singular points because the computation time increases enormously in the neighborhood of the bifurcation.

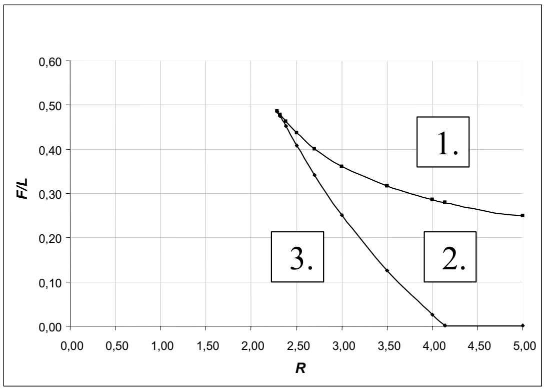


Figure 4. Domains of distinct maps. Domain 1 is feasible.

Example results for the case of continuous extractive distillation are shown in **Figure 4**. The main parameters are R and F/L. The three regions denoted by square labels correspond to different phase maps; domain 1 is the feasible domain. The points of the domain border are determined by finding bifurcation points. (Note: this figure is valid according to the analysis of one criterion only, namely connection of the specified rectifying profile with an extractive profile. Connection of the specified stripping profile with an extractive profile is not analyzed here.)

5. Summary

As it has earlier been shown by researchers of Budapest Univ., Hungary, feasibility of these processes, and the feasible domain of the process parameters, can be well estimated by studying a system of an explicite autonomous first order ordinary differential equation system coupled with a system of algebraic equations. Existence and location of the singular points play a crutial role in feasibility of these processes.

From mathematical point of view, the main problem consists of reliably finding the existence and loci of all the singular points in a given domain, and determining the bifurcations of the phase map. This problem is solved by applying an interval-arithmetic based branch and bound optimization algorithm developed at Univ. Szeged, Hungary.

Feasibility study of extractive distillation variants is a successful application area of interval arithmetics based reliable computation of zeroes and global extrema of real functions.

Singular points in the neighborhood of bifurcation is rather difficult to find because the computation time increases enormously. Determination of the bifurcation points in this way, i.e. by repeated computation of singular points and plotting bifurcation diagrams, is unreliable. Instead, criterion of bifurcation is applied, based on linearization of the differential equation in the neighborhood of the singular points. Implicite function theorem was also applied to compute elements of the *Jacobian*.

The mathematical model is detailed, and the capability of the method is demonstrated on example results related to the separation of acetone – methanol mixture (forming minimum boiling azeotrope) with water as a heavy boiling entrainer, in both batch and continuous extractive distillation processes. All the singular points inside (and even partially outside) the physically valid composition domain are found. Bifurcation points are also found, and domains of radically different maps are determined, including the feasible domain.

Acknowledgments

This research was partially supported by OTKA F046282, T048377, T046822, and T037191.

References

[1] Lelkes, Z., P. Lang, B. Benadda and P. Moszkowicz (1998) ”Feasibility of Extractive Distillation in a Batch Rectifier”, *AIChE Journal* **44** 810.

[2] Lelkes, Z., E. Rév, C. Stéger and Z. Fonyó (2002) ”Batch Extractive Distillation of Maximal Azeotrope with Middle Boiling Entrainer”, *AIChE Journal* **48** 2524.

[3] Rév, E., Z. Lelkes, V. Varga, C. Stéger and Z. Fonyó (2003) ”Separation of minimum boiling binary azeotrope in batch extractive rectifier with intermediate boiling entrainer”, *Ind. Eng. Chem. Research* **42** 162.

[4] Csendes, T. and D. Ratz. (1997) ”Subdivision direction selection in interval methods for global optimization”, *SIAM J. Num. Anal.* **34**, 922.

[5] Csendes, T. (2001) ”Interval Analysis: Subdivision Directions in Interval B&B Methods” in: Floudas, C.A. and Pardalos, P.M. (eds.): *Encyclopedia of Optimization*, Kluwer, Dordrecht

[6] Knüppel, O. (1993) *PROFIL - Programmer's Runtime Optimized Fast Interval Library.* Bericht 93.4, Technische Universität Hamburg-Harburg

[7] Hammer, R., M. Hocks, U. Kulisch and D. Ratz (1993) *Numerical Toolbox for Verified Computing I.* Springer-Verlag, Berlin

Rigorous Affine Lower Bound Functions for Multivariate Polynomials and Their Use in Global Optimisation

Juergen Garloff and Andrew P. Smith

University of Applied Sciences / FH Konstanz, Department of Computer Science, Postfach 100543, D-78405 Konstanz, Germany {garloff,smith}@fh-konstanz.de

Abstract We address the problem of finding tight affine lower bound functions for multivariate polynomials, which may be employed when global optimisation problems involving polynomials are solved with a branch and bound method. These bound functions are constructed by using the expansion of the given polynomial into Bernstein polynomials. The coefficients of this expansion over a given box yield a control point structure whose convex hull contains the graph of the given polynomial over the box. We introduce different methods for computing tight affine lower bound functions based on these control points, using either a linear interpolation of certain specially chosen control points or a linear approximation of all the control points. We present a bound on the distance between the given polynomial and its affine lower bound function, which, at least in the univariate case, exhibits quadratic convergence with respect to the width of the domain. We also address the problem of how to obtain a verified affine lower bound function in the presence of uncertainty and rounding errors. Some numerical examples are presented.

Keywords: Bernstein polynomials, relaxation, affine bound functions, constrained global optimisation

1. Introduction

In our talk we wish to contribute to the solution of the constrained global optimisation problem

$$\min_{x \in F} f(x). \tag{1}$$

The set of *feasible solutions* F is defined by

$$F \ := \left\{ x \in S \ \middle| \ \begin{array}{l} g_i(x) \leq 0 \ \text{ for } \ i = 1, \ldots, m \\ h_j(x) = 0 \ \text{ for } \ j = 1, \ldots, l \\ x \in X \end{array} \right\},$$

where $S \subseteq \mathbf{R}^n$, X is a box contained in S, and f, g_i, h_j are real-valued functions defined on S. The optimisation problem

$$\min_{x \in R} \underline{f}(x) \tag{2}$$

is called a *relaxation* of (1) if the set of feasible solutions fulfils $F \subseteq R$ and $\underline{f}(x) \leq f(x)$ holds for all $x \in F$.

To generate an affine relaxation for problem (1), the functions f, g_i $(i = 1, \ldots, m)$, and h_j $(j = 1, \ldots, l)$ are replaced by affine lower bound functions $\underline{f}$, $\underline{g}_i$, and $\underline{h}_j$, respectively. Then the relaxed problem (2) with the respective set of feasible solutions yields a linear programming problem. Its solution provides a lower bound for the solution of (1). This relaxation may be used in a branch and bound framework for solving problem (1).

We address the use of relaxations for global optimisation problems (1) in which the objective function f and the functions defining the constraints are all multivariate polynomials.

2. Bernstein Expansion

In [3–6] we have shown how diverse affine and convex lower bound functions for polynomials can be constructed by applying the expansion of such a polynomial into Bernstein polynomials.

Using multiindices and multivariate extensions of summation and binomial coefficients, an n-variate polynomial p,

$$p(x) = \sum_{i=0}^{l} a_i x^i, \quad x = (x_1, \ldots, x_n), \tag{3}$$

can be represented over $I = [0,1]^n$ as

$$p(x) = \sum_{i=0}^{l} b_i B_i(x), \tag{4}$$

where

$$B_i(x) = \binom{l}{i} x^i (1-x)^{l-i} \tag{5}$$

and the so-called *Bernstein coefficients* b_i are given by

$$b_i = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} a_j, \quad 0 \leq i \leq l. \tag{6}$$

We may consider the case of the unit box I without loss of generality, since any nonempty box in $\mathbf{R}^n$ can be mapped affinely thereupon.

A fundamental property for our approach is the *convex hull property*

$$\left\{ \begin{pmatrix} x \\ p(x) \end{pmatrix} : x \in I \right\} \subseteq conv \left\{ \begin{pmatrix} i/l \\ b_i \end{pmatrix} : 0 \leq i \leq l \right\}, \tag{7}$$

where the convex hull is denoted by $conv$.

Figure 1 illustrates the convex hull property and the straighforward construction of an affine lower bound function based upon the convex hull of a univariate polynomial of degree 5.

3. Affine Lower Bound Functions

We consider two contrasting approaches for deriving a tight affine lower bound function from the Bernstein control points (coefficients).

The first approach relies on the minimum control point and an appropriate choice of n others. The linear interpolant of these points constitutes a lower bound function and coincides with one of the lower facets of the convex hull of the control points. In the general case, the computation of such an affine lower bound function requires the solution of a linear programming problem, cf. [3]. For special facets, only the solution of a system of linear equations together with a sequence of back substitutions is needed, cf. [5]. Also, an affine transformation may be applied to flatten the global shape of the convex hull (7) such that the obtained affine lower bound functions more tightly approximate the polynomial, cf. [6].

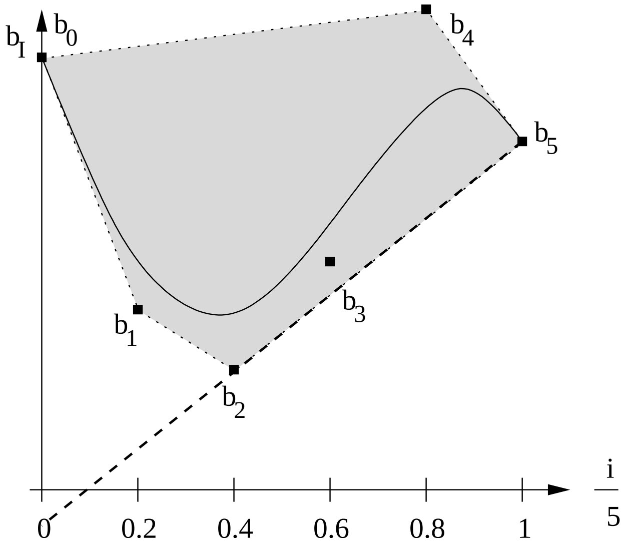


Figure 1. The curve of a polynomial of fifth degree (bold), the convex hull (shaded) of its control points (marked by squares), and an example affine lower bound function (dashed).

The second approach is to derive a linear approximation to the whole set of control points (and thereby the graph of the polynomial) over the box. We consider the use of either a linear least squares approximation or a discrete Chebyshev approximation. This yields a linear function which closely approximates the graph of the polynomial, with valuable shape information. It must be adjusted by a downward shift so that it passes under all the control points, yielding a valid lower bound function.

4. Examples

4.1 Example 1

$$p(x) = \sum_{i=0}^{l} \frac{(-1)^{i+1}}{i+1} x^i, \quad x \in [0,1].$$

Figure 2 shows the graph of p over the box, its control points, and affine lower bound function, for $l = 3, 8, 13, 17$, respectively. This example is taken from [3].

4.2 Example 2

$$\begin{aligned} p_1(x_1,x_2) &= \alpha_1 x_1^2 x_2 + \alpha_2 x_1^2 + \alpha_3 x_1 x_2 + \alpha_4 x_1 + \alpha_5 x_2, \\ p_2(x_1,x_2) &= \alpha_6 x_1^2 x_2 + \alpha_7 x_1 x_2^2 + \alpha_8 x_1 x_2 + \alpha_9 x_2^3 + \alpha_{10} x_2^2 + \alpha_{11} x_2 + \alpha_{12}, \end{aligned}$$

$$\begin{array}{ll} \alpha_1 = -1.697 \times 10^7 & \alpha_7 = 4.126 \times 10^7 \\ \alpha_2 = 2.177 \times 10^7 & \alpha_8 = -8.285 \times 10^6 \\ \alpha_3 = 0.5500 & \alpha_9 = 2.284 \times 10^7 \\ \alpha_4 = 0.4500 \times 10^7 & \alpha_{10} = 1.918 \times 10^7 \\ \alpha_5 = -1.0000 \times 10^7 & \alpha_{11} = 48.40 \\ \alpha_6 = 1.585 \times 10^{14} & \alpha_{12} = -27.73 \end{array}$$

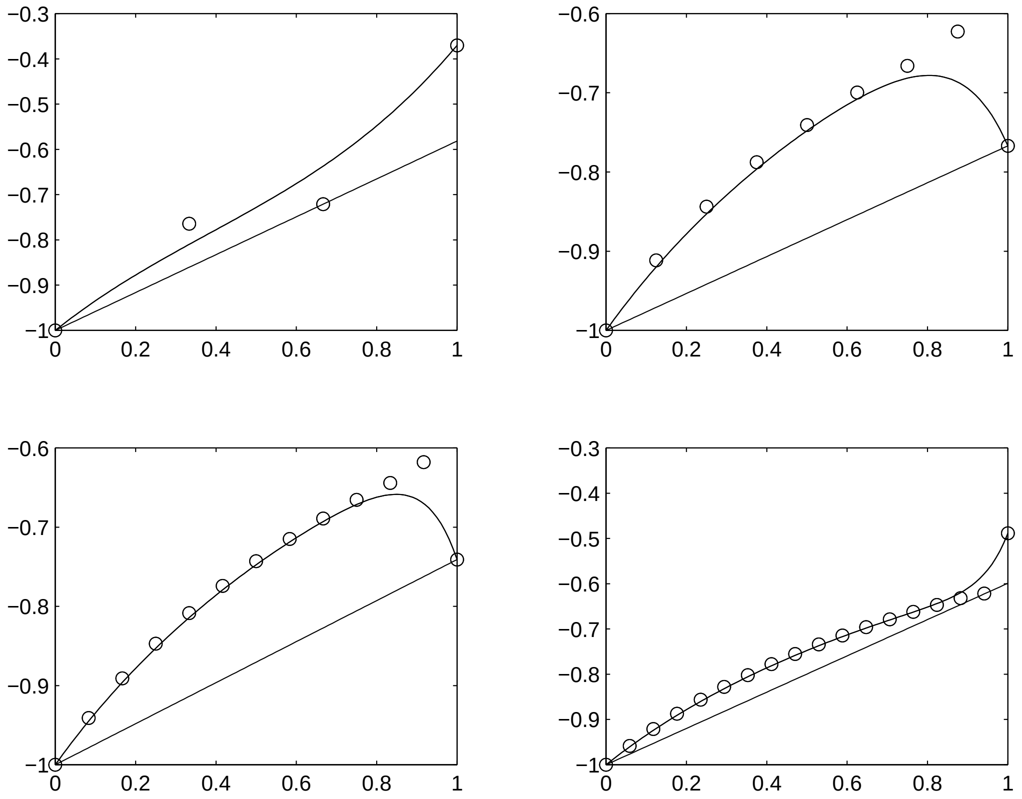


Figure 2. The polynomial of Example 1 for $l = 3, 8, 13, 17$, with its control points (circles), and corresponding lower bound function.

Figure 3 shows the graphs of p_1 and p_2 over the unit box, their control points, and affine lower bound functions. This example, a model for hydrogen combustion with excess fuel, is taken from [7].

5. Rigorous Bound Functions

Due to rounding errors, inaccuracies may be introduced into the calculation of the bound functions. As a result, the computed affine function may not stay below the given polynomial over the box of interest. We also wish to consider the case of uncertain (interval) input data. In our talk, we focus on a method by which an affine lower bound functions based on Bernstein expansion can be computed such that it can be *guaranteed* to stay below the given polynomial. The aforementioned methods can be adjusted to work with interval data, and the safe interpolation of interval control points can be facilitated by a method similar to that introduced in [2].

6. Future Work

We report on the integration of our software into the environment of the COCONUT project [1] funded by the European Union which aims at the solution of global optimisation and continuous constraint satisfaction problems.

We conclude with some suggestions for the extension of our approach to the construction of affine lower bound function for arbitrary sufficiently differentiable functions.

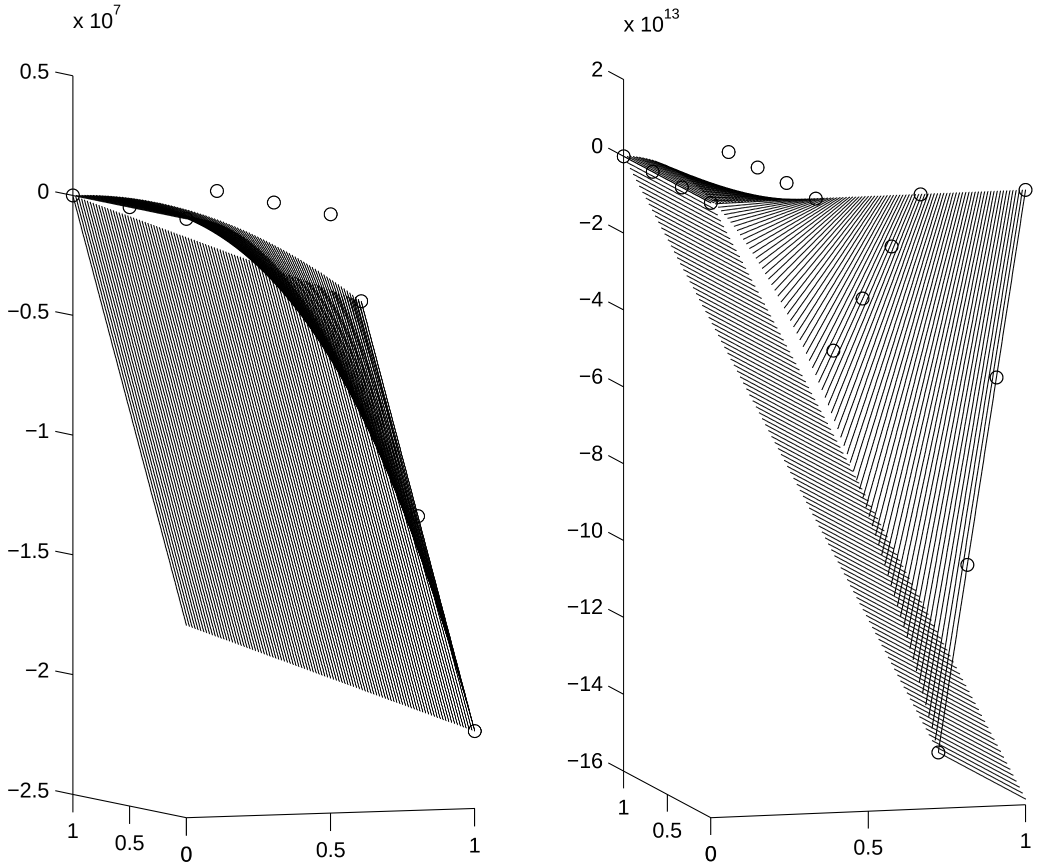


Figure 3. Polynomials p_1 and p_2 from Example 2, with their control points (circles), and corresponding lower bound functions.

References

[1] Bliek C., Spellucci P., Vicente L. N., et al. (2001), "Algorithms for solving nonlinear constrained and optimization problems: The state of the art," a progress report of the COCONUT project, available under `http://www.mat.univie.ac.at/Λneum/glopt/coconut/StArt.html`.

[2] Borradaile G. and Van Hentenryck P. (2005), "Safe and tight linear estimators for global optimization," *Mathematical Programming Vol. 102 (3)*, 495–518.

[3] Garloff J., Jansson C., and Smith A. P. (2003), "Lower bound functions for polynomials," *J. Computational and Applied Mathematics Vol. 157*, 207–225.

[4] Garloff J., Jansson C., and Smith A. P. (2003), "Inclusion isotonicity of convex-concave extensions for polynomials based on Bernstein expansion," *Computing Vol. 70*, 111–119.

[5] Garloff J. and Smith A. P. (2004), "An improved method for the computation of affine lower bound functions for polynomials," in "Frontiers in Global Optimization," Floudas C. A. and Pardalos P. M., Eds., *Series Nonconvex Optimization with its Applications*, Kluwer Acad. Publ., Dordrecht, Boston, London, 135–144.

[6] Garloff J. and Smith A. P. (2005), "A comparison of methods for the computation of affine lower bound functions for polynomials," in "Proceedings of COCOS'03 – 2nd International Workshop on Global Optimization and Constraint Satisfaction," Sam-Haroud J. and Jermann C., Eds., *Lecture Notes in Computer Science*, No. 3478, Springer-Verlag, Berlin, Heidelberg, 71–85.

[7] Morgan A. P. and Shapiro V. (1987), "Box-bisection for solving second-degree systems and the problem of clustering," *ACM Trans. Math. Software Vol. 13 (2)*, 152–167.

Global multiobjective optimization using evolutionary methods: An experimental analysis

C. Gil,[1] R. Baños,[1] M. G. Montoya,[1] A. Márquez,[1] and J. Ortega[2]

[1] *Departamento de Arquitectura de Computadores y Electrónica, Universidad de Almería, La Cañada de San Urbano s/n, 04120 Almería, Spain,* {cgil,rbanos,mari,amarquez}@ace.ual.es

[2] *Departamento de Arquitectura y Tecnología de Computadores, Universidad de Granada, Campus de Fuentenueva s/n, Granada, Spain,* julio@atc.ugr.es

Abstract The objective of this work is to compare the performance of several recent multiobjective optimization algorithms (MOEAs) with a new hybrid algorithm. The main attraction of these algorithms is the integration of selection and diversity maintenance. Since it is very difficult to exactly describe what a good approximation is in terms of a number of criteria, the performance is quantified with specific metrics and is based on two main aspects: the proximity of solutions to the global Pareto font and the suitable distribution of the located front.

Keywords: multiobjective evolutionary optimization, global Pareto-optimal front.

1. Introduction

The aim of Global Optimization (GO) is to find the best solution of decision models, in presence of the multiple local solutions. Having several objective functions, the notion of optimum changes, because in Multiobjective Optimization Problems (MOPs) we are really trying to find good compromises (or trade-offs) rather than a single solution as in global optimization. Since most of the real design problems involve the achievement of several objectives, then the presence of multiple objectives in a problem, in principle, gives rise to a set of optimal solutions (largely known as Pareto-optimal solutions), instead of a single optimal solution [2]. In the absence of any further information, one of these Pareto-optimal solutions cannot be said to be better than the other. This demands a user to find as many Pareto-optimal solutions as possible. Classical optimization methods (including the multicriterion decision-making methods) suggest converting the multiobjective optimization problem to a single-objective optimization problem by emphasizing one particular Pareto-optimal solution at a time. When such a method is to be used for finding multiple solutions, it has to be applied many times, hopefully finding a different solution at each simulation run.

Generating the Pareto set can be computationally expensive and is often infeasible, because the complexity of the underlying application prevents exact methods from being applicable. For this reason, a number of stochastic search strategies such as evolutionary algorithms, tabu search, simulated annealing, and ant colony optimization have been developed. Over the past decade, a number of multiobjective evolutionary algorithms (MOEAs) have been suggested [3–6, 9]. The primary reason for this is their ability to find multiple Pareto-optimal solutions in one single simulation run. Since evolutionary algorithms (EAs) work with a population of solutions, a simple EA can be extended to maintain a diverse set of solutions. With an emphasis for moving toward the global Pareto-optimal region, an EA can be used to find multiple

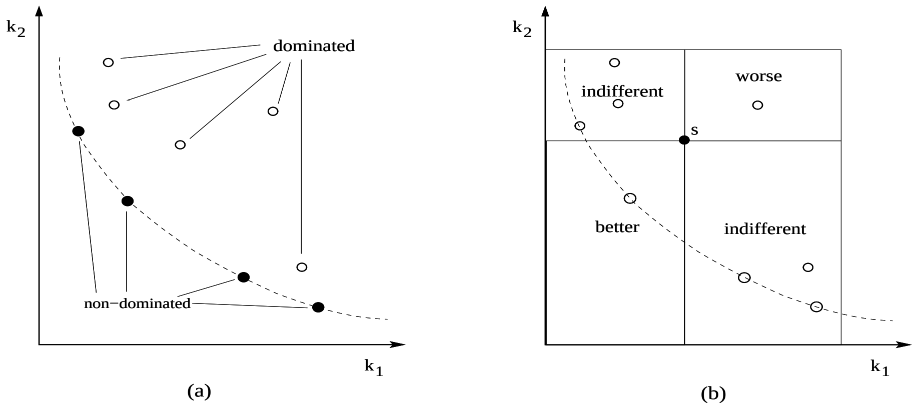


Figure 1. Pareto-dominance relations.

Pareto-optimal solutions in one single simulation run. Since the goal of approximating the Pareto set is itself multiobjective, for instance, we would like to minimize the distance of the generated solutions to the Pareto set and to maximize the diversity of the achieved Pareto set approximation, then it is impossible to exactly describe what a good approximation is in terms of a number of criteria such as closeness to the Pareto set, diversity, etc [10, 11]. In the following, we present some general concepts of Multi-Objective Optimization.

Definition 1. Multi-Objective Optimization (MOO) is the process of searching one or more decision variables that simultaneously satisfy all constraints, and optimize an objective function vector that maps the decision variables to two or more objectives.

$$\text{minimize/maximize}(f_k(s)),\ \forall k \in [1,K]$$
$$\textit{subject to } s \in F$$

Definition 2. Decision vector or solution (s) = (s_1, s_2, .., s_n) represents accurate numerical qualities for an optimization problem. The set of all decision vectors constitutes the *decision space*.

Definition 3. Feasible set (F) is the set of decision vectors that simultaneously satisfies all the constraints.

Definition 4. Objective function vector (f) maps the decision vectors from the decision space into a *K-dimensional* objective space $Z \in \Re^K$,
$z=f(s)$, where: $f(s)=\{f_1(s), f_2(s), ..., f_K(s)\}$, $z \in Z$, $s \in F$.

Let P be a MOO problem, with $K \geq 2$ objectives. Instead of giving a scalar value to each solution, a partial order is defined according to Pareto-dominance relations, as we detail below.

Definition 5. Order relation between decision vectors. Let s and s' two decision vectors. The dominance relations in a minimization problem are:

$$s \textit{ dominates } s' \ (s \prec s') \quad \textit{iff} \quad f(s) < f(s')$$
$$s \textit{ weakly dominates } s' \ (s \preceq s') \quad \textit{iff} \quad f(s) \leq f(s')$$
$$s,s' \text{ are } \textit{indifferent} \ (s \sim s') \quad \textit{iff} \quad f(s) \npreceq f(s') \wedge f(s) \nsucceq f(s')$$

Definition 6. Pareto-optimal solution. A solution s is *Pareto-optimal* if there is no other $s' \in F$, such that $f(s') < f(s)$. All the Pareto-optimal solutions define the *Pareto-optimal set*.

Definition 7. Non-dominated solution. A solution $s \in F$ is *non-dominated* with respect to a set $S' \in F$ if only if $\nexists s' \in S'$, verifying that $s' \prec s$.

Definition 8. Non-dominated set. Let $S' \in F$ and $Z'=f(S')$. The function ND(S') returns the set of non-dominated solutions from S':

$$ND(S')=\{\forall s' \in S' \mid s' \text{ is non-dominated by any other s''}, s'' \in S'\}$$

Figure 1 graphically describes the Pareto-dominance concept for a minimization problem with two objectives (*k1* and *k2*). Figure 1(a) shows the location of several solutions. The filled circles represent non-dominated solutions, while the non-filled ones symbolize dominated solutions. Figure 1(b) shows the relative distribution of the solutions in reference to *s*. There exist solutions that are *worse* (in both objectives) than *s*, *better* (in both objectives) than *s*, and *indifferent* (better in one objective and worse in the other).

In this paper, we address all of these issues for four MOEAS, SPEA2, NSGA-II, PESA and a hybrid version, msPESA, mainly based on PESA and NSGA-II. From the simulation results on a number of difficult test problems, we find that msPESA outperforms three other contemporary MOEAs in terms of finding a diverse set of solutions and in converging near the global Pareto-optimal set.

In the remainder of the paper, we briefly mention these MOEAs in Section II. Thereafter, Section III presents simulation results and compares msPESA with three other elitist MOEAs (SPEA2, NSGA-II and PESA). Finally, we outline the conclusions of this paper.

2. Multiobjective Optimization Algorithms Implemented

We mentioned earlier that, along with convergence to the Pareto-optimal set, it is also desired that a MOEA maintains a good spread of solutions in the obtained set of solutions. In the following, we describe this issues in each algorithm.

SPEA2 (The Strength Pareto Evolutionary Algorithm). SPEA2 [6] combines, in the same fitness value, dominance information about the individual (with rank and count of dominance) as well as the density information about its niche, computed by the nearest neighbor technique. Therefore SPEA2 also implements fitness-sharing. As the file size is fixed, when the number of non-dominated individuals exceeds the file size, a clearing process is followed, considering the smallest distance to the neighbors. SPEA2 uses a fine grain assignment fitness technique incorporating the density information that is useful for the selection process. In addition, the size of the file is fixed and the file fills up with dominated individuals when there are not enough non-dominated ones. The clustering technique is replaced (in comparison with SPEA) by a method which has similar characteristics but does not eliminate extreme solutions (see [6]). Finally, with SPEA2 only the members of the file participate in the mating process.

NSGA-II (Non-dominated Sorting Genetic Algorithm-II). NSGA-II [3] proposes a new partial order between two solutions defined by the crowded comparison operator. The crowded operator acts in the following way: given two solutions, their rankings are first checked. If the rankings are different, the solution with the lower ranking is assumed to be the best one. If they are equal, the density information is checked. In this case, the solution with the less populated niche is assumed to be the best. Although both mechanisms seek to maintain diversity, NSGA-II does not use the crowding mechanism of niches. NSGA-II uses tournament selection based on the defined operator. In the algorithm operation (see [3]), when fronts are added, the size of the allowed population can be exceeded. In this case, the least diversified solutions of the last front are eliminated. This is a clearing procedure. Even so, this clearing is subject to the dominance relation as established in the definition of the comparison operator. In conclusion, while SPEA2 presented a hybrid operation between fitness-sharing and crowding, NSGA-II uses a combination of fitness-sharing and clearing as its niching mechanism.

PESA (The Pareto Envelope-based Selection Algorithm). PESA [4] is a mixed algorithm between PAES [9] and SPEA [5]. It uses a small $internal\ population$ and a (usually) larger $external\ population$. The external population is actually the archive which stores the current approximation to the Pareto front, and the internal population are new candidate solutions vying for incorporation into the archive. PESA implicitly maintains a hyper-grid division of phenotype space which allows it to keep track of the crowding degree in different regions of the archive. However, unlike both PAES and SPEA, selection in PESA is based on this crowding measure. Replacement (deciding what must leave the archive if it becomes overfull) is also based on a crowding measure.

msPESA (Mixed Spreading between PESA and NSGA-II). A new algorithm, which is a hybrid version between PESA and NSGA-II is implemented in order to improve the concept of spreading. In the design of msPESA, the goal was to eliminate the potential weaknesses of other MOEAs and to create a powerful MOEA. The main characteristics of msPESA are:

- It uses a variation of the fast non-dominated sorting algorithm of NSGA-II where only one front is calculated.
- A new archiving strategy is implemented for the external population. Once a candidate has entered the external archive, members of the archive which is dominated will not be removed.If the archive temporarily exceeding the maximum size, one solution must therefore be removed from the archive. The choice is made by first finding the maximal squeeze factor [4] in the population, and removing an arbitrary chromosome which has this squeeze factor.

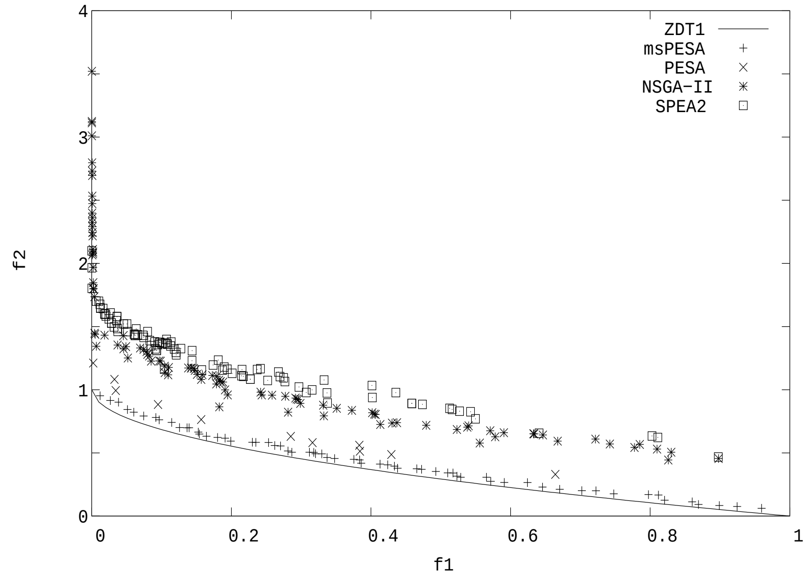


Figure 2. Non-dominated solutions obtained using NSGA-II, SPEA2, PESA and msPESA on ZDT1.

3. Experimental Analysis

In this section we consider the issues necessary to compare the performance of these algorithms over a set of benchmarks. The elaboration of a merit ranking between the various methods is not a trivial procedure. In general, an order of MOEA merit is impossible due to the NFL Theorem [12], although it is possible to extract some results from the behavior of each algorithm. In this paper we have taken into account the proximity to the Pareto front and the uniformity in the distribution of the solutions. In the search for an impartial, accurate comparison that excludes the effects of chance, it is necessary to consider many aspects, and for this reason we have made our own implementation of each algorithm, and all of them have been integrated in the same platform. Since EAs are highly configurable procedures [1],

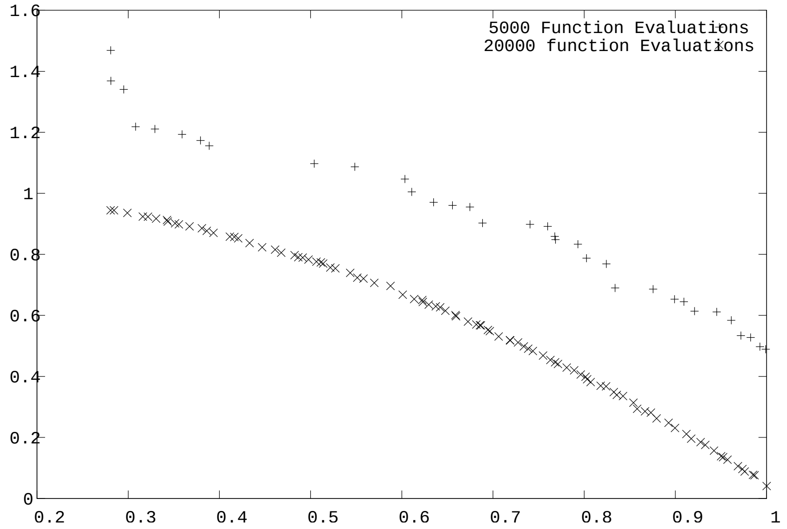


Figure 3. Non-dominated solutions using msPESA with 5000 and 20000 fitness function evaluations.

the first step is to assign values to the parameters. Some values are fixed and equal for all the methods during the whole comparison. Although undoubtedly, each algorithm could be improved by assigning particular values to the parameters, the authors do not suggest anything in this respect and leave the users freedom of choice. Therefore, and for reasons of simplicity, the following values, also chosen in other studies [5] have been used. Number of generations: $T = 250$, size of the internal population $IP = 100$, external population $EP = 100$ excepts for PESA where $PE = 10$, crossover probability: $P_x = 0.8$ and mutation probability: $P_\mu = 0.1$. The benchmarks used were ZDT1 to ZDT6 [5], but for reason of space, we have not included all these results in this work.

Figure 2 shows the non-dominated solutions obtained using msPESA, PESA, NSGA-II and SPEA2 for test (a)ZDT1, (b)ZDT3, and (c)ZDT6. It is clear that msPESA is able to better distribute its population along the obtained front than other MOEAs. Figure 2 (d) shows the non-dominated solutions when the fitness evaluations are increased from 5000 to 20000 in msPESA.

Metric C presented in [5] have been used to evaluate the performance of the different optimization methods. This metric establishes a comparison between two algorithms, indicating how the results of the first cover those of the second. Table 1 summarizes results for the set of experiments in which each trial run was allowed just 5000 fitness evaluations for test ZDT1, ZDT3 and ZDT6. This table shows that the best performing algorithm is msPESA (see msPESA row in Table 1). That is true in proximity to the Pareto front as in diversity and spread of solutions.

Table 1. Comparison of C metric using msPESA, NSGA-II, PESA and SPEA2.

	msPESA		**NSGA-II**		**PESA**		**SPEA2**	
msPESA	*ZDT1*	*ZDT3*	1.00	1.00	1.00	1.00	1.00	1.00
	ZDT6	-	1.00	-	1.00	-	1.00	-
NSGA-II	0.06	0.06			0.08	0.18	0.23	0.00
	0.01	-			0.75	-	0.12	-
PESA	0.02	0.00	1.00	1.00			1.00	0.00
	0.00	-	0.57	-			0.5	-
SPEA2	0.09	0.16	0.79	1.00	0.18	1.00		
	0.01	-	1.0	-	0.9	-		

4. Conclusions of the work

In this paper, we have proposed a new hybrid multiobjective evolutionary algorithm based on non-dominated sorting approach of NSGA-II and internal and external archiving approach of PESA. We have compared its performance with three others recent MOEAs on a suite of test function. As we have commented in the experimental results section, although an order of merit between different algorithms is very difficult, we have focused this work to obtain a more precise analysis about spreading of solutions. Comparative performance was measured using a coverage metric and we found that msPESA was able to maintain a better spread of solutions and convergence better in the obtained nondominated front. However, results on a limited set of test functions must always be regarded as tentative, and hence much further work is needed.

Acknowledgments

This work was supported by the Spanish MCyT under contracts TIC2002-00228. Authors appreciate the support of the "Structuring the European Research Area" program, R113-CT-2003-506079, funded by the European Commission. R. Baños acknowledges a FPI doctoral fellowship from the regional government of Andalucia.

References

[1] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, New York, 1989.

[2] Fonseca, C. M.; Flemming, P. J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discusion and Generalization. In *S. Forrest (eds.): Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, 1993, 416-423.

[3] Deb, K., Agrawal, S., Pratap, A. Meyarivan, T. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multiobjective Optimization: NSGA-II. In: *M. Schoenauer (eds)* Parallel Problem Solving from Nature, 2000, 849-858.

[4] Corne, D.W., Knowles, J.D., Oates, H.J. The Pareto-Envelope based Selection Algorithm for Multiobjective Optimisation. In: *M. Schoenauer (eds)* Parallel Problem Solving from Nature,Lecture Notes in Computer Science, 1917, 2000, 869-878.

[5] Zitler, E., Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, Vol.3 No.4, 1999, 257-271.

[6] Zitzler, E., Laumanns, M., Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, 2001.

[7] Deb, K., Goel, T. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. EMO In: *E. Zitzler et al. (eds): Lecture Notes in Computer Science*, 2001, 67-81.

[8] Laumanns, M., Zitzler, E., Thiele, L. On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization. EMO In: *E. Zitzler et al. (eds):Lecture Notes in Computer Science*, 2001, 181-196

[9] Knowles, J. D., Corne, D. W. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation. In Congress on Evolutionary Computation, Vol. 1, Piscataway, NJ. IEEE Press, 1999, 98-105.

[10] Deb, K. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, 2002.

[11] Coello, C. A., Van Veldhuizen, D. A., Lamont, G.B. Evolutionary Algorithms for Solving Multi-Objective Problems. *Kluwer Academic Publishers*, 2002.

[12] Macready, W.G.; Wolpert, D.H. The No Free Lunch theorem. *IEEE Trans. on Evolutionary Computing*, Vol. 1, No. 1, 1997, 67-82.

Conditions for ε-Pareto Solutions in Multiobjective Optimization

C. Gutiérrez,[1] B. Jiménez,[2] and V. Novo[3]

[1] *Universidad de Valladolid, Valladolid, Spain,* cesargv@mat.uva.es

[2] *Universidad de Salamanca, Salamanca, Spain,* bjimen1@encina.pntic.mec.es

[3] *Universidad Nacional de Educación a Distancia, Madrid, Spain,* vnovo@ind.uned.es

Abstract In this paper, approximate Pareto solutions of nondifferentiable constrained multiobjective optimization problems are studied via a metrically consistent ε-efficient concept introduced by Tanaka [14]. Necessary and sufficient conditions for these solutions are obtained from a nonconvex penalized scalarization process. Necessary conditions are provided in convex multiobjective optimization problems through Kuhn-Tucker multiplier rules. Sufficient conditions are obtained via Kuhn-Tucker multiplier rules under convexity hypotheses and via approximate solutions of a scalar Lagrangian problem for nonconvex multiobjective optimization problems.

Keywords: Multiobjective optimization, ε-Pareto solution, scalarization, ε-subdifferential.

1. Introduction

During the last decades, interest in approximate solutions or ε-efficient solutions (ε-Pareto solutions in the Paretian context) of vector optimization problems is growing, since these solutions exist under very mild hypotheses and they are obtained by a lot of usual resolution methods (for example, by iterative algorithms, heuristic methods, etc.).

The first and most popular ε-efficient concept was introduced by Kutateladze [8] and has been used to establish approximate Kuhn-Tucker type conditions and approximate duality theorems [2, 4, 9–13, 15]. However, Kutateladze's ε-efficiency concept gives approximate solutions which are not metrically consistent, i.e., it is possible to obtain feasible points (x_n), x_0 such that their objective values verify $f(x_n) \to f(x_0)$, x_n is an ε_n-efficient solution for each n, $\varepsilon_n \to 0$, and $f(x_0)$ is far from the optimal value set.

In [1, 4–7, 15, 18], various metrically consistent ε-efficient concepts based on a previously fixed scalar functional have been studied. However, there are a lot of problems for which is not possible to choose any previous scalar functional and so several other metrically consistent notions have been introduced without consider any additional scalar functional (see, for example, the concepts defined by White [16] and Tanaka [14]).

Classical conditions for efficient solutions via multiplier rules, Lagrangian functionals and saddlepoint theorems must be extended to ε-Pareto solutions in order to develop new and better resolution methods. In [2, 3, 9–12, 17] some results have been obtained following this line, but using not metrically consistent ε-Pareto concepts. In this work, Tanaka's ε-Pareto notion is analyzed from these points of view in order to extend those classical conditions to a metrically consistent ε-Pareto concept without consider any additional scalar functional.

In Section 2, the nondifferentiable constrained multiobjective optimization problem is presented and some notations are fixed. Moreover, the Tanaka's ε-Pareto concept is recalled and some properties are established. Then, we describe a general method to convert a constrained multiobjective optimization problem into a scalar optimization problem without inequality constraints in such a way that ε-Pareto solutions for the first problem are approximate solutions for the second problem. In Section 3, Fritz John and Kuhn-Tucker necessary and sufficient conditions for ε-Pareto solutions are proved via the ε-subdifferential and the scalarization method developed in Section 2. In obtaining these conditions, convexity hypotheses are assumed. In Section 4, a sufficient condition for ε-Pareto solutions is proved through approximate solutions of a scalar Lagrangian functional. From this result, it is possible to obtain approximate metrically consistent solutions of constrained multiobjective optimization problems via suboptimal solutions of unconstrained scalar optimization problems.

2. Metrically consistent ε-Pareto solutions

Let X be a normed space and let us fix $p+m$ functionals $f_i, g_j : X \to \mathbb{R}$, $i = 1,2,\dots,p$, $j = 1,2,\dots,m$, a continuous linear map $A : X \to \mathbb{R}^k$ and a nonempty set $G \subset X$. In the sequel, the following constrained multiobjective optimization problem is considered

$$\operatorname{Min}\{f(x) : x \in S\}, \tag{1}$$

where $f = (f_1, f_2, \dots, f_p) : X \to \mathbb{R}^p$, $S = K \cap Q \cap G$,

$$K = \{x \in X : g_j(x) \le 0,\ j = 1,2,\dots,m\},$$
$$Q = \{x \in X : Ax = b\},$$

and $b \in \mathbb{R}^k$.

In solving (1), the componentwise partial order in the final space is assumed. Let us denote by $\operatorname{cl}(M)$ and $\operatorname{int}(M)$ the closure and interior of a set M, respectively. The nonnegative orthant in $\mathbb{R}^p$ is denoted by $\mathbb{R}^p_+$ and we write the set $\operatorname{int}(\mathbb{R}^p_+)$ as $\mathbb{R}^p_{++}$.

Definition 1. *A point $x_0 \in S$ is a Pareto (resp. weak Pareto) solution of (1) if*

$$(f(x_0) - \mathbb{R}^p_+ \backslash \{0\}) \cap f(S) = \emptyset$$

(resp. $(f(x_0) - \mathbb{R}^p_{++}) \cap f(S) = \emptyset$).

The set of Pareto solutions and weak Pareto solutions of (1) will be denoted by $\mathrm{E}(f,S)$ and $\mathrm{WE}(f,S)$, respectively. It is clear that $\mathrm{E}(f,S) \subset \mathrm{WE}(f,S)$.

Consider $\varepsilon \ge 0$ and the scalar optimization problem

$$\operatorname{Min}\{h(x) : x \in M\}, \tag{2}$$

where $h : X \to \mathbb{R}$ and $M \subset X$, $M \ne \emptyset$. In the following definition, the well-known notion of approximate (suboptimal) solution of (2) is recalled.

Definition 2. *A point $x_0 \in M$ is an ε-solution of (2) if*

$$h(x_0) - \varepsilon \le h(x), \quad \forall\, x \in M.$$

The set of ε-solutions of (2) is denoted by $\operatorname{AMin}(h, M, \varepsilon)$.

Let $\mathbb{B} \subset \mathbb{R}^p$ be the unit closed ball of $\mathbb{R}^p$ defined by a norm $\|\ \|$. Next, we recall a concept introduced by Tanaka [14], which extends Definition 2 to multiobjective optimization problems.

Definition 3. *A point $x_0 \in S$ is an ε-Pareto (resp. weak ε-Pareto) solution of (1) if*

$$(f(x_0) - ((\varepsilon\mathbb{B})^c \cap \mathbb{R}^p_+)) \cap f(S) = \emptyset$$

(resp. $(f(x_0) - ((\varepsilon\mathbb{B})^c \cap \mathbb{R}^p_{++})) \cap f(S) = \emptyset$).

We denote by $\mathrm{AE}(f, S, \|\ \|, \varepsilon)$ and $\mathrm{WAE}(f, S, \|\ \|, \varepsilon)$ the sets of ε-Pareto solutions and weak ε-Pareto solutions of (1), respectively. Let us observe that these sets depend on the norm $\|\ \|$. Moreover, it is clear that $\mathrm{AE}(f, S, \|\ \|, \varepsilon) \subset \mathrm{WAE}(f, S, \|\ \|, \varepsilon)$ for each $\varepsilon \geq 0$ and $\mathrm{AE}(f, S, \|\ \|, 0) = \mathrm{E}(f, S)$, $\mathrm{WAE}(f, S, \|\ \|, 0) = \mathrm{WE}(f, S)$.

Next, we show under different hypotheses that Tanaka's concept is a metrically consistent ε-efficiency notion.

Theorem 4. *Let $f : X \to \mathbb{R}^p$ be a continuous map at $x_0 \in S$ and let $(\varepsilon_n) \subset \mathbb{R}_+$, $(x_n) \subset S$ be such that $\varepsilon_n \downarrow 0$ and $x_n \to x_0$.*

1. *If $x_n \in \mathrm{AE}(f, S, \|\ \|, \varepsilon_n)$ for each n, then $x_0 \in \mathrm{WE}(f, S)$.*
2. *If $(f(x_n))$ is a nonincreasing sequence, i.e.,*

$$f(x_m) \in f(x_n) - \mathbb{R}^p_+, \quad \forall\, m > n$$

and $x_n \in \mathrm{AE}(f, S, \|\ \|, \varepsilon_n)$ for each n, then $x_0 \in \mathrm{E}(f, S)$.

3. *Suppose that $f(S)$ is externally stable with respect to the efficient set:*

$$f(S) \subset f(\mathrm{E}(f, S)) + \mathbb{R}^p_+.$$

If $x_n \in \mathrm{AE}(f, S, \|\ \|, \varepsilon_n)$ for each n, then $f(x_0) \in \mathrm{cl}(f(\mathrm{E}(f, S)))$.

3. Multiplier rules for ε-Pareto solutions

In this section, multiplier rules for ε-Pareto solutions of (1) are proved under convexity hypotheses. So, let us suppose that f_i and g_j are $p + m$ continuous convex functionals and G is a convex set. We use $\mathrm{Ker}(A)$ and $\mathrm{Ker}(A)^\perp$ to denote the kernel of A and the orthogonal complement of $\mathrm{Ker}(A)$. The topological dual space of X is denoted by X^*. We write I_G for the indicator functional of the set G and $\|\ \|_1$ for the l_1 norm in $\mathbb{R}^p$.

To attain our objective, we use the ε-subdifferential of a proper convex functional and a penalized scalarization process.

Definition 5. *Let $h : X \to \mathbb{R} \cup \{\infty\}$ be a convex proper functional, $x_0 \in dom(h)$ and $\varepsilon \geq 0$. The ε-subdifferential of h at x_0 is the set $\partial h_\varepsilon(x_0)$ defined by*

$$\partial_\varepsilon h(x_0) = \{x^* \in X^* : h(x) \geq h(x_0) - \varepsilon + \langle x^*, x - x_0\rangle, \forall\, x \in X\}.$$

Let us recall that the subdifferential $\partial h(x_0)$ of h at x_0 in the sense of Convex Analysis is obtained taking $\varepsilon = 0$ in Definition 5.

Theorem 6. *If $Q \cap \mathrm{int}(G) \neq \emptyset$ and $x_0 \in \mathrm{WAE}(f, S, \|\ \|_1, \varepsilon)$ then there exists $(\tau, \nu, \alpha) \in \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}$ and multipliers $(\lambda, \gamma, \mu) \in \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^m$ such that*

$$(\tau, \nu, \alpha, \lambda, \gamma, \mu) \geq 0, \tag{3}$$

$$\sum_{i=1}^p \lambda_i + \gamma + \sum_{j=1}^m \mu_j = 1, \tag{4}$$

$$0 \in \sum_{i=1}^p \partial_{\tau_i}((\lambda_i + \gamma) f_i)(x_0) + \sum_{j=1}^m \partial_{\nu_j}(\mu_j g_j)(x_0) + \mathrm{Ker}(A)^\perp + \partial_\alpha \mathrm{I}_G(x_0), \tag{5}$$

$$\sum_{i=1}^p \tau_i + \sum_{j=1}^m \nu_j - \gamma\varepsilon + \alpha \leq \sum_{j=1}^m \mu_j g_j(x_0). \tag{6}$$

Theorem 7. *Consider $x_0 \in S$. If there exists $(\tau, \nu, \alpha, \lambda, \gamma, \mu) \in \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^m$ verifying conditions (3)-(6) with strict inequality in (6), then $x_0 \in \mathrm{AE}(f, S, \|\ \|_1, \varepsilon)$.*

4. Conditions for ε-Pareto solutions via a Lagrangian functional

In this section, a sufficient condition for ε-Pareto solutions of (1) is established through approximate solutions of a unconstrained scalar optimization problem obtained from (1) via a Lagrangian functional.

In the sequel, we extend the final space to $\mathbb{R}^p \cup \{\pm\infty\}$ and we assume that the usual algebraic and ordering properties hold. Let $L : X \times \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R} \cup \{\pm\infty\}$ be the Lagrangian scalar functional defined by

$$L(x, \lambda, \mu, \rho) = \begin{cases} \infty & \text{if} \quad x \notin G \\ \langle \lambda, f(x)\rangle + \langle \mu, g(x)\rangle + \langle \rho, A(x)\rangle & \text{if} \quad x \in G,\ \mu \in \mathbb{R}^m_+ \\ -\infty & \text{if} \quad x \in G,\ \mu \notin \mathbb{R}^m_+. \end{cases}$$

Fixed $(\lambda, \mu, \rho) \in \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}^k$, let us consider the following Lagrangian scalar optimization problem of (1):

$$\operatorname{Min}\{L_{\lambda,\mu,\rho}(x) : x \in X\}, \tag{7}$$

where $L_{\lambda,\mu,\rho} : X \to \mathbb{R} \cup \{\pm\infty\}$ is the functional defined for all $x \in X$ by $L_{\lambda,\mu,\rho}(x) = L(x, \lambda, \mu, \rho)$. Next, we obtain a sufficient condition for ε-Pareto solutions of (1) through approximate solutions of (7).

Theorem 8. *Let $(\lambda_0, \mu_0, \rho_0) \in \mathbb{R}^p_{++} \times \mathbb{R}^m_+ \times \mathbb{R}^k$ and $x_0 \in S$ be such that $x_0 \in \operatorname{AMin}(L_{\lambda_0,\mu_0,\rho_0}, X, \varepsilon)$. Then $x_0 \in \operatorname{AE}(f, S, \|\,\|_1, \varepsilon_0)$, where*

$$\varepsilon_0 = (\varepsilon - \langle \mu_0, g(x_0)\rangle)/ \min_{1\le i\le p}\{\lambda_i\}.$$

Acknowledgments

This research was partially supported by Ministerio de Ciencia y Tecnología (Spain), project BFM2003-02194.

References

[1] D. Dentcheva and S. Helbig. On variational principles, level sets, well-posedness, and ε-solutions in vector optimization. *J. Optim. Theory Appl.*, 89(2):325–349, 1996.

[2] J. Dutta and V. Vetrivel. On approximate minima in vector optimization. *Numer. Funct. Anal. Optim.*, 22(7&8):845–859, 2001.

[3] M. G. Govil and A. Mehra. ε-optimality for multiobjective programming on a Banach space. *European J. Oper. Res.*, 157:106–112, 2004.

[4] C. Gutiérrez. *Condiciones de ε-Eficiencia en Optimización Vectorial*. PhD thesis, Universidad Nacional de Educación a Distancia, Madrid, 2004.

[5] C. Gutiérrez, B. Jiménez, and V. Novo. A chain rule for ε-subdifferentials with applications to approximate solutions in convex Pareto problems. *J. Math. Anal. Appl.*, In press, 2005.

[6] C. Gutiérrez, B. Jiménez, and V. Novo. Multiplier rules and saddle-point theorems for Helbig's approximate solutions in convex Pareto problems. *J. Global Optim.*, 32(3), 2005.

[7] S. Helbig and D. Pateva. On several concepts for ε-efficiency. *OR Spektrum*, 16:179–186, 1994.

[8] S. S. Kutateladze. Convex ε-programming. *Soviet Math. Dokl.*, 20(2):391–393, 1979.

[9] J. C. Liu. ε-duality theorem of nondifferentiable nonconvex multiobjective programming. *J. Optim. Theory Appl.*, 69(1):153–167, 1991.

[10] J. C. Liu. ε-Pareto optimality for nondifferentiable multiobjective programming via penalty function. *J. Math. Anal. Appl.*, 198:248–261, 1996.

[11] J. C. Liu. ε-properly efficient solutions to nondifferentiable multiobjective programming problems. *Appl. Math. Lett.*, 12:109–113, 1999.

[12] J. C. Liu and K. Yokoyama. ε-optimality and duality for multiobjective fractional programming. *Comput. Math. Appl.*, 37:119–128, 1999.

[13] P. Loridan. ε-duality theorem of nondifferentiable nonconvex multiobjective programming. *J. Optim. Theory Appl.*, 74(3):565–566, 1992.

[14] T. Tanaka. A new approach to approximation of solutions in vector optimization problems. In M. Fushimi and K. Tone, editors, *Proceedings of APORS, 1994*, pages 497–504. World Scientific Publishing, Singapore, 1995.

[15] I. Vályi. Approximate saddle-point theorems in vector optimization. *J. Optim. Theory Appl.*, 55(3):435–448, 1987.

[16] D. J. White. Epsilon efficiency. *J. Optim. Theory Appl.*, 49(2):319–337, 1986.

[17] K. Yokoyama. Epsilon approximate solutions for multiobjective programming problems. *J. Math. Anal. Appl.*, 203:142–149, 1996.

[18] K. Yokoyama. Relationships between efficient set and ε-efficient set. pages 376–380. Proceedings of the International Conference on Nonlinear Analysis and Convex Analysis. World Scientific Publishing, River Edge, New York, 1999.

On the goodness of Global Optimisation Algorithms

Eligius M.T. Hendrix

Operationele Research en Logistiek Groep, Wageningen Universiteit, eligius.hendrix@wur.nl

Abstract The aim of the paper is to arrive at introductory text for students on the concepts of Global Optimization algorithms. Target is to learn to read and interpret optimisation algorithms and to analyse them on goodness. Before going deeper into mathematical analysis, it would be good for students to get a flavour of the difficulty by letting them experiment with simple algorithms that can be followed by hand or spreadsheet calculations. Two simple one-dimensional examples are introduced and in this abstract two NLP algorithms are elaborated. In the final talk this is widened to some deterministic and stochastic GO methods.

Keywords: Efficiency, Effectiveness.

1. Introduction

In this presentation, several criteria are discussed to measure effectiveness and efficiency of algorithms. Examples are given of basic algorithms. To do so, one should first introduce the concept of optimisation algorithms. An algorithm is a description of steps to be taken preferably implemented into a computer program with the aim to find an approximation of an optimum point. The aim as such can be several: reach a local optimum point, reach a global optimum point, find all global optimum points, reach all global and local optimum points. We will come back to that. In general, an algorithm generates a series of points x_k that approximates a (or the or all) optimum point. According to the generic description of [6]:

$$x_{k+1} = Alg(x_k, x_{k-1}, ..., x_0, \xi) \tag{1}$$

where ξ is a random variable and index k is the iteration counter. So the idea it describes is that a next point x_{k+1} is generated and evaluated based on the information in all former points $x_k, x_{k-1}, ..., x_0$ (x_0 is usually called the starting point) and possibly some random effect. In the complete paper, three types of algorithms are described.

- Nonlinear optimisation algorithms, that based on a starting point will try to capture the "nearest" local minimum point.
- Deterministic GO methods which guarantee to approach the global optimum and require a certain mathematical structure.
- Stochastic GO methods which are based on the random generation of feasible trial points and nonlinear local optimization procedures.

We will consider several examples for illustrative purposes. There are two questions to address if we investigate the quality of algorithms.

- Effectiveness: does the algorithm find what we want?
- Efficiency: what are the computational costs?

Several measurable performance indicators can be defined for these global criteria.

1.1 Effectiveness

Focusing on effectiveness there are several targets a user of the algorithm may have:

1. To discover all global minimum points. This of course can only be realised when the number of global minimum points is finite.
2. To detect at least one global optimal point.
3. To find a solution with a function value as low as possible.
4. To produce a uniform covering: This idea as introduced by [3], can be relevant for population based algorithms.

The first and second targets are typical satisfaction targets; was the search successful or not? What are good **measures of success**? In the older literature, often convergence was used i.e. $x_k \to x^*$, where x^* is one of the minimum points. Alternatively one observes $f(x_k) \to f(x^*)$. In tests and analyses, to make results comparable, one should be explicit in the definitions of success. We need not only to specify ϵ and/or δ such that

$$\|x_k - x^*\| < \epsilon \text{ and/or } f(x_k) < f(x^*) + \delta \tag{2}$$

but we should also specify whether success means that there is an index K such that (2) is true for all $k > K$. Alternatively, a record $min_k f(x_k)$ may have reached level $f(x^*) + \delta$ and this can be considered a success. Whether the algorithm is effective also depends on the stochastic nature of it. When we are dealing with stochastic algorithms, the effectiveness can be expressed as the probability that a success has been reached. In analysis, this probability can be derived when having sufficient assumptions on the behaviour of the algorithm and in numerical experiments it can be estimated by looking over repeated runs how many times the algorithm leads to convergence. We will give some examples of such analysis.

1.2 Efficiency

Globally efficiency is defined as the effort the algorithm needs to be successful. A usual indicator for algorithms is the (expected) number of **function evaluations** necessary to reach the optimum. This indicator depends on many factors such as the shape of the test function and the termination criteria used. The indicator more or less suggests that the calculation effort of function evaluations dominates the other calculation effort of the algorithm. Several other indicators appear in literature that can be considered to be derived from the main indicator.

In nonlinear programming (e.g. [5] and [2]) the concept of **convergence speed** is common. It is a limiting concept on the convergence of the series x_k. Let $x_0, x_1, \ldots, x_k, \ldots$ converge to point x^*. The largest number α for which

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^\alpha} = \beta < \infty \tag{3}$$

gives the order of convergence, whereas β is called the convergence factor. In this terminology, among others the following concepts appear

- linear convergence means $\alpha = 1$ and $\beta < 1$
- quadratic convergence means $\alpha = 2$ and $0 < \beta < 1$
- superlinear convergence: $1 < \alpha < 2$ and $\beta < 1$, i.e. $\beta = 0$ for $\alpha = 1$.

Mainly in deterministic GO algorithms, information on the past evaluations is stored in the computer memory. This requires efficient data handling for looking up necessary information during the iterations. As well **memory requirement** as retrieving actions become a part of the computational burden that cannot be neglected compared to the computational effort due to function evaluations. These aspects become important from the perspective of efficiency of algorithms.

In stochastic GO algorithms an efficiency indicator is the **success rate** defined as the probability that the next iterate is an improvement on the record value found thus far $P(f(x_k) < min_{l=1,\ldots,k-1}f(x_l))$. Its theoretical relevance to convergence speed was analysed by [7] and [1], who showed that a fixed success rate of an effective algorithm (in the sense of uniform covering) gives an algorithm with the expected number of function evaluations growing polynomially with the dimension of the problem. However, the empirical measurements can only be established in the limit when such an algorithm stabilises, and only for specifically designed test cases [4].

We do not go deeper into theoretical aspects here of performance indicators on algorithms. Instead some basic algorithms will be introduced and analysed.

2. Illustrative functions

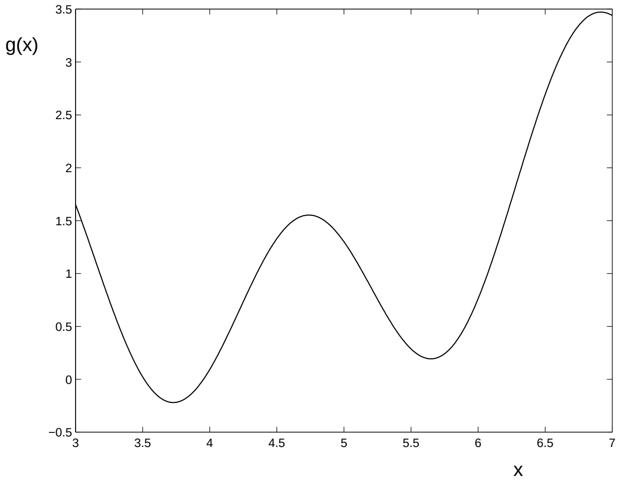


Figure 1. Test case $g(x)$ with two optima

Two testcases are introduced first for which the algorithms are inspected. We consider the minimisation of the following functions.

$$g(x) = \sin(x) + \sin(3x) + \ln(x), x \in [3, 7] \tag{4}$$

Function g is depicted in Figure 1 and has three minimum points on the interval. The global minimum is attained at about $x^* = 3.73$, where $f(x^*) = -0.220$. The derivative function is

$$g'(x) = \cos(x) + 3\cos(3x) + \frac{1}{x} \tag{5}$$

on the interval $[3, 7]$. Alternatively to function g, we introduce a function h with more local minimum points by adding to function g a bubble function based on $\text{frac}(x) = x - \text{round}(x)$ where $\text{round}(x)$ rounds x to the nearest integer. Now the second case is defined as

$$h(x) = \begin{cases} g(x) + 1.5\text{frac}^2(4x) & \text{for} \quad x \neq \frac{1}{4}k + \frac{1}{8}, k \in \mathbb{Z} \\ g(x) + 0.375 & \text{for} \quad x = \frac{1}{4}k + \frac{1}{8}, k \in \mathbb{Z}. \end{cases} \tag{6}$$

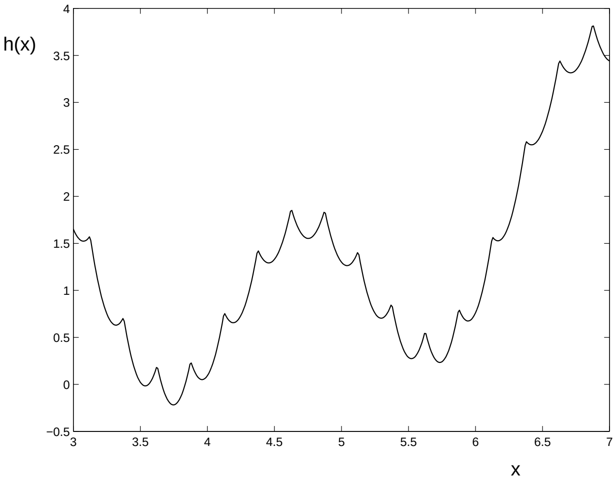


Figure 2. Test case $h(x)$ with 17 optima

As the function frac is not well defined in the points $x = k + \frac{1}{2}, k \in \mathbb{Z}$, the function has been made continuous for the points in $\mathbb{S} = \{x = \frac{1}{4}k + \frac{1}{8}, k \in \mathbb{Z}\}$. The result can be observed in Figure 2 Function h has 17 local minimum points on the interval $[3, 7]$. Although g nor h are convex on the interval, at least function h is piecewise convex on the intervals in between the points of $\mathbb{S}$. At these points, h is not differentiable. For the rest of the interval one can define the derivative

$$h'(x) = g'(x) + 12 \times \text{frac}(4x) \;\; \text{for} \;\; x \in \mathbb{S} \tag{7}$$

The global minimum point of h on $[3, 7]$ is shifted slightly compared to g towards $x^* = 3.75$, where $f(x^*) = -0.217$. In the following Sections, we will consider algorithms on their ability to find minima of the two test cases. One should set a target on what is considered an acceptable or successful result. For instance one can aim at detecting a local minimum or detecting the global minimum. For the neighbourhood we will take an acceptance of $\epsilon = 0.01$. For determining an acceptable low value of the objective function we take $\delta = 0.01$. Notice that the range the functions take between minimum and maximum on the interval is about 4, such that δ represents about 0.25% of the function values range.

3. Two algorithms: Bisection and Newton

Two algorithms from nonlinear programming are sketched and their performance measured for the two test cases. First the bisection algorithm is considered.

Algorithm 2 Bisect($[l, r], f, \epsilon$**)**

```
Set $k = 0$, $l_0 = l$ and $r_0 = r$
while ($r_k - l_k > \epsilon$)
    $x_k = \frac{l_k + r_k}{2}$
    if $f'(x_k) < 0$
        $l_{k+1} = x_k$ and $r_{k+1} = r_k$
    else
        $l_{k+1} = x_k$ and $r_{k+1} = r_k$
    $k = k + 1$
End while
```

The algorithm departs from a starting interval $[l, r]$ that is halved iteratively based on the sign of the derivative in the midpoint. This means that the method is only applicable when the derivative is available at the generated midpoints. The point x_k converges to a minimum point within the interval $[l, r]$. If the interval contains only one minimum point, it converges to that. In our test cases, several minima exist and one can observe the convergence to one of them. The algorithm is effective in the sense of converging to a local (nonglobal) minimum point for both cases. Another starting interval could have lead to another minimum point. In the end we are certain that the current iterate x_k is not further away than ϵ from a minimum point. Many other stopping criteria like convergence of function value or of the derivative to zero could be used. The current stopping criterion is easy for analysis on the efficiency. One question could be: How many iterations corresponding (derivative) function evaluations are necessary to come closer than ϵ to a minimum point. The bisection algorithm is a typical case of linear convergence with a convergence factor of $\frac{1}{2}$, $\frac{|r_{k+1}-l_{k+1}|}{|r_k-l_k|} = \frac{1}{2}$. This means one can determine the number of iterations necessary for reaching ϵ-convergence:

Table 1. Bisection for functions h and g, $\epsilon = 0.015$

	function h					function g				
k	l_k	r_k	x_k	$h'(x_k)$	$h(x_k)$	l_k	r_k	x_k	$g'(x_k)$	$g(x_k)$
0	3.00	7.00	5.00	-1.80	1.30	3.00	7.00	5.00	-1.80	1.30
1	5.00	7.00	6.00	3.11	0.76	5.00	7.00	6.00	3.11	0.76
2	5.00	6.00	5.50	-1.22	0.29	5.00	6.00	5.50	-1.22	0.29
3	5.50	6.00	5.75	0.95	0.24	5.50	6.00	5.75	0.95	0.24
4	5.50	5.75	5.63	-6.21	0.57	5.50	5.75	5.63	-0.21	0.20
5	5.63	5.75	5.69	-2.64	0.29	5.63	5.75	5.69	0.36	0.20
6	5.69	5.75	5.72	-0.85	0.24	5.63	5.69	5.66	0.07	0.19
7	5.72	5.75	5.73	0.05	0.23	5.63	5.66	5.64	-0.07	0.19
8	5.72	5.73	5.73	-0.40	0.23	5.64	5.66	5.65	0.00	0.19

$$
\begin{aligned}
\mid r_k - l_k \mid &= (\frac{1}{2})^k \times \mid r_0 - l_0 \mid < \epsilon \\
(\frac{1}{2})^k &< \frac{\epsilon}{\mid r_0 - l_0 \mid} \\
k &> \frac{\ln \epsilon - \ln \mid r_0 - l_0 \mid}{ln \frac{1}{2}}
\end{aligned}
$$

The example case would require at least $k = 12$ iterations to reach an accuracy of $\epsilon = 0.01$.

An alternative for finding the zero point of an equation, in our case the derivative, is the so-called method of **Newton**. The idea is that its efficiency is known to be superlinear (e.g. [5]), so it should be faster than bisection. We will analyse its efficiency and effectiveness for the two test cases. In general, the aim of this algorithm is to converge to a point where the derivative is zero. Depending on the starting point x_0 the method may converge to a maximum. It may also not converge at all, for instance when a minimum point does not exist. If x_0 is in the

Algorithm 3 Newt(x_0, f, ϵ)

```
Set $k = 0$,
**while** ($| f'(x_k) |> \epsilon$)
    $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
    $k = k + 1$
End **while**
```

neighbourhood of a minimum point where f is convex, then convergence is guaranteed and the algorithm is effective in the sense of reaching a minimum point. Let us consider what happens for the two test cases. When choosing the starting point x_0 in the middle of the interval $[3, 7]$, the algorithm converges to the closest minimum point for function h and to a maximum point for the function g. This gives rise to introducing the concept of a **region of attraction** of a minimum point x^* as that region of starting points x_0 where the local search procedure converges to point x^*. One can observe here when experimenting further, that when x_0 is close to a minimum point of g the algorithm converges to one of the minimum points. Morever, one should remark that the algorithm requires a safeguard to keep the iterates in the interval $[3, 7]$. This means that if for instance $x_{k+1} < 3$, it should be forced to a value of 3. In that case, also the left point $x = 3$ is an attraction point of the algorithm. Function h is piecewise convex, such that the algorithm always converges to the closest minimum point.

Table 2. Newton for functions h and g, $\epsilon = 0.001$

	function h				function g			
k	x_k	$h'(x_k)$	$h''(x_k)$	$h(x_k)$	x_k	$g'(x_k)$	$g''(x_k)$	$g(x_k)$
0	5.000	-1.795	43.066	1.301	5.000	-1.795	-4.934	1.301
1	5.042	0.018	43.953	1.264	4.636	0.820	-7.815	1.511
2	5.041	0.000	43.944	1.264	4.741	-0.018	-8.012	1.553
3	5.041	0.000	43.944	1.264	4.739	0.000	-8.017	1.553

4. Summary

One of the targets of the educational system on optimisation algorithms is to teach students to think and analyse critically. At least to see through the evolutionary, neural, self learning and replicating humbug. Therefore it is good to start with analysing simple algorithms. This paper aims at being an introductory text for that by showing simple cases that differ in structure and analysing simple algorithms for that.

References

[1] W. P. Baritompa, R.H. Mladineo, G. R. Wood, Z. B. Zabinsky, and Zhang Baoping. Towards pure adaptive search. *Journal of Global Optimization*, 7:73–110, 1995.

[2] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, New York, 1981.

[3] E.M.T. Hendrix and O. Klepper. On uniform covering, adaptive random search and raspberries. *Journal of Global Optimization*, 18:143–163, 2000.

[4] E.M.T. Hendrix, P. M. Ortigosa, and I. García. On success rates for controlled random search. *Journal of Global Optimization*, 21:239–263, 2001.

[5] L.E. Scales. *Introduction to Non-Linear Opimization,*. Macmillan, London, 1985.

[6] A. Törn and A. Zilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, Berlin, 1989.

[7] Z. B. Zabinsky and R. L. Smith. Pure adaptive search in global optimization. *Mathematical Programming*, 53:323–338, 1992.

Proceedings of GO 2005, pp. 133 – 140.

An Adaptive Radial Basis Algorithm (ARBF) for Mixed-Integer Expensive Constrained Global Optimization

Kenneth Holmström

Department of Mathematics and Physics, Mälardalen University, P.O. Box 883, SE-721 23 Västerås, Sweden,
kenneth.holmstrom@mdh.se

Abstract A mixed-integer constrained extension of the radial basis function (RBF) interpolation algorithm for computationally costly global non-convex optimization is presented. Implementation in TOMLAB (http://tomlab.biz) solver rbfSolve is discussed. The algorithm relies on mixed-integer nonlinear (MINLP) sub solvers in TOMLAB, e.g. OQNLP, MINLPBB or the constrained DIRECT solvers (glcDirect or glcSolve). Depending on the initial experimental design, the basic RBF algorithm sometimes fails and make no progress. A new method how to detect when there is a problem is presented. We discuss the causes and present a new faster and more robust Adaptive RBF (ARBF) algorithm. Test results for unconstrained problems are discussed.

Keywords: Expensive, global, mixed-integer, nonconvex, optimization, software, black box.

1. Introduction

Global optimization with emphasis on costly objective functions and mixed-integer variables is considered, i.e., the problem of finding the global minimum to (1) when each function value $f(x)$ takes considerable CPU time, e.g. more than 30 minutes to compute [7, 10, 11, 13].

The Mixed-Integer Expensive (Costly) Global Black-Box Nonconvex Problem

$$\begin{array}{llcccll} & \min\limits_{x} & f(x) & & & & \\ & & & & & & \\ & -\infty < & x_L & \leq & x & \leq & x_U \quad < \infty \\ s/t & & b_L & \leq & Ax & \leq & b_U \\ & & c_L & \leq & c(x) & \leq & c_U, \quad x_j \in \mathbb{N} \;\; \forall j \in \mathrm{I}, \end{array} \tag{1}$$

where $f(x) \in \mathbb{R}$, $x_L, x, x_U \in \mathbb{R}^d$, $A \in \mathbb{R}^{m_1 \times d}$, $b_L, b_U \in \mathbb{R}^{m_1}$ and $c_L, c(x), c_U \in \mathbb{R}^{m_2}$. The variables x_I are restricted to be integers, where I is an index subset of {1,. . . ,d}. Let $\Omega \in \mathbb{R}^d$ be the feasible set defined by the constraints in (1).

Such problems often arise in industrial and financial applications, where a function value could be the result of a complex computer program, or an advanced simulation, e.g., CFD, tuning of trading strategies, or design optimization. In such cases, derivatives are most often hard to obtain (the algorithms discussed make no use of such information) and $f(x)$ is often noisy or nonsmooth. One of the methods for this problem type utilizes radial basis functions (RBF) and was presented by Gutmann and Powell in [4, 13]. The idea of the RBF algorithm is to use radial basis function interpolation to define a utility function. The next point, where the original objective function should be evaluated, is determined by optimizing on this utility function. The combination of our need for efficient global optimization software and the interesting ideas of Gutmann led to the development of an improved RBF algorithm [1] implemented in MATLAB. This method was based on interpolating the function values so far computed by RBF and the algorithm handled cheap box constraints. Global non-costly sub problems were solved with TOMLAB [5] implementations of DIRECT methods [8, 9] combined with good local solvers.

This paper presents an important extension to the method in [1] — the Adaptive-RBF (ARBF) method implemented in TOMLAB [6]. The standard RBF algorithm very often result in points sampled on the boundary, which leads to poor performance and non-convergence. Therefor we propose a one-dimensional search for a suitable target value f_n^* to improve convergence. This leads to a sequence of global optimization problems to be solved in each iteration. In addition the standard RBF algorithm has bad local convergence properties and we suggest new ways to improve local convergence.

The algorithm is described in detail and we analyze its efficiency on the Shekel test problems, that are part of the standard test problem set of Dixon-Szegö [2]. The results show that this improved implementation of the RBF algorithm is very efficient on the standard test problems compared to the standard RBF algorithm.

2. The RBF method

The idea of the RBF algorithm is to use radial basis function interpolation and a measure of 'bumpiness' of a radial function, σ say. A target value f_n^* is chosen that is an estimate of the global minimum of f. For each $x \notin \{x_1, \ldots, x_n\}$ there exists a radial basis function $s_n(x)$ that satisfies the interpolation conditions

$$\begin{array}{rcl} s_n(x_i) & = & f(x_i), \ \ i = 1, \ldots, n, \\ s_n(x) & = & f_n^*. \end{array} \tag{2}$$

The next point x_{n+1} is then calculated as the value of x in the feasible region that minimizes $\sigma(s_n)$. It turns out that the function $x \mapsto \sigma(s_n)$ is much cheaper to compute than the original function.

The smoothest radial basis interpolation is obtained by minimizing the semi-norm [3]

$$s_n = arg \min_s \quad < s, s > . \tag{3}$$

Here, the radial basis function interpolant s_n has the form

$$s_n(x) = \sum_{i=1}^{n} \lambda_i \phi \left(\|x - x_i\|_2 \right) + b^T x + a, \tag{4}$$

with $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$, $b \in \mathbb{R}^d$, $a \in \mathbb{R}$ and ϕ is either cubic with $\phi(r) = r^3$ or the thin plate spline $\phi(r) = r^2 \log r$, see Table 1. The unknown parameters λ_i, b and a are obtained as the solution of the system of linear equations

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}, \tag{5}$$

where Φ is the $n \times n$ matrix with $\Phi_{ij} = \phi \left(\|x_i - x_j\|_2 \right)$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_n^T & 1 \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \cdot \\ \cdot \\ \lambda_n \end{pmatrix}, c = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_d \\ a \end{pmatrix}, F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \cdot \\ \cdot \\ f(x_n) \end{pmatrix}. \tag{6}$$

If the rank of P is $d + 1$, then the matrix $\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix}$ is nonsingular and the linear system (5) has a unique solution [12].

Table 1. Different choices of Radial Basis Functions.

RBF	$\phi(r) > 0$	$p(x)$	m
cubic	r^3	$a^T \cdot x + b$	1
thin plate spline	$r^2 \log r$	$a^T \cdot x$ + b	1
linear	r	b	0
multiquadric	$\sqrt(r^2 + \gamma^2)$		-
Gaussian	$\exp(-\gamma r^2)$		-

2.1 The Radial Basis Algorithm (RBF)

- Find initial set of $n \geq d + 1$ sample points x using experimental design.
- Compute costly f(x) for initial set of n points, best point (x_{Min}, f_{Min}).
- Use the n sampled points to build a smooth RBF interpolation model (surrogate model, response surface model) as an approximation of the f(x) surface.
- Iterate until f_{Goal}, known goal for f(x), achieved, $n > n_{Max}$, MaxCycle iteration cycles with no progress or maximal CPU time used.
  - Find minimum of RBF surface, $s_n(x_{s_n}) = \min_{x \in \Omega} s_n(x)$.
  - In every iteration in sequence pick one of the $N + 2$ step choices.
    1. **Cycle step -1 (InfStep).** Set target value $f_n^* = -\infty$, i.e. solve the global problem
       $$g_n^\infty = \min_{x \in \Omega} \mu_n(x), \tag{7}$$
       where the coefficient $\mu_n(x)$ is an estimate of the new λ-coefficient if the trial x is included in the RBF interpolation
    2. **Cycle step $k = 0, 1, ..., N - 1$ (Global Search).** Define target value $f_n^* \in (-\infty, s_n(x_{s_n})]$ as $f_n^*(k) = s_n(x_{s_n}) - w_k \cdot \left(\max_i f(x_i) - s_n(x_{s_n})\right)$, with $w_k = (1 - k/N)^2$ or $w_k = 1 - k/N$. Solve the global optimization problem
       $$g_n(x_{g_n}^k) = \min_{x \in \Omega} \quad \mu_n(x) \left[s_n(x) - f_n^*(k)\right]^2 . \tag{8}$$
    3. **Cycle step N (Local search).**
       If $s_n(x_{s_n}) < f_{Min} - 10^{-6}|f_{Min}|$, accept x_{s_n} as the new trial point.
       Otherwise set $f_n^*(k) = f_{Min} - 10^{-2}|f_{Min}|$ and solve (8).
  - Compute and validate new $(x, f(x))$, increase n.
  - Update (x_{Min}, f_{Min}) and compute RBF surface.

2.2 Properties of the basic RBF algorithm

Gutmann had the view that the global optimum of the surface, x_{s_n} would be close to the current best point x_{Min}. However, as seen in practice, this is not case. In cycle step N, x_{s_n} should give local convergence. If the point is far from x_{Min}, this is not true. In *rbfSolve* a local optimization solver (SNOPT or NPSOL) has been used, however in almost all cases it has found the global minimum, not the closest local minimum.

Define the number of active variables $\alpha(x)$ as the number of coefficients in the point x that has components close to the bounds in the box, i.e

$$\alpha(x) = \sum_{i=1,\ldots,d} \{\|x_i - x_{L_i}\| \leq \epsilon + \|x_i - x_{U_i}\| \leq \epsilon\}$$

If the target value f_n^* is made sufficiently low, then $\alpha(x) = d$, i.e. all components in the global optimum are on a bound. If $\alpha(x) > 0$ the point is on the boundary for at least one component, and normally is useless for the RBF interpolation. It is difficult to find a suitable f_n^* for any problem. It might very well be the case that f_n^* is too low, making $\alpha(x) > 0$, and the point is on the boundary for at least one component, The choice made in the RBF algorithm is not robust. Solving the global optimization problem for a sequence of decreasing f_n^* values shows that in general, for sufficiently high values x is interior ($\alpha(x) = 0$). Decreasing f_n^* further will result in $\alpha(x) > 0$. Finally the threshold where $\alpha(x) = d$ is reached, and decreasing f_n^* more just results in the same point reached, with all components on the boundary. The above suggests that an adaptive strategy should be tried. Important, using the measure $\alpha(x)$ makes it possible to do adjustments in the algorithmic step without actually computing the costly $f(x)$.

For many problems, for some selection of points, no f_n^* value that gives $\alpha(x) = 0$ exists. When things still work, adding a few points leads to f_n^* values having $\alpha(x) = 0$, i.e. the target value strategy then works fine. An oscillating behavior is often noticed, some iterations have $\alpha(x) = 0$, some not. When the RBF interpolations do generate $\alpha(x) = 0$ rather often, the algorithm works well and is fast converging. However, the basic RBF algorithm is still not working well if trying to achieve many digits of accuracy, local convergence properties need improvement. We have therefor formulated a new adaptive RBF algorithm.

3. The Adaptive Radial Basis Algorithm (ARBF)

- Find initial set of $n \geq d + 1$ sample points x using experimental design.
- Compute costly f(x) for initial set of n points, best point (x_{Min}, f_{Min}).
- Iterate until f_{Goal}, known goal for f(x), achieved, $n > n_{Max}$ or maximal CPU time used.
  - Solve global optimization problem finding minimum of RBF surface.
  - Solve global optimization problem finding maximal distance from any point in the feasible region to the closest sampled point.
  - In every third iteration cycle select one the following three types of search steps:
    1. Global Search Step (G-step)
    2. Global-Local Search Step (GL-step)
    3. Local Search Step (L-step)
  - Compute and validate new $(x, f(x))$, increase n.
  - Update (x_{Min}, f_{Min}) and compute RBF surface.
- Save and return all information to enable warm start, after user evaluation.

3.1 Initial step in every iteration of ARBF

IS1. The **global** minimum on the RBF surface

$$s_n(x_{s_n}^{Glob}) = \min_{x \in \Omega} \quad s_n(x) \tag{9}$$

Given the previously evaluated points $x_1, x_2, ..., x_n$, find the maximal distance from any point in the box to the sampled points x_i (*maximin distance*, Regis and Schoemaker [14]), i.e.

$$\Delta = \max_{x_L \leq x \leq x_U} \min_{1 \leq j \leq n} ||x - x_j||$$

Solve this problem formulating the global non-convex optimization problem:
IS2. The maximal distance Δ from any point in the feasible region Ω to the closest sample point

$$\begin{array}{lccccl} \min\limits_{x,\Delta} & & & -\Delta & & \\ & x_L & \leq & x & \leq & x_U \\ s/t & 0 & \leq & \Delta & \leq & \infty \\ & 0 & \leq & ||x-x_i||^2-\Delta^2 & \leq & \infty, \quad i=1,...,n. \end{array} \tag{10}$$

3.2 The Global Search (G-step) in ARBF

To find a good point first a sequence of global optimization problems are solved. If this phase fails, the global solution $x_{s_n}^{Glob}$ of the RBF surface is considered. If rejected, the point most distance to all the sampled points, as well as to the boundary, is computed by solving a global optimization problem.

G-step Phase 1. For a sequence of decreasing target values $f_{n_k}^*$, solve the global optimization problem

$$g_n(x_{g_n}^k) = \min_{x\in\Omega} \quad \mu_n(x)\left[s_n(x)-f_{n_k}^*\right]^2 \tag{11}$$

giving the global minimum $g_n(x_{g_n}^k)$ for each $f_{n_k}^*$. Find

$$f_n^{\hat{k}} = \min_{\alpha(x_{g_n}^k))=0} f_{n_k}^*$$

If $f_n^{\hat{k}}$ is non-empty, pick the global minimum corresponding to this target value, $x_{g_n}^{\hat{k}}$, as the G-step. Otherwise turn to G-step Phase 2. If $\alpha(x_{g_n}^k))=d$ during the search, there is no point in decreasing $f_{n_k}^*$ further, and the sequence is interrupted. If a minimum with $\alpha(x_{g_n}^k))=0$ is found, a refined set of target values are tried in order to try to decrease the target values $f_{n_k}^*$ further.

G-step Phase 2. Select the global RBF solution $x_{s_n}^{Glob}$ if it is interior, with sufficiently low value and sufficiently far from currently best point x_{Min}. If $\alpha(x_{s_n}^{Glob})=0$, $f_{Min}-s_n(x_{s_n}^{Glob}) \geq 0.01\max(1,f_{Min})$ and $||x_{s_n}^{Glob}-x_{Min}|| > 0.2\Delta$ pick $x_{s_n}^{Glob}$ as the G-step point. Otherwise turn to G-step Phase 3.

G-step Phase 3. Find the point which have the maximal distance to any sample point as well as the boundary, solving the global optimization problem

$$\begin{array}{lccccl} \delta_{IP} = & & \min\limits_{x\,\in\,\Omega,\,\delta} & -\delta & & \\ s/t & & & & & \\ & x_{L_i} & \leq & x_i-\delta/d & & ,\ i=1,...,n. \\ & & & x_i+\delta/d & \leq\ x_{U_i} & ,\ i=1,...,n. \\ & 0 & \leq & \delta & \leq\ \Delta & \\ & 0 & \leq & ||x-x_i||^2-\delta^2 & \leq\ \infty & ,\ i=1,...,n. \end{array} \tag{12}$$

3.3 The Global-Local Search (GL-step) in ARBF

To find a good point first a sequence of global optimization problems are solved. If this phase fails, the global solution $x_{s_n}^{Glob}$ of the RBF surface is considered. If rejected, the InfStep point ($f_n^*=-\infty$) is used.

GL-step Phase 1. For a sequence of decreasing target values $f^*_{n_k}$, solve the global optimization problem

$$g_n(x^k_{g_n}) = \min_{x\in\Omega} \quad \mu_n(x)\left[s_n(x) - f^*_{n_k}\right]^2 \tag{13}$$

giving the global minimum $g_n(x^k_{g_n})$ for each $f^*_{n_k}$. Find all minima which satisfies $\alpha(x^k_{g_n}) = 0$, and among these select the element

$$\mu^{\hat{k}}_n = \max_{\{k:\, \alpha(x^k_{g_n}))=0\}} \mu_n(x^k_{g_n})$$

If $\mu^{\hat{k}}_n$ is non-empty, pick the global minimum $x^{\hat{k}}_{g_n}$ as the GL-step. Otherwise turn to GL-step Phase 2. This choice most often gives a point among the higher up to median $f^*_{n_k}$ values among the global solutions with $\alpha(x^k_{g_n})) = 0$, i.e. a more local search than in G-step Phase 1, where the lowest possible $f^*_{n_k}$ value was used.

GL-step Phase 2. Select the global RBF solution $x^{Glob}_{s_n}$ if it is interior, with a value lower than f_{Min} and not too close to the currently best point x_{Min}

If $\alpha(x^{Glob}_{s_n}) = 0$, $f_{Min} - s_n(x^{Glob}_{s_n}) \geq \epsilon_{s_n} \max(1, f_{Min})$ and $\|x^{Glob}_{s_n} - x_{Min}\| > 10^{-6}\Delta$, pick $x^{Glob}_{s_n}$ as the G-step point. $\epsilon_{s_n} = 10^{-6}$ is currently used. Otherwise turn to G-step Phase 3. This point is often rather close to x_{Min} and help to improve local convergence.

GL-step Phase 3. Set $f^*_n = -\infty$, i.e. solve the global optimization problem (7).

3.4 The Local Search (L-step) in ARBF

Case 1: $s_n(x^{Glob}_{s_n}) < f_{Min} - \epsilon_{s_n}|f_{Min}|$. The global minimum value of the RBF surface is sufficiently lower than f_{Min}. Try to find the **local** minimum on the RBF surface closest to the currently best point x_{Min} . Solve a sequence of local optimization problems with increasing β_k in the range $[0.005, 0.20]$ until a local optimum is found.

$$\begin{array}{llcccc} s_n(x^{Loc}_{s_n}) = & \min\limits_{x\in\Omega} & & s_n(x) & & \\ s/t & (10^{-4}||x_{Min}||)^2 & \leq & ||x - x_{Min}||^2 & \leq & (\beta_k\Delta)^2. \end{array} \tag{14}$$

For a local minimum to be accepted, the distance constraint must not be active. If no local minimum is found, except with active distance constraints, the solution for $\beta_k = 0.05$ is used.

Case 2: $s_n(x^{Glob}_{s_n}) \geq f_{Min} - \epsilon_{s_n}|f_{Min}|$.
In this case the global minimum value of the RBF surface is very close to the currently best function value found. Then there is a danger that the local solution is too close to x_{Min}. Set the target value $f^*_n = f_{Min} - 0.1|f_{Min}|$ and solve

$$\begin{array}{llcccc} g_n(x^{Loc}_{g_n}) = & \min\limits_{x\in\Omega} & & \mu_n(x)\left[s_n(x) - f^*_n(k)\right]^2 & & \\ s/t & (10^{-4}||x_{Min}||)^2 & \leq & ||x - x_{Min}||^2 & \leq & (0.1\Delta)^2. \end{array} \tag{15}$$

4. Test on Shekel problems with steep minima

Tests on number of function evaluations needed to reach the goal f_{Goal} $\frac{f_{Min}-f_{Goal}}{|f_{Goal}|} \leq \epsilon$, with $\epsilon = 10^{-3}$ and $\epsilon = 10^{-5}$ are shown in Table 2. The Shekel test problem has very steep minima, see Figure 1. The initial experimental design used was the Ellipsoid method producing 15 points. The $-$ sign indicates that the algorithm did not converge to the global minimum in 400

function evaluations. For almost all the failures a local minimum was found. For the Shekel 5 problem, the number of G-step Phase 1 and GL-step Phase 1 that produced interior points are shown (row *OK* in the table). Also shown is the iteration from where these steps never got an interior point (row *Iter* in the table). Only very few iterations at the start actually produced interior points, and the convergence is entirely dependent on if these steps produced good enough points to find the steep global minima. The Results show that *ARBFMIP* is about twice as fast *rbfSolve* and solves slightly more of the problems. *ARBFMIP* is also much better in getting local convergence and achieving higher accuracy in the global minimum, as expected.

5. Conclusions

The global search with target values on the RBF surface often does not produce interior points. The cause must be further investigated. If it is not possible to add, delete or change the RBF interpolation or problem formulation in a way that produce interior points, then some other

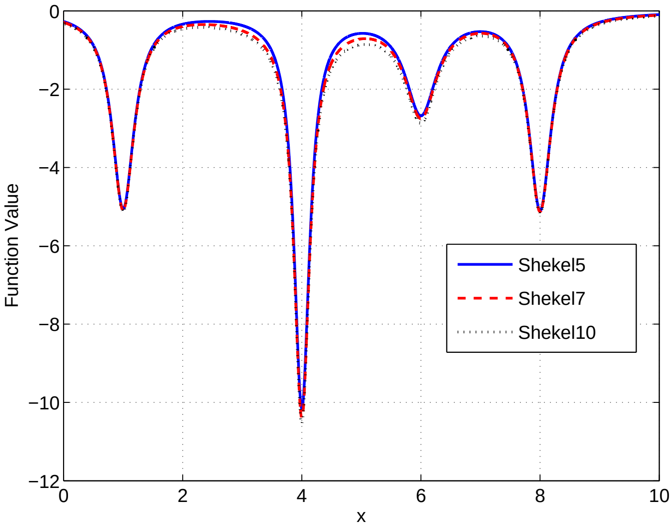


Figure 1. Plot of the Shekel 5, Shekel 7 and Shekel 10, functions along the line segment from $(0, 0, 0, 0)$ to $(10, 10, 10, 10)$.

Table 2. Test of Shekel problems for rbfSolve and ARBFMIP

rand('state')	ϵ	0	1	2	3	4	5	6	7	8	9	Mean	Fail
Shekel 5 with	10^{-3}	74	62	66	-	77	-	-	87	54	-	70	4
ARBFMIP	10^{-5}	84	72	90	-	89	-	-	105	68	-	85	4
G,GL Phase 1	OK	7	7	6	4	5	6	5	6	5	5	5.6	
G,GL bad from	Iter	20	18	11	11	8	60	16	12	23	8	18.7	
rbfSolve	10^{-3}	153	153	-	-	171	-	-	141	129	-	149	5
rbfSolve	10^{-5}	-	195	-	-	-	-	-	165	171	-	177	7
Shekel 7 with	10^{-3}	-	66	69	78	78	-	66	63	65	-	69	3
ARBFMIP	10^{-5}	-	87	96	89	87	-	72	84	74	-	84	3
rbfSolve	10^{-3}	135	123	123	-	123	129	135	141	-	123	129	2
rbfSolve	10^{-5}	-	-	-	-	-	171	171	151		-	164	7
Shekel 10 with	10^{-3}	66	-	66	66	72	90	72	75	-	72	72	2
ARBFMIP	10^{-5}	75	-	72	78	90	96	80	89	-	87	87	2
rbfSolve	10^{-3}	123	123	177	117	117	147	111	159	-	123	133	1
rbfSolve	10^{-5}	159	147	-	171	-	-	153	195	-	141	161	4

surrogate model, e.g. Kriging models, is needed in such iteration steps. Checking for an interior point makes it possible to detect the problem and do something else before computing the costly $f(x)$. The new ARBF algorithm is very efficient when the RBF interpolation has global minima that most often are interior points. It has better local convergence properties than the original RBF algorithm. The experimental version of *ARBFMIP* takes about 2-4 seconds per iteration for up to 200 function values, and 3-7 seconds per iteration if doing 400 function values. The final version should be fast enough for practical use. The use of radial basis interpolation methods for costly (expensive) mixed-integer nonlinear optimization problems is promising.

References

[1] M. Björkman and K. Holmström. Global Optimization of Costly Nonconvex Functions Using Radial Basis Functions. *Optimization and Engineering*, 1(4):373–397, 2000.

[2] L. C. W. Dixon and G. P. Szegö. The global optimisation problem: An introduction. In L. Dixon and G Szego, editors, *Toward Global Optimization*, pages 1–15, New York, 1978. North-Holland Publishing Company.

[3] Hans-Martin Gutmann. On the semi-norm of radial basis function interpolants. *Journal of Approximation Theory*, 2001.

[4] Hans-Martin Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[5] K. Holmström. The TOMLAB Optimization Environment in Matlab. *Advanced Modeling and Optimization*, 1(1):47–69, 1999.

[6] Kenneth Holmström and Marcus M. Edvall. Chapter 19: The TOMLAB optimization environment. In Ludwigshafen Germany Josef Kallrath, BASF AB, editor, *Modeling Languages in Mathematical Optimization*, APPLIED OPTIMIZATION 88, ISBN 1-4020-7547-2, Boston/Dordrecht/London, January 2004. Kluwer Academic Publishers.

[7] Waltraud Huyer and Arnold Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14:331–355, 1999.

[8] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, October 1993.

[9] Donald R. Jones. DIRECT. *Encyclopedia of Optimization*, 2001.

[10] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2002.

[11] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive Black-Box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[12] M. J. D. Powell. The theory of radial basis function approximation in 1990. In W.A. Light, editor, *Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions*, pages 105–210. Oxford University Press, 1992.

[13] M. J. D. Powell. Recent research at Cambridge on radial basis functions. In M. D. Buhmann, M. Felten, D. Mache, and M. W. Müller, editors, *New Developments in Approximation Theory*, pages 215–232. Birkhäuser, Basel, 1999.

[14] Rommel G. Regis and Christine A. Shoemaker. Constrained Global Optimization of Expensive Black Box Functions Using Radial Basis Functions. *Journal of Global Optimization*, 31(1):153–171, 2005.

A Distributed Global Optimisation Environment for the European Space Agency Internal Network

Dario Izzo and Mihály Csaba Markót

Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands.
{Dario.Izzo, Mihaly.Csaba.Markot}@esa.int

Abstract Global optimisation problems arise daily in almost all operational and managerial phases of a space mission. Large computing power is often required to solve these kind of problems, together with the development of algorithms tuned to the particular problem treated. In this paper a generic distributed computing environment built for the internal European Space Agency network but adaptable to generic networks is introduced and used to distribute different global optimisation techniques. Differential Evolution, Particle Swarm Optimisation and Monte Carlo Method have been distributed so far and tested upon different problems to show the functionality of the environment. Support for both simple and multi-objective optimisation has been implemented, and the possibility of implementing other global optimisation techniques and integrating them into one single global optimiser has been left open. The final aim is that of obtaining a distributed global multi-objective optimiser that is able to 'learn' and apply the best combination of the available solving strategies when tackling a generic "black-box" problem.

Keywords: Distributed computing, idle-processing, global optimisation, differential evolution, particle swarm optimisation, Monte Carlo methods.

1. The Distributed Computing Environment

Our distributed computing architecture follows the scheme of a generic server-client model [10]: it consists of a central computer (server) and a number of user computers (clients). The architecture is divided into three layers on both the server and the client side (Figure 1). This promotes the modular development of the whole system: for example, the computation layer of the clients (involving the optimisation solver modules) can be developed and maintained independently of the other client layers.

The main tasks of the server are the following: pre-processing the whole computing (optimisation) problem by disassembling it into subproblems; distributing sets of subproblems among the clients; and generating the final result of the computation by assembling solutions received from the clients.

On the opposite side, the client computers ask for subproblems, solve them and send back the results to the server. As with many current distributed applications (employing common user desktop machines), our approach is also based on the utilisation of the idle time of the clients. More precisely, a client only asks for subproblems when there is no user activity detected either on the mouse or on the keyboard. In our environment we hide the client computations behind a screen saver, similarly to the most widely-known distributed computing project, the SETI@home project (`http://setiathome.ssl.berkeley.edu`).

As shown in Figure 1, the basic functionalities of the individual layers are the same for both the client and the server. The uppermost layers are visible to the client users and for the server administrator, respectively. These layers contain the screen saver (client) and a server

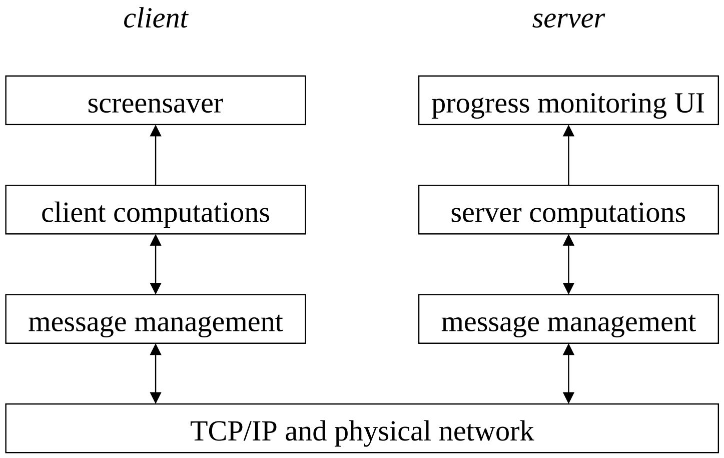


Figure 1. Architecture of the distributed environment.

progress-monitoring user interface (displaying the actual state of the computation process), on the server.

The top-level layers communicate with the real computation layers that are performing the numerical tasks. The computation layers are responsible for disassembling, distributing and assembling the whole computation task (server), and evaluating the subproblems (clients).

The lowermost, so-called message management layers maintain connection with the computation layers, and send and receive the problem and solution 'packages' between the client and the server. This service was implemented by network sockets using the Windows Sockets version 2 Application Programming Interface [12].

The environment was programmed in Visual C++, strongly utilising the advantages of the object-oriented language. The introduction of the detailed architecture is a subject of separate, forthcoming publications (due to its extent); here we restrict ourselves to giving a short overview on the key concepts and building blocks.

The environment provides several data storage classes to program the various solvers in an easy way. The basic data type is called SOL_PAIR; it is the representation of an $(x, f(x)) \subset \mathbb{R}^n \times \mathbb{R}^m$ pair. Both components are implemented as a variable size vector, which allows us to deal with single and multi-objective optimisation problems, constraint satisfaction problems (with no objective function), and virtually every kind of distributable (even not necessarily optimisation!) problems. For population-based solvers, it was particularly useful to have a POPULATION storage class, which is simply a set (list) of SOL_PAIRs. The basic storage types used during the client-server communication are called 'packages': the most important one is a base class called PS_PACKAGE, which is used to derive the specific problem and solution packages for the particular solvers. A package typically involves a data storage object (such as a POPULATION) together with instruction (server) and solution (client) information. All the above classes have member functions which transform the data to and from a stream of characters. The latter data type is used by the message management routines to transfer the data on the network.

The server computation layer is based on a C++ abstract class called SERVER. The particular servers (implementing various strategies to solve the whole problem) are derived from this class. On the other hand, each client computation layer contains the set of available solvers. The solvers are derived from an abstract SOLVER class. This means that the environment can be arbitrarily extended by adding server and solver classes. Moreover, since each problem package (sent out to a client) is always solved by one specified solver, various server strategies can be tested without changing the client application. It is very important that the solution packages should always be kept in a consistent state by the solver: if the computation is interrupted by a user action, a fraction (but still useful part) of the whole solution is sent back to the server.

The optimisation problems are implemented as instances of an OPT_PROB class: in practice, this means that for every particular optimisation problem the user have to provide the following routines: function evaluation at a given point, random generation of a feasible point, feasibility checking of a given point (and its substitution with a feasible point in case of infeasibility), and a routine implementing a preference relation between every two feasible solutions.

2. The global optimisation algorithms

In the present version of the software, there are three different optimisation solvers available:

1. Monte–Carlo search (MC, [7]). In this method, the server requests the clients to create a certain number of independent random samples from the feasible search space (using uniform distribution in each variable) and sends the most promising solution back to the server. The server maintains a set of the best solutions received (Pareto optimality criteria are used in the case of multi-objective optimisation problems).

2. Differential Evolution (DE). This novel optimisation algorithm is based on updating each element of a set (population) of feasible solutions by using the difference of two other randomly selected population elements. The method is described in detail in [9]. In our environment the DE server updates a fixed-size main population, and each problem package consists of a request to evolve a randomly-selected subpopulation for a specified number of iterations. The main population is then updated by the returned population with respect to the preference relation of the particular problem.

3. Particle Swarm optimisation (PSO). This is another population-based algorithm inspired by the social behaviour of bird or fish flockings [5]. In a PSO method, each element (particle) evolves by taking the combination of the current global best and individual best solutions into account. In the proposed distributed version of the PSO method, the server updates a main population and sends request to the clients to evolve a random subpopulation (in the same way as for the DE algorithm). This distributed variant shows similarities with the Multi-Swarm optimisation techniques [1] developed as a possible improvement of the PSO algorithm.

3. The optimisation problems

Originally, our method was designed to deal with bound-constrained optimisation problems:

$$\min f(x) \tag{1}$$

$$\text{subject to } x \in D, \tag{2}$$

where $D = [L_i, U_i], L_i, U_i \in \mathbb{R}$, and the objective function $f : \mathbb{R}^n \to \mathbb{R}^m$ is continuous in D. Nevertheless, the currently implemented solvers allow us to handle a certain group of inequality-constrained problems as well: namely, optimisation problems which, in addition to (2), have further inequality constraints in the form of

$$g_{i,1}(x_1, \ldots, x_{i-1}) \leq x_i \leq g_{i,2}(x_1, \ldots, x_{i-1}),\ i = 2, \ldots, n, \tag{3}$$

where the exact upper bound of $g_{i,1}$ and the exact lower bound of $g_{i,2}$ can be determined in a machine computable form (e.g. as an expression or a subroutine) for all $x_j \in [L_j, U_j], j = 1, \ldots, i-1$. This property allows us to replace a infeasible solution with a 'close' feasible solution, e.g. one located on the boundary of the feasible set. (Note, that the radio occultation problem below can be formalized as a constrained problem in the above form: when generating feasible satellite orbits, the orbital elements *eccentricity* and *orbital period* are bounded by a function of the *semi-major axis*.)

We performed the numerical tests on the following hard test problems:

1. SAT: The radio-occultation problem described in [4] in detail. This is an optimisation problem of satellite constellations with a complex objective function structure. The optimisation has a dual objective: maximising the number of satellite occultations while distributing the occultations as uniformly as possible on the latitudes. This last objective is described by the standard deviation of the number of occultations occurring at different latitude stripes.

2. RB: The generalisation of the Rosenbrock global optimisation test function [8] given by minimising
$$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2). \tag{4}$$
We have used $n = 50$ and $x_i \in [-5.12, 5.12],\ i = 1, \ldots, n$.

3. LJ: A potential energy minimisation problem for the Lennard-Jones atom cluster for $d = 38$ atoms [11]. The potential is given by
$$f(p) = \sum_{1 \le i < j \le d} 4((1/r_{ij})^{12} - (1/r_{ij})^6), \tag{5}$$
where $p_i = (x_i, y_i, z_i),\ i = 1, \ldots, d$ is the location of the ith atom, and r_{ij} is the Euclidean distance between atoms i and j. We have used $x_1 = y_1 = z_1 = y_2 = z_2 = z_3 = 0$ and $x_i \in [0, 6],\ y_i, z_i \in [-3, 3]$ for all other variables. (Thus, (5) with $d = 38$ corresponds to an $n = 108$–dimensional problem.) This problem has important practical generalisations and it serves as a good test case for parallel and distributed solvers. The chosen problem instance is perhaps the most challenging one in the range of $1 \le n \le 50$. (Note that we did not intend to improve the best existing solution – this would definitely require far more sophisticated algorithms and problem formulation.)

4. Preliminary test results

We solved the above problems with each solver 10 times. The solver parameters related to the distributed implementations were the following:

- Population-related settings of the DE and PSO solvers:
    - The size of the main population was set to $MP = 5n$ for all problems.
    - The size of the subpopulations was set to $SP = MP/5$ for problems SAT and RB, and to $SP = MP/10$ for problem LJ. The reason for the latter setting was that we had to limit the size of problem and solution packages to about 30Kbytes in order to keep the network communication traffic within certain bounds.
    - The allowed number of iterations for evolving the subpopulations was chosen to be $IT = 2000$ for problems RB and LJ, and to $IT = 200$ for problem SAT.
- In the case of the MC solver, the allowed number of random sample generation per package was $IT \cdot SP$.
- For all solvers, the number of function evaluations was limited to 240 000, 5 000 000, and 5 400 000 for problems SAT, RB, and LJ, respectively.

The further control parameters of the DE and PSO algorithms were the default values taken from the implementations [2] and [3], respectively. For DE, we employed the algorithm variant cited as 'DE1' in the above reference.

The computations were performed during normal working days at the European Space Research and Technology Centre (ESTEC). The client application (hidden behind the screen saver) was installed on nine Windows–XP desktops. The sum of the CPU frequencies of the computers was approximately 15.1 GHz, which corresponds to a double-precision theoretical peak performance of about 15.1 Gflops. The results are summarized in Table 1 with respect to the **best** achieved objective function values. To measure the efficiency for the multi-objective SAT problem, we used a further criterion ([4]) in order to compare two solutions. (The variance of the individual test results was small for all strategies, thus, the displayed values can serve as a valid base of a comparison.) The last line of the table shows the previously known best solutions. These values come from [4] for SAT, and from [11] for LJ, respectively. The value given for RB is the known global minimum.

Table 1. The best solutions found during the test runs

solver	*SAT*	*RB*	*LJ*
MC	(1 134, 6.20)	1.574e+5	-10.16
DE	(2 722, 4.15)	24.98	-25.67
PSO	(2 174, 4.78)	27.20	-27.19
best known	(1 535, 7.78)	0	-173.93

Summarising the results, we can state that on the SAT and RB problems the DE algorithm outperformed the other two methods, while on the LJ problem the PSO method worked best. As we expected, these more sophisticated methods behaved far more efficiently than the MC search. In particular, on the SAT problem the Differential Evolution resulted in a big improvement on the previously known best solution (obtained by Monte–Carlo search, [4]). Our main future task is to find suitable distributed generalisations of these (and other) solvers in order to reach further performance improvements.

5. Conclusion and future research

Besides our future plans to extend the system with further global optimisation methods, the most promising way of improving the performance of the distributed environment is to employ the available solvers in an intelligent, co-operative way. This idea requires the comparison of the behaviour of the different solvers. We plan to develop and investigate a set of heuristics to direct the allocation of the packages and the selection of solvers. One such heuristic indicator can be the average (or expected) effort needed to improve the best existing solution by a unit amount, while using a given number of function evaluations and solver strategy. This indicator can be continuously updated for each solver in running time, and can be used as the basis measurement for further decisions.

References

[1] T. Blackwell and J. Branke, "Multi-swarm optimisation in Dynamic Environments", Proceedings of the Applications of Evolutionary Computing, EvoWorkshops, 2004, pp. 489–500.

[2] `http://www.icsi.berkeley.edu/Λstorn/code.html`

[3] `http://www.engr.iupui.edu/Λshi/pso.html`

[4] D. Izzo, M.Cs. Markót, and I. Nann, “A distributed global optimiser applied to the design of a constellation performing radio-occultation measurements". Proceedings of the 15th AAS/AIAA Space Flight Mechanics Conference, Copper Mountain, Colorado, 2005.

[5] J. Kennedy and R.C. Eberhart, “Particle swarm optimisation", Proceedings of the IEEE international conference on neural networks, 1995, Vol. IV, pp. 1942–1948. IEEE service center, Piscataway, NJ.

[6] J. Lampinen, “Differential Evolution — New Naturally Parallel Approach for Engineering Design optimisation". In: Barry H.V. Topping (ed.): Developments in Computational Mechanics with High Performance Computing. Civil-Comp Press, Edinburgh, 1999, pp. 217–228.

[7] N. Metropolis and S. Ulam, “The Monte Carlo Method", J. Amer. Stat. Assoc. (44), 1949, pp. 335–341.

[8] H.H. Rosenbrock, “An automatic method for finding the greatest or least value of a function. Computer Journal (3), 1960, pp. 175–184.

[9] R. Storn and K. Price, “Differential Evolution - a Simple and Efficient Heuristic for Global optimisation over Continuous Spaces", J. Global optimisation (11), 1997, pp. 341-359.

[10] A.S. Tanenbaum, *Computer Networks*, Prentice Hall, 2003.

[11] D.J. Wales and J.P.K. Doye, “Global optimisation by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 Atoms", J. Phys. Chem. A. (101), 1997, pp. 5111–5116.

[12] Windows Sockets 2 API documentation,
`ftp://ftp.microsoft.com/bussys/winsock/winsock2/WSAPI22.DOC`

Variable Neighbourhood Search for the Global Optimization of Constrained NLPs

Leo Liberti[1] and Milan Dražić[2]

[1] *DEI, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy,* liberti@elet.polimi.it

[2] *Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia and Montenegro,* mdrazic@matf.bg.ac.yu

Abstract We report on the theory and implementation of a global optimization solver for general constrained nonlinear programming problems based on Variable Neighbourhood Search, and we give comparative computational results on several instances of continuous nonconvex problems. Compared to an efficient multi-start global optimization solver, the VNS solver proposed appears to be significantly faster.

Keywords: VNS, global optimization, nonconvex, constrained, NLP.

1. Introduction

This paper describes a Variable Neighbourhood Search (VNS) solver for the global solution of continuous constrained nonlinear programming problems (NLPs) in general form:

$$\left.\begin{array}{rrcl} \min\limits_{x\in\mathbb{R}^n} & & f(x) & \\ \text{s.t.} & l \leq & g(x) & \leq u \\ & x^L \leq & x & \leq x^U. \end{array}\right\} \qquad (1)$$

In the above formulation, x are the problem variables. $f : \mathbb{R}^n \to \mathbb{R}$ is a possibly nonlinear function, $g : \mathbb{R}^n \to \mathbb{R}^m$ is a vector of m possibly nonlinear functions, $l, u \in \mathbb{R}^m$ are the constraint bounds (which may be set to $\pm\infty$ as needed), and $x^L, x^U \in \mathbb{R}^n$ are the variable bounds.

Previous work on Variable Neighbourhood Search applied to global optimization was restricted to box-constrained NLPs ($m = 0$ in the above formulation) [19]. To the best of our knowledge, a VNS solver for constrained global optimization targeted at problems in general form (1) has not been implemented yet. It is worth noting, however, that the box-constrained VNS solver described in [19] is currently being tested on a reformulation of constrained problems based on penalization of explicit constraints.

2. The Variable Neighbourhood Search algorithm

Variable Neighbourhood Search (VNS) is a relatively recent metaheuristic which relies on iteratively exploring neighbourhoods of growing size to identify better local optima [6–8]. More precisely, VNS escapes from the current local minimum x^* by initiating other local searches from starting points sampled from a neighbourhood of x^* which increases its size iteratively until a local minimum better than the current one is found. These steps are repeated until a given termination condition is met.

VNS has been applied to a wide variety of problems both from combinatorial and continuous optimization. Its early applications to continuous problems were based on a particular problem structure. In the continuous location-allocation problem the neighbourhoods are defined according to the meaning of problem variables (assignments of facilities to custmers, positioning of yet unassigned facilities and so on) [3]. In bilinearly constrained bilinear problems the neighbourhoods are defined in terms of the applicability of the successive linear programming approach, where the problem variables can be partitioned so that fixing the variables in either set yields a linear problem; more precisely, the neighbourhoods of size k are defined as the vertices of the LP polyhedra that are k pivots away from the current vertex [6]. However, none of the early applications of VNS to continuous problems was ever designed for solving problems in general form (1).

The first VNS algorithm targeted at problems with fewer structural requirements, namely, box-constrained NLPs, was given in [19] (the paper focuses on a particular class of box-constrained NLPs, but the proposed approach is general). Since the problem is assumed to be box-constrained, the neighbourhoods arise naturally as hyperrectangles of growing size centered at the current local minimum x^*.

Algorithm 4 The VNS algorithm.

```
Input: maximum number of neighbourhoods $k_{\max}$, number of local searches in each neighbourhood $L$.
loop
  Set $k \leftarrow 1$, pick random point $\tilde{x}$, perform a local search to find a local minimum $x^*$.   (†)
  while $k \leq k_{\max}$ do
    Consider a neighbourhood $N_k(x^*)$ of $x^*$ such that $\forall\ k > 1\ (N_k(x^*) \supset N_{k-1}(x^*))$.
    for $i = 1$ to $L$ do
      Sample a random point $\tilde{x}$ from $N_k(x^*)$.
      Perform a local search from $\tilde{x}$ to find a local minimum $x'$.   (†)
      If $x'$ is better than $x^*$, set $x^* \leftarrow x'$, $k \leftarrow 0$ and exit the FOR loop.
    end for
    Set $k \leftarrow k + 1$.
    Verify termination condition; if true, exit.
  end while
end loop
```

In Algorithm 4, the termination condition can be based on CPU time, number of non-improving steps and so on.

We implemented Algorithm 4 so that steps (†), i.e. the local search phases, are carried out by an SQP algorithm which is capable of locally solving constrained NLPs.

The definition of the neighourhoods may vary. Consider hyperrectangles $H_k(x)$, centered at x and proportional to the hyperrectangle $x^L \leq x \leq x^U$ given by the original variable bounds, such that $H_{k-1}(x) \subset H_k(x)$ for each $k \leq k_{\max}$. Letting $N_k(x) = H_k(x)$, sampling becomes extremely easy. There is a danger, though, that sampled points will actually be inside $H_{k-1}(x)$, which had already been explored at the previous iteration. Even though the likelihood of this situation arising lessens as the dimension of the Euclidean space increases (since the higher the dimension, the higher the ratio of the volume of $H_k(x)$ to the volume of $H_{k-1}(x)$), we would like to make sure that the sampled points are outside $H_{k-1}(x)$.

Naturally, taking $N_k(x) = H_k(x) \backslash H_{k-1}(x)$ solves this particular difficulty. Sampling in $H_k(x) \backslash H_{k-1}(x)$ is not as straightforward as sampling in $H_k(x)$, however. Let τ be the affine map sending the hyperrectangle $x^L \leq x \leq x^U$ into the unit L_∞ ball (i.e., hypercube) B centered at 0. Let $r_k = \frac{k}{k_{\max}}$ be the radii of the balls B_k (centered at 0) such that $\tau(H_k(x)) = B_k$ for each $k \leq k_{\max}$. In order to sample a random vector $\tilde{x}$ in $B_k \backslash B_{k-1}$ we proceed as follows:

1. sample a random direction vector $d \in \mathbb{R}^n$;
2. normalize d (i.e., set $d \leftarrow \frac{d}{||d||_\infty}$);
3. sample a random radius $r \in [r_{k-1}, r_k]$ yielding a uniformly distributed point in the shell;
4. let $\tilde{x} = rd$.

Finally, of course, we set $\tilde{x} \leftarrow \tau^{-1}(\tilde{x})$. With this construction, we obtain $\tilde{x} \in H_k(x) \backslash H_{k-1}(x)$.

3. The implementation

The search space is defined as the hyperrectangle given by the set of variable ranges $x^L \le x \le x^U$. At first we pick a random point $\tilde{x}$ in the search space, we start a local search and we store the local optimum x^*. Then, until k does not exceed a pre-set k_{max}, we iteratively select new starting points $\tilde{x}$ in an increasingly larger neighbourhood $N_k(x^*)$ and start new local searches from $\tilde{x}$ leading to local optima x'. As soon as we find a local optimum x' better than x^*, we update $x^* = x'$, re-set $k = 1$ and repeat. Otherwise the algorithm terminates.

For each $k \le k_{\text{max}}$ we consider hyperrectangles $H_k(x^*)$ proportional to $x^L \le x \le x^U$, centered at x^*, whose sides have been scaled by $\frac{k}{k_{\text{max}}}$. More formally, let $H_k(x^*)$ be the hyperrectangle $y^L \le x \le y^U$ where, for all $i \le n$,

$$\begin{aligned} y_i^L &= x_i^* - \frac{k}{k_{\text{max}}}(x_i^* - x_i^L) \\ y_i^U &= x_i^* + \frac{k}{k_{\text{max}}}(x_i^U - x_i^*). \end{aligned}$$

This construction forms a set of hyperrectangular "shells" centered at x^* and proportional to $x^L \le x \le x^U$. As has been mentioned above, we define each neighbourhood $N_k(x^*)$ as $H_k(x^*) \backslash H_{k-1}(x^*)$.

The main solver parameters control: the minimum neighbourhood size, the number of sampling points and local searches started in each neighbourhood (L in Algorithm 4), an ε tolerance to allow moving to a new x^* only when the improvement was sufficiently high, and the maximum CPU time allowed for the search.

The solver was coded within the *oo$\mathcal{OPS}$* optimization software framework [17]. *oo$\mathcal{OPS}$* allows global optimization solvers to be deployed quickly and efficiently by offering an API which is very rich in functionality. Solvers can call each other as black-box procedures; the solver library is still modest but growing, including SNOPT [5], the NAG library NLP solver [21], `lp_solve` as local solvers and sBB [12, 13], SobolOpt [10] and the VNS solver described in this paper as global solvers; a rich symbolic computation library is provided, whose capabilities extend to the symbolic computation of derivatives and automatic simplification of expressions, as well as to generating a convex relaxation of the problem at hand.

4. Computational results

In Table 1 we have collected a number of test instances of various constrained NLPs (small, medium and large sized, ordered by number of problem variables) and reported solution times of the VNS solver described in this paper versus those obtained with the SobolOpt solver [10], a Multi Level Single Linkage algorithm based on low-discrepancy Sobol' sequences sampling. In both cases, the local NLP solver used is SNOPT [5]. Both solvers managed to locate the global optima in all test instances. The instances were solved on a Pentium IV 2.66MHz CPU with 1GB RAM, running Linux. All parameters were set to their default values. k_{max} was set to 10 for most of the test instances, apart from the largest ones (the last four in Table 1), where it was set to 50.

The `griewank-2` instance is a modified Griewank function described in [18]. `sixhump` is a classic test function for global optimization algorithm. `sntoy` is the "toy problem" found in the SNOPT documentation [5]. `bilinear-eg3` was taken from [14]: it is a MINLP with 1 binary variable, reformulated to continuous subject to $x^2 = x$. `haverly`, `ben-tal4`, `example4` and `foulds3` are instances of bilinear multi-quality blending problems arising from the oil industry. `kissing-24_4` solves the Kissing Number problem in 4 dimensions. `lavor50` and `more4` are instances from the Molecular Distance Geometry Problem; although `more4` has more atoms and nonlinear terms, the particular way in which the instance `lavor4` was generated makes it harder to solve.

Instance	From	N	C	T	VNS	SobolOpt
`griewank-2`	[18] (Test function)	2	0	4	0.005*	1.03
`sixhump`	[22] (Classic test function)	2	0	6	0.006	0.001*
`sntoy`	[5] (SNOPT test problem)	4	3	5	7.01	2.64*
`bilinear-eg3`	[14] (MINLP test problem)	6	5	15	0.09	0.03*
`haverly`	[9] (Haverly's pooling problem)	9	8	6	0.01*	0.04
`ben-tal4`	[2] (Blending problem)	10	8	6	0.01*	0.44
`example4`	[1] (Blending problem)	26	35	48	0.62*	1.51
`kissing-24_4`	[16] (Kissing Number problem)	97	300	1200	51.92*	213.70
`lavor50`	[11] (Molecular conformation)	150	0	16932	707.15*	3153.02
`foulds3`	[4] (Blending problem)	168	48	136	0.46*	12.65
`more4`	[20] (Molecular conformation)	192	0	45288	418.818*	2903.38

Table 1. Computational results comparing the VNS with the SobolOpt solvers in *oo*$\mathcal{OPS}$. User CPU timings are in seconds. Values marked with * denote the best timings.

5. Conclusion

We presented a new global optimization solver for constrained NLPs based on the VNS algorithm. The computational comparison with an efficient Multi-Level Single Linkage (MLSL) algorithm called SobolOpt seems to show that in general VNS performs better than MLSL. Future work will be two-fold: on one hand, another VNS global solver based on box-constrained reformulation of constrained NLPs based on penalization of explicit constraints is being tested so that more relevant comparative computational data can be gathered. On the other hand, we plan to take into account the explicit constraint structure of the problem even in the global phase, and not only in the local phase as is currently the case. The latter development should be beneficial particularly for those problems for which the local solver has trouble finding a feasible starting point.

Acknowledgments

We are particularly thankful to Dr. Sergei Kucherenko for providing the code for the SobolOpt MLSL solver.

References

[1] N. Adhya, M. Tawarmalani, and N.V. Sahinidis. A Lagrangian approach to the pooling problem. *Industrial and Engineering Chemistry Research*, 38:1956–1972, 1999.

[2] A. Ben-Tal, G. Eiger, and V. Gershovitz. Global minimization by reducing the duality gap. *Mathematical Programming*, 63:193–212, 1994.

[3] J. Brimberg and N. Mladenović. A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Studies in Location Analysis*, 10:1–12, 1996.

[4] L.R. Foulds, D. Haughland, and K. Jornsten. A bilinear approach to the pooling problem. *Optimization*, 24:165–180, 1992.

[5] P.E. Gill. *User's Guide for SNOPT 5.3*. Systems Optimization Laboratory, Department of EESOR, Stanford University, California, February 1999.

[6] P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, 130:449–467, 2001.

[7] P. Hansen and N. Mladenović. Variable neighbourhood search. In P. Pardalos and M. Resende, editors, *Handbook of Applied Optimization*, Oxford, 2002. Oxford University Press.

[8] P. Hansen and N. Mladenović. Variable neighbourhood search. In F.W. Glover and G.A. Kochenberger, editors, *Handbook of Metaheuristics*, Dordrecht, 2003. Kluwer.

[9] C.A. Haverly. Studies of the behaviour of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 25:19–28, 1978.

[10] S. Kucherenko and Yu. Sytsko. Application of deterministic low-discrepancy sequences to nonlinear global optimization problems. *Computational Optimization and Applications*, 30(3):297–318, 2004.

[11] C. Lavor. On generating instances for the molecular distance geometry problem. In Liberti and Maculan [15].

[12] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[13] L. Liberti. Writing global optimization software. In Liberti and Maculan [15].

[14] L. Liberti. Linearity embedded in nonconvex programs. *Journal of Global Optimization*, (to appear) 2004.

[15] L. Liberti and N. Maculan, editors. *Global Optimization: from Theory to Implementation*. Springer, Berlin, (to appear).

[16] L. Liberti, N. Maculan, and S. Kucherenko. The kissing number problem: a new result from global optimization. In L. Liberti and F. Maffioli, editors, *CTW04 Workshop on Graphs and Combinatorial Optimization*, volume 17 of *Electronic Notes in Discrete Mathematics*, pages 203–207, Amsterdam, 2004. Elsevier.

[17] L. Liberti, P. Tsiakis, B. Keeping, and C.C. Pantelides. *ooOPS*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, January 2001.

[18] M. Locatelli. A note on the griewank test function. *Journal of Global Optimization*, 25:169–174, 2003.

[19] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving a spread-spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operations Research*, 151:389–399, 2003.

[20] J.J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7:814–836, 1997.

[21] Numerical Algorithms Group. *NAG Fortran Library Manual Mark 11*. 1984.

[22] E.M.B. Smith. *On the Optimal Design of Continuous Processes*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, October 1996.

Obtaining the globe optimization of set-covering based p-center problems

Fuh-Hwa Franklin Liu, Chi-Wei Shih

Department of Industrial Engineering and Management, National Chiao Tung University, Hsin Chu, Taiwan 300,
fliu@mail.nctu.edu.tw

Abstract In this paper we introduce a procedure with polynomial complexity for solving the set-covering-based p-center problem that is inspired from *BsearchEx* (Elloumi et al., 2004). We present our new formulation (*PC-SC2*) and our new efficient exact procedure, slim bisecting search, *SBsearch*. Furthermore, we show how to simplify the continuous problem with the same distance matrix and increasing p value. The computational results for *SBsearch* are compared with other existing procedures.

Keywords: p-center problem, Network flow, Mathematical programming, Minimax problem, Set covering problem.

1. Introduction

Locating fixed facilities throughout the logistics network is an important decision problem that gives form, structure and shape to the entire logistics system. Location decisions involve determining the number, location and size of the facilities to be used. Facility location models can be classified under four main topics p-center problem, p-median problem, location set covering problem, maximum covering location problem, see Owen and Daskin (1998). The location of emergency service facilities such as hospitals or fire stations is frequently modeled by the p-center problem. The p-center problem is NP-hard; see Kariv and Hakimi (1979) and Masuyama et al. (1981).

Let N be the number of clients, M be the number of potential sites or facilities, and d_{ij} be the distance from client i to facility j. The p-center problem consists of locating p facilities and assigning each client to its closest facility so as to minimize the maximum distance between a client and the facility it is assigned to. Many authors consider the particular case where the facilities are identical to the clients, i.e., $N = M$, and distances are symmetric and satisfy triangle inequalities. We call this particular case the *symmetric* p-center problem.

Main mathematical location methods may be categorized as heuristic and exact. Exact methods refer to those procedures with the capability to guarantee either a mathematically globe optimum solution to the location problem or at least a solution of known accuracy; see Drezner (1984), Handler (1990) and Daskin (1995). In many respects, this is an ideal approach to the location problem; however, the approach can result in long computer running times, huge memory requirements, and a compromised problem definition when applied to practical problems.

In this paper we introduce a procedure with polynomial complexity for solving the set-covering-based p-center problem that is inspired from *BsearchEx* (Elloumi et al., 2004). In Section 2, we present our new formulation (*PC-SC2*) and our new efficient exact procedure, slim bisecting search, *SBsearch*. Furthermore, we show how to simplify the continuous problem with the same distance matrix and increasing p value. In Section 3, we present computational results for *SBsearch* and make comparison with other past research. Conclusions of this approach are outlined in Section 4.

2. The Formulation PC-SC2 and the Efficient Exact Procedure

The formulation (*PC-SC*) due to *BsearchEx* (Elloumi et al., 2004) to solve the p-center problem, which is based on its well-known relation to the set-covering problem, by using a polynomial algorithm for computing a tighter lower bound and then solving the exact solution method. In the paper, the authors show that its linear programming relaxation provides a lower bound tighter than the classical p-center (*PC*) formulation, the lower bound can be computed on polynomial time, their method outperforms the running time of other recent exact methods by an order of magnitude, and it is the first one to solve large instances of size up to $N = M$=1817.

Though the formulation (*PC-SC*) performs better than does formulation (*PC*) for given values of the lower bound and the upper bound, it is hard to solve the large scale problem by directly solving (*PC-SC*) within reasonable time limit. The authors proposed two algorithms to obtain the optimal solution, with complex programming procedure, complicated heuristics, and difficult concept in linear programming such as reduced cost. In this paper, we introduce a modified formulation (*PC-SC2*) and an easier repeating procedure, *SBsearch* to transform a large scale problem into several small scale problems, and then obtain the optimal solution within reasonable time limit. Let $D^{min} = D^0 \quad < D^1 \quad < \quad D^2 \quad < \ldots < \quad D^{K-1} < \quad D^K = D^{max}$ be the sorted different values in the distance matrix. The formulation (*PC-SC2*) is the following:

(*PC-SC2*)

$$\min \ z^k \tag{1}$$

$$s.t. \qquad \sum_{j=1}^{M} y_j = p; \tag{2}$$

$$z^k + \sum_{j:d_{ij}<D^k} y_j \geq 1, \qquad i = 1, 2, ..., N; \tag{3}$$

$$z^k \in \{0\ ,\ 1\}\ ; \tag{4}$$

$$y_j \in \{0\ ,\ 1\}\ , \qquad j = 1, 2, ..., M \tag{5}$$

where y_j and z^k are binary decision variables. Let the superscript "*" denotes the optimal solution of the decision variable. y_j^*=1 if and only if facility j is open, and z^{k*}=0 only if it is possible to choose p facilities and cover all the clients i within the radius D^{k-1}. Constraint (2) limits the number of open facilities to p; constraints (3) mean that, for a given k, z^{k*}=0, if and only if all clients can be served at a distance strictly lower than D^k.

In the optimal solution of (*PC-SC2*), note that z^{k*}= 0 implies z^{k+1} = z^{k+2} = … = z^K = 0. Similarly, z^k = 1 implies z^{k-1} = z^{k-2} = … = z^1 =1. Since z^{k*}= 1 and $z^{(k+1)*}$= 0 implies the optimal min-max value Δ_p^* is the exact solution of the p-center problem, our procedure, based on the bisecting search method, can be considered solving a series of integer linear programming formulation (*PC-SC2*), at most O(log$_2$(*MN*)) integer linear programs. Using the following p-SBsearch procedure, one would obtain the optimal solution for p-center problem, Δ_p^* which is equal to the L_p^* largest distance of the series $D^0, \ldots, D^K$. D^L and D^U are respectively the lower and upper bounds for searching the optimal solution in each iteration.

Step 1 is using bisecting search method. Step 2 is performing the (*PC-SC2*) to obtain the preliminary optimal solution z^k. Step 3 is performed to check the p-center problem optimality of the current solution z^k. Until reaching the optimality, we continue to solve (*PC-SC2*) by updating the upper bound D^U and the lower bound D^L.

```
p-SBsearch
Initialization: given $p$, $D^0, \dots, D^K$, set $L$ = 0, $U = K$ and $\Delta_p^* = D^U$
Step 1. $k$=$\lfloor$(L+U)/2$\rfloor$.
Step 2. Solving (PC-SC2) with $D^k$ to obtain the optimal solution $z^{k*}$.
Step 3. If $z^{k*}$=1, then
Step 3.1. If $z^{k+1}$=0, then STOP, set $\Delta_p^* = D^L$, $L_p^* = L$ and $U_p^* = U$ else, let $L = k$, goto step 1.
If $z^k$=0, then
Step 3.2. If $z^{k-1}$=0, then STOP, set $\Delta_p^* = D^L$, $L_p^* = L$ and $U_p^* = U$, else, let $U = k$, goto step 1.
```

3. Computational Results for p-SBsearch

The procedure was implemented with the code written by C++ and the MIP solver of CPLEX 7.1. We use a notebook with 512 MB of RAM and Intel P-M 1.30 GHz of CPU. The *time limit* of CPLEX is set to 3600 seconds, so the solution of sub-problem stops if no integer solution is found after one hour of CPU time. The results of the comparison on 40 OR-Lib (Beasley 1990) p-median instances are given in Table 1. The first three columns characterize the instance, and the optimal radius is in column 4. Columns 5, 6 and 7 are quoted from Elloumi et al. (2004). Column 8 gives the CPU time of our procedure. Even if it is not straightforward to compare CPU times on different machines, we can show the maximum and the average CPU time as indication in Table 1.

Table 2 gives the results for TSP-Lib (Reinelt, 1991) instances and makes comparison of our procedure with Elloumi et al. (2004). The first three columns characterize the instance. Columns 4 through 8 give the results of algorithm *Bsearch* and *BsearchEx* (2004). Columns *LB** and *UB** give the lower bound and upper bound obtained by *Bsearch*, and Column *cpu1* is the CPU time devoted to *Bsearch*. Column *Opt* gives the optimal solution or the best found solution obtained by *BsearchEx*, and Column *cpu2* is the CPU time devoted to *BsearchEx*.

Columns 9 through 12 give the results of our procedure. There is tradeoff between solution time and the preciseness of solution in large scale problem. Based on the updating of *max* and *min* of out procedure, we could set the bound tolerance in advance to obtain a narrow solution bound in shorter CPU time. If the relative bound tolerance $(D^{max} - D^{min})/D^{min}$ does not exceed 5% for any sub-problem, we stop the procedure and record the current bound. Column *5% Bound* gives the results of 5% bound tolerance, and Column *cpu* is the CPU time of 5% bound tolerance. Column *Opt* gives the results of our procedure, and Column *cpu2* is the CPU time of our *p-SBsearch* procedure. If the optimal solution is not reached in an hour, set z^k = 1 and $L = k$ to solve the next sub-problem. When this happens we are no longer sure that our solution is optimal, and then we give the best solved bound in Column *Opt*.

4. Conclusion

In this paper, we introduced a new efficient exact procedure to obtain the globe optimization of the set-covering-based p-center problems, which have been proved better than the classical formulation (*PC*), to solve the large scale problem with the simple repeating procedure. We could maintain the globe optimality in reasonable time limit and without performing the complex algorithm of *Bsearch* and *BsearchEx*.

Moreover, upon the proposed bisecting search method, we introduce the concept of simplification to the continuous problem with the same distance matrix and increasing p value. The global optimal solution to the original problem still obtainable as the number of sub-problems is decreased and the total CPU time is shortened; simultaneously.

Table 1. Results of 40 OR-Lib instances

Instance	$N = M$	p	*Opt*	Total CPU Time in seconds			
				Daskin	Ilhan et al.	Elloumi et al.	*p-SBsearch*
Pmed1	100	5	127	5.8	2.1	0.9	0.2
Pmed2	100	10	98	2.7	0.9	0.2	0.1
Pmed3	100	10	93	2.2	0.8	0.1	0.1
Pmed4	100	20	74	2.4	0.6	0.1	0.1
Pmed5	100	33	48	0.2	0.6	0.1	0.1
Pmed6	200	5	84	14.8	6.1	1.1	0.6
Pmed7	200	10	64	9.8	2.7	0.5	0.3
Pmed8	200	20	55	10.8	1.9	0.4	0.4
Pmed9	200	40	37	3.6	1.7	0.1	0.3
Pmed10	200	67	20	3.9	1.4	0.3	0.1
Pmed11	300	5	59	17.1	9.1	2.1	0.9
Pmed12	300	10	51	20.3	8.2	1.3	1.0
Pmed13	300	30	36	9.2	4.2	0.8	0.9
Pmed14	300	60	26	9.3	3.4	0.9	0.5
Pmed15	300	100	18	4.8	2.7	1.0	0.3
Pmed16	400	5	47	35.1	13.9	1.6	1.4
Pmed17	400	10	39	39.2	13.4	2.1	2.0
Pmed18	400	40	28	16.6	19.4	1.4	1.4
Pmed19	400	80	18	6.9	4.9	0.8	0.5
Pmed20	400	133	13	8.9	4.1	1.8	0.4
Pmed21	500	5	40	87.0	42.3	5.2	2.4
Pmed22	500	10	38	38.6	130.5	11.2	6.6
Pmed23	500	50	22	211.0	35.8	3.3	2.3
Pmed24	500	100	15	9.9	7.8	4.5	1.2
Pmed25	500	167	11	6.3	7.1	2.7	0.8
Pmed26	600	5	38	93.9	121.7	14.9	3.5
Pmed27	600	10	32	87.2	73.5	8.2	4.5
Pmed28	600	60	18	24.4	18.2	2.1	3.3
Pmed29	600	120	13	23.6	10.2	5.1	1.6
Pmed30	600	200	9	8.6	10.0	5.4	1.1
Pmed31	700	5	30	191.1	108.2	8.1	4.4
Pmed32	700	10	29	1402.5	460.3	58.4	7.2
Pmed33	700	70	15	39.7	32.4	7.4	7.5
Pmed34	700	140	11	24.9	15.6	6.5	1.5
Pmed35	800	5	30	246.2	66.5	13.7	6.7
Pmed36	800	10	27	441.8	342.1	55.7	25.6
Pmed37	800	80	15	58.7	35.2	2.0	14.8
Pmed38	900	5	29	102.3	96.0	18.5	11.5
Pmed39	900	10	23	252.1	536.5	48.5	27.2
Pmed40	900	90	13	89.1	404.9	7.8	6.4
Maximum				**1402.5**	**536.5**	**58.4**	**27.2**
Average				**91.6**	**66.4**	**7.7**	**3.8**
Coefficient of Variation (%)				**249.5**	**195.4**	**182.5**	**161.6**

Table 2. Results of TSPLIB instances

Instance	$N = M$	p	Elloumi et al.					*p-SBsearch*			
			*LB**	*UB**	*Opt*	*cpu1*	*cpu2*	*5% Bnd*	*opt*	*cpu*	*cpu2*
u1060	1060	10	2273	2273	2273	27	53	2272-2386	2273	36	52
u1060	1060	20	1556	1768	1581	63	2778	1531-1590	1581	135	2329
u1060	1060	30	1205	1275	1208	50	298	1185-1210	1208	36	257
u1060	1060	40	1013	1079	1021	35	366	1005-1029	1021	26	121
u1060	1060	50	895	963	905	21	383	905-921	905	191	273
u1060	1060	60	765	807	781	21	233	761-790	781	4	437
u1060	1060	70	707	761	711	17	135	708-738	710* (708-721)	7	3608
u1060	1060	80	652	711	652	18	60	640-670	652	2	8
u1060	1060	90	604	636	608	19	38	600-609	608	2	6
u1060	1060	100	570	570	570	18	29	570-599	570	1	2
u1060	1060	110	539	539	539	18	30	538-552	539	1	1
u1060	1060	120	510	538	510	29	44	510-515	510	3	3
u1060	1060	130	495	510	500	28	44	495-510	500	1	3
u1060	1060	140	452	500	452	28	46	452-474	452	1	2
u1060	1060	150	430	447	447	34	50	447-452	447	1	1
rl1323	1323	10	3062	3329	3077	106	1380	3017-3155	3077	62	265
rl1323	1323	20	2008	2152	2016	115	480	1949-2036	2016	97	2543
rl1323	1323	30	1611	1797	1632	99	900	1587-1640	1632	193	5147
rl1323	1323	40	1334	1521	1352	76	3000	1339-1381	1365*(1339-1366)	3233	14132
rl1323	1323	50	1165	1300	1187	61	8580	1164-1197	1187*(1164-1188)	156	14571
rl1323	1323	60	1047	1194	1063	55	9120	1048-1076	1066*(1048-1067)	23	13382
rl1323	1323	70	959	1040	972	42	1740	970-1018	980*(872-981)	3603	16665
rl1323	1323	80	889	948	895	37	420	894-936	903*(805-904)	3603	18116
rl1323	1323	90	830	857	832	30	120	824-864	834*(832-835)	3	7503
rl1323	1323	100	777	803	787	26	120	763-796	788*(779-789)	145	8645
u1817	1817	10	455	467	458	611	2700	457-480	458	789	3973
u1817	1817	20	306	342	310*	660	4920	306-318	314*(306-315)	937	8499
u1817	1817	30	240	287	250*	355	16500	251-257	251*(232-252)	7718	8321
u1817	1817	40	205	234	210*	247	6420	211-221	216*(169-217)	4344	9049
u1817	1817	50	180	205	187*	242	9840	183-190	189*(145-190)	4287	15087
u1817	1817	60	163	183	163	177	1260	162-169	162	175	348
u1817	1817	70	148	152	148	166	420	143-149	148	82	106
u1817	1817	80	137	148	137	150	1140	142-148	137*(127-140)	3696	3814
u1817	1817	90	127	148	130*	161	7202	127-131	130*(127-131)	96	7296
u1817	1817	100	127	130	127	159	300	126-130	127*(126-128)	13	3670
u1817	1817	110	108	127	109	119	420	109-111	109	181	453
u1817	1817	120	108	108	108	131	120	107-109	108	12	18
u1817	1817	130	105	109	108*	121	3720	105-108	107*(99-108)	3605	7212
u1817	1817	140	102	108	105*	121	4020	102-107	105*(97-107)	3606	7212
u1817	1817	150	92	108	94*	144	5640	99-102	99*(89-102)	7256	7256

Note. "*" = opt is the best found solution for that instance. The range in brackets is the best-solved value of *p-SBsearch*. Columns *cpu1*, *cpu2*, *cpu3* and *cpu4* are recorded in seconds.

The procedure allows one to set the bound tolerance in advance to obtain a narrow solution bound to have shorter CPU time is another advantage of bisecting search method. For the real-world large scale problem, the mathematical optimal solution may be not desired. Instead, an acceptable narrow bound would be acceptable as a shorter CPU time that is achieved by setting a bound tolerance. The proposed formulation (*PC-SC2*) and procedure are efficient in the reasonable time limit and exact with good quality of the solution bounds.

References

[1] Beasley, J.E . (1990). " OR-Library: distributing test problems by electronic mail". Journal of the Operational Research Society 41, 1069-1072.

[2] Daskin, M. (1995). *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, Inc., New York.

[3] Daskin, M. (2000). "A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results". Communications of the Operations Research Society of Japan 45, 428-436.

[4] Drezner, Z. (1984). "The$p$-center problem: Heuristic and optimal algorithms". Journal of the Operational Research Society 35, 741-748.

[5] Elloumi, S., Labbe, M., Pochet, Y. (2004). " A New Formulation and Resolution Method for the p-center Problem". INFORMS Journal on Computing 16, 89-94.

[6] Handler, G. Y. (1990). "p-center Problems, in Discrete Location Theory". John Wiley Inc., New York, 305-347.

[7] Kariv, O., Hakimi, S. L. (1979). "An algorithmic approach to network location problems, Part 1: The p-centers". SIAM Journal on Applied Mathematics 37, 513-538.

[8] Masuyama, S., Ibaraki, T., Hasegawa, T. (1981). "The computational complexity of the m-center problems on the plane". Transactions IECE Japan E64, 57-64.

[9] Owen, S. H., Daskin, M. S. (1998). "Strategic facility location: A review". Europe Journal of Operational Research 111, 423-447.

[10] Reinelt, G. (1991). "TSPLIB-a travelling salesman problem library". ORSA Journal on Computing 3, 376-384.

On the Solution of Interplanetary Trajectory Design Problems by Global Optimisation Methods

Pierluigi Di Lizia,[1] Gianmarco Radice,[2] and Massimiliano Vasile[3]

[1] *Department of Aerospace Engineering, Politecnico di Milano, via La Masa 34, 20156, Milan, Italy* dilizia@aero.polimi.it

[2] *Department of Aerospace Engineering, University of Glasgow, Glasgow, G12 8QQ, UK* gradice@aero.gla.ac.uk

[3] *Department of Aerospace Engineering, Politecnico di Milano, via La Masa 34, 20156, Milan, Italy* vasile@aero.polimi.it

Abstract A study of global optimisation methods in the field of interplanetary trajectory has been performed. From the No Free Lunch Theorem, it is impossible that an algorithm outperforms all others in all the possible applications, therefore the aim of this work was two fold: to identify a suitable global optimisation algorithm that outperforms all others in a particular transfer typology; to identify a suitable global optimisation algorithm family that outperforms all others in all mission analysis transfer problems. At first a characterisation of the different transfer families, depending on propulsion system and number of planetary bodies involved was conducted. The model characterisation was performed within the search space to describe the morphological features of the objective function, and within the objective function to identify continuity and convexity. Once the optimisation problem has been fully defined, an exhaustive and systematic analysis of the resulting objective function structure has been performed in order to identify typical features which would mostly affect the global search.

Keywords: mission analysis, trajectory optimisation, global optimisation tools

1. Introduction

In the last two decades, global optimisation approaches have been extensively used towards the solution of complex interplanetary transfers. As operational costs have been increasingly reduced, space systems engineers have been facing the challenging task of maximising the payload-launch mass ratio while still achieving the primary mission goals. Methods ranging from genetic algorithms [1] to neurocontrollers [2], from shooting methods [3] to collocation methods [4] have been used with varying effectiveness. Unfortunately the efficiency, both computational and performance-wise, of these approaches are strongly linked to the type of problem that has to be solved. It would therefore be hugely beneficial if mission designers could rely on a limited number of global optimisation methods depending on the type of trajectory design, which has to be accomplished.

To achieve this ambitious goal, initially, a thorough identification and modelling of the main types of orbital transfers has to be performed. The orbital transfer typologies will be identified both on the basis of the propulsive system (impulsive or low thrust) and on the number of planetary bodies contributing to the dynamics of the system. The models identified previously will then have to be characterised, in order to hopefully identify some common features and recognize different transfer families within the same transfer typology as a function of the parameters of the problem. This will be performed through a two-fold analysis: within the search space, by means of a topological analysis aiming to identify variables which are useful in the description of the morphological structure of the objective function; within the objective function aiming to identify its structure and evaluating its continuity and convexity characteristics.

Systematic and/or probabilistic methodologies will be used: we first proceed by analysing the main characteristics of common trajectory design problems in mission analysis. In doing this we make use of two simple and basic algorithms: a random start search with SQP local optimisation and a grid search with regular sampling of the objective function. This analysis will contain the seed for the development of the appropriate solution algorithm since the complexity of the problem is intrinsically associated to the solving algorithm.

2. Optimisation Algorithms

Algorithms for global optimisation can be mainly classified in three classes: *stochastic algorithms*, which involve a suitably chosen random sample of points and subsequent manipulation of the sample to find good local minima; *guaranteed algorithms*, which are deterministic algorithms that guarantee the identification of a global optimum with a required accuracy; *metamodel algorithms* that exploit the construction of metamodels, and do not perform the global search on the real objective function, but on a metamodel of it.

Further, stochastic algorithms two main subclasses have been analysed: *Simulated Annealing (SA)*, which performs the global search based on successive update steps, where the update step length is proportional to an arbitrarily set parameter which can play the role of a temperature; *Evolutionary Algorithms (EAs)*, which globally search the solution space by simulating the self-optimising natural process of evolution. *Evolutionary Algorithms (EAs)* can be further divided in three main branches: *Genetic Algorithms (GAs)*, where a wide exploration of the search space and the exploitation of promising areas are ensured by means of the mutation, crossover and selection operators; *Evolutionary Programming (EP)*, whose classical scheme makes use of the only mutation operator and simulate the natural evolution at phenotypic level; *Evolutionary Strategies (ESs)*, which simulate the natural evolution at a phenotypic level, but also make use of recombination operators.

The set chosen embraces classical genetic algorithms including different genetic operators for performing the global search (GAOT and GATBX), genetic algorithms with sharing and migration operators (GAOT-shared and GATBX-migr respectively), evolutionary programming (Fast Evolutionary Programming, FEP), differential evolution (DE), an improved simulated annealing (Adaptive Simulated Annealing, ASA), branching methods (glbSolve and MCS), response surface based optimisation algorithms (rbfSolve) and, an innovative hybrid systematic-heuristic method combing branching techniques and evolutionary programming (EPIC).

3. 2-Impulse Transfers

As an example of a 2-impulse transfer, let us consider a direct transfer from Earth to Mars. The objective function has been taken as the overall impulsive ΔV: the sum of the magnitudes of the relative velocities at the beginning, ΔV_I, and the end, ΔV_F, of the interplanetary transfer phase. The search space is characterized by two design variables: date of departure from Earth, t_0, and Earth-Mars transfer time, tt.

A systematic analysis of the objective function structure allowed the identification of a remarkable quasi-periodicity with respect to the date of departure from Earth, which can be related to the synodic period of the Earth-Mars system.

The best identified solution is characterized by a ΔV value of 5678.904 m/s, corresponding to a transfer time of 203.541 d starting from the 7^{th} June 2003.

The performances of each global optimization tool in solving the 2-impulse direct planet-to-planet transfer are now reported. The evaluation criteria is based on the analysis of the optimal

Table 1. Summary of results for the two impulse direct planet-to-planet transfer problem.

Method	ΔV [m/s]	Function evaluations	Runtime [STU]
GAOT	5688.424 (σ = 10.352)	1011.200 (σ = 5.903)	$7.645 \cdot 10^{-3}$ ($\sigma = 2.082 \cdot 10^{-3}$)
GAOT-shared	5993.229 (σ = 246.338)	1011.200 (σ = 9.259)	$7.424 \cdot 10^{-3}$ ($\sigma = 1.706 \cdot 10^{-3}$)
GATBX	5912.194 (σ = 468.795)	1010 (σ = 0)	$5.491 \cdot 10^{-3}$ ($\sigma = 9.684 \cdot 10^{-4}$)
GATBX-migr	5750.769 (σ = 186.124)	1010 (σ = 0)	$4.669 \cdot 10^{-3}$ ($\sigma = 1.664 \cdot 10^{-3}$)
FEP	5751.066 (σ = 185.936)	1027.900 (σ = 16.045)	$9.640 \cdot 10^{-3}$ ($\sigma = 1.156 \cdot 10^{-3}$)
DE	5825.500 (σ = 183.588)	1013.600 (σ = 10.384)	$3.429 \cdot 10^{-3}$ ($\sigma = 2.601 \cdot 10^{-4}$)
ASA	5892.491 (σ = 500.108)	1001 (s = 0)	$2.994 \cdot 10^{-3}$ ($\sigma = 1.543 \cdot 10^{-4}$)
GlbSolve	5931.243	1005	$1.149 \cdot 10^{-3}$
MCS	5678.904	1010	$1.261 \cdot 10^{-2}$
EPIC	5679.100 (σ = 0.579)	1040 (σ = 11)	N/A

solution reached with a fixed number of model function evaluations. Due to the presence of not optimized codes among the tested ones, timing will not be considered as a main evaluation criterion. Table 1 shows the summary of results. It can be seen that all the algorithms reach the main basin of attraction, corresponding to the global minimum. It is interesting to observe the improvement gained by the MCS algorithm compared with the performances of the more classic globSolve tool: MCS and globSolve algorithms have been both inspired by DIRECT method for global optimization [6]; however, unlike the globSolve, MCS uses a branching method which allows for a more irregular splitting procedure. The MCS approach leads to obvious improvements in the effectiveness at identifying the basin of attraction of the best known solution in the 2-impulse direct planet-to-planet transfer problem, making the algorithm performances less dependent on the upper lower bounds, especially referring to design variables associate to objective function periodicities. It is also worth highlighting the effects of the sharing operator on the GAOT performances: by promoting the diversity of the individuals in the population, the GAOT-shared algorithm hinders the concentration of the individuals around the optimal solutions; this can lead to low accuracy at describing the optimum solutions. We can then conclude that, in the simple case of 2-impulse direct planet-to-planet transfer problem, the MCS algorithm have shown to be the best performing one.

4. Low-Thrust Transfers

As an example of a low-thrust transfer, let us consider a direct transfer from Earth to Mars. The thrust level has been supposed to be constant throughout the whole transfer and bounded in intervals corresponding to real thrusters values. The thrust direction during the transfer trajectory is however a design variable and is evaluated by means of azimuth and elevation angles defined in the local horizontal plane. The objective function is assumed to be a weighted sum of several terms: the magnitude of the spacecraft relative position with respect to Mars at the end of the integration of motion, R_F; the magnitude of the spacecraft relative velocity with respect to Mars at the end of the integration of motion, V_F; the propellant mass required by the thrusters for the interplanetary transfer only, m_{prop}. The search space is therefore characterized by sixteen design variables: date of departure from Earth t_0, Earth-Mars transfer time, tt, magnitude of Earth escape velocity, V_E, thrust level u and six parametrization values of the thrust azimuth and elevation during the transfer.

In order to further analyse the structure of the objective function, the distribution of the local minima over the whole search domain has been studied. The analysis of the normalized mean distance of the local minima and the corresponding objective function values [5] lead to the identification of a big-valley structure, mainly related to the quasi-periodicity with respect to the date of departure from Earth.

Table 2. Summary of results for the Earth-Mars low thrust transfer problem.

Method	Objective function	Function evaluations	Runtime [STU]
GAOT	140.301 (σ = 105.016)	80011.500 (σ = 6.451)	20.157 (σ = 2.013)
GAOT-shared	321.426 (σ = 79.997)	80007.600 (σ = 5.379)	38.441 (σ = 2.937)
GATBX	145.060 (σ = 52.383)	80020 (σ = 0)	39.279 (σ = 7.476)
GATBX-migr	119.016 (σ = 74.373)	80020 (σ = 0)	14.934 (σ = 2.1802)
FEP	160.067 (σ = 89.894)	80062.800 (σ = 29.001)	20.888 (σ = 2.096)
DE	87.454 (σ = 15.589)	80020.800 (σ = 11.153)	12.676 (σ = 0.091)
ASA	199.431 (σ = 37.710)	80001 (σ = 0)	12.993 (σ = 1.090)
GlbSolve	154.175	80069	31.902
MCS	330.712	80000	17.7233
RbfSolve	352.787	1000	77.754
EPIC	10.24 (σ = 11.33)	80799 (σ = 16952)	N/A

The best local minimum found, whose main features can be resumed with a transfer time of 203.541 d, a thrust level of 0.151 N, a propellant mass of 124.59 kg and an Earth escape velocity of 2739.5 m/s, corresponds to an objective function value of 6.37.

The performances of each global optimization tool in solving the low thrust Earth-Mars transfer are reported in Table 2. Results seem to indicate that GAOT and GAOT-shared tools, as well as the non randomized algorithms glbSolve, MCS and rbfSolve are not suitable for global optimisation of low-thrust direct planet-to-planet transfer problems using the mathematical models here employed. Among the remaining tools, DE, and GATBX-migr showed good performances in a Pareto optimal sense: in particular, DE and GATBX-migr resulted in similar, even though low, rate of success in identifying the basin of attraction of the global minimum. However, by considering that the rate of success is evaluated by performing local optimisation processes requiring similar further objective function evaluations and by noting that DE reaches a lower, and less fluctuating, value for the objective function, the DE tool seems to be preferable with respect to GATBX-migr.

5. Low-Energy Transfers

The possibility of designing low energy lunar space trajectories exploiting more than one gravitational attraction is now investigated. In particular, the framework of the Restricted Three-Body Problem (R3BP) is here analysed and lunar transfers are studied which take advantage of the dynamic of the corresponding libration points. The interior stable manifold associated to the libration point L1 in the Earth-Moon system, W_{L1}^{s} , is propagated backward for an interval of time t_W. Corresponding to W_{L1}^{S}, the exterior unstable manifold, W_{L1}^{U} , can be evaluated. The manifolds W_{L1}^{S} and W_{L1}^{U} constitute in fact a transit orbit between the forbidden region through the corresponding thin transit region. However, the manifold W_{L1}^{S} does not reach low distances from Earth. To solve this problem, starting from a circular orbit around the Earth, an arc resulting from the solution of a Lambert's three-body problem is used for targeting a point on the stable manifold. It is worth noting that such an approach leads to a final unstable orbit around the Moon with mean altitude equal to 21600 km.

As a consequence of the previously described formulation, a first impulsive manoeuvre, ΔV_1, is used to put the spacecraft in the Lambert's three-body arc from the initial circular orbit around the Earth. A second impulsive manoeuvre, ΔV_2, is performed to inject the spacecraft on the capture trajectory W_{L1}^{S}. Hence, the sum of the two impulsive manoeuvres, ΔV, is chosen to be the objective function for the optimisation processes. As a consequence the search space is characterized by the following design variables: the angle identifying the starting point over the initial circular orbit, θ; the time of the backward propagation of the stable manifold, t_W, and the transfer time corresponding to the Lambert's three-body arc, t_L.

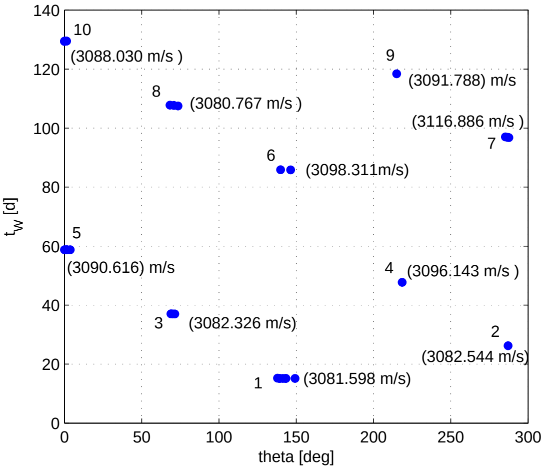


Figure 1. Families of Lunar transfers comparable with the best identified one (subgroup 8).

Table 3. Summary of results for the low-energy transfer problem.

Method	ΔV	Function evaluations	Runtime [STU]
GAOT	3160.307 (σ = 74.987)	5012.200 (σ = 4.638)	24.788 (σ = 2.093)
GAOT-shared	3321.952 (σ = 101.235)	5010.900 (σ = 6.657)	20.197 (σ = 2.488)
GATBX	3158.640 (σ = 63.572)	5010 (σ = 0)	38.556 (σ = 4.180)
GATBX-migr	3218.439 (σ = 127.349)	5010 (σ = 0)	35.895 (σ = 5.012)
FEP	3134.626 (σ = 60.928)	5017.200 (σ = 15.640)	31.578 (σ = 5.232)
DE	3233.064 (σ = 94.020)	5019.600 (σ = 10.276)	16.106 (σ = 0.428)
ASA	3194.447 (σ = 109.524)	4783.600 (s = 58.971)	37.544 (σ = 5.110)
GlbSolve	3343.104	5025	21.640
MCS	3148.107	5010	32.575
RbfSolve	3579.249	474	6.128

The best solution identified is characterized by an overall transfer time of 108.943 d and corresponds to an objective function value of 3080.767 m/s.
A careful analysis of the distribution of the local minima over the search space lead to the identification of several sets of local optimal solutions, comparable with the best identified one in terms of objective function values, which are characterized by similar values of the time spent on the three-body Lambert's arc, t_L. The presence of such subgroups can be related to the presence of big valley structures deriving from the periodicity of the objective function on t_W. We can state that such subgroups describe a set of different families of Lunar transfers where the term "family" is referred to solutions lying on different niches on the search space, as defined in [7]. In particular ten local minima groupings can be isolated over the analysed search space, as shown in Figure 1.

The performances of each global optimization tool in solving the low-energy transfer problem are reported in Table 3. The results allow us to infer the following. The stochastic algorithms GAOT-shared, GATBX-migr and DE and the non randomized codes glbSolve and rbfSolve, cannot be considered as suitable for solving the previously identified problem, due to their inability in identifying basin of attractions corresponding to either the best known solution or the comparable ones. GAOT, GATBX and ASA identify the basin of attraction of good solutions but present relatively large standard deviations, implying that not always the attraction basins are identified successfully. As a consequence MCS and FEP turn out to be the best performing tools for the problem of lunar transfer using libration points.

6. Summary

The aim of this study was an investigation of the effectiveness of some global optimisation techniques at solving practical problems related to space trajectory design. Following a complete and comprehensive objective function structure analysis a set of global optimisation tools has been selected for testing purposes. By considering the objective function value reached at the end of the optimisation process, the number of objective function evaluations performed required and the effectiveness at identifying the basin of attraction of the best known solution as well as of good solutions comparable to the best known one, results of the test phase can be resumed as follows.

Two impulse direct planet-to-planet transfer problem: due to its deterministic features, the success at reaching the best known solution and the corresponding relatively low number of required objective function evaluations, Multilevel Coordinate Search (MCS) turned out to outperform all the remaining algorithms, thus resulting as the best performing one.

Low thrust direct planet-to-planet transfer problem: due to the highly complex nature of the search space, low rate of success characterized all the tested algorithms at identifying the basins of attraction of both the best known solution and solutions comparable to it. In such an environment in particular, DE and GATBX-migr resulted in similar, even though low, rate of success in identifying the basin of attraction of the global minimum. However, by noting that DE reaches a lower, and less fluctuating, value for the objective function, the DE tool seems to be preferable with respect to GATBX-migr.

Low energy Lunar transfer problem: the structure of the objective function and the search space results in comparable performances for the majority of global optimisation algorithms used. However, by taking into account the minimum value of the objective function reached, and the standard deviation, we can identify MCS and FEP as the best performing tools for the problem of lunar transfer using libration points.

It is worth noting that limitations affects the achieved results. First of all, each mission analysis class has been investigated by selecting a particular transfer problem and by facing it with proper, but anyway particular, mathematical models. Further analyses should be performed, including additional transfer problems, alternative mathematical models and search space definitions. Secondly, it is widely known that optimisation algorithms can be suitably tuned to enhance their performances. However, as already occurred in remarkable existing comparative studies [7], due to the comparative purposes of this work, the large scale of comparisons performed, the available devices and the high time required by some optimisation case, it was impossible to do such tuning.

References

[1] Hughes, G., and McInnes, C.R.: Solar Sail Hybrid Trajectory Optimisation. Advances in the Astronautical Sciences, Vol. 109, pp. 2369-2380, 2001.

[2] Dachwald, B.: Optimisation of Solar Sail Interplanetary Trajectories Using Evolutionary Neurocontrol. Journal of Guidance Control and Dynamics, Vol. 27, No. 1, pp. 66-72, 2004

[3] Wirthman, D.J., Park, S.Y., Vadali, S.R.: Trajectory Optimisation Using Parallel Shooting Method on Parallel Computer. Journal of Guidance, Control and Dynamics, Vol. 18, No. 2, pp. 377-379, 1995.

[4] Betts, J.T., Orb, S.O.: Optimal Low Thrust Trajectories to the Moon. SIAM Journal of Applied Dynamical Systems, Vol. 2, No. 2, pp. 144-170, 2003.

[5] Reeves, C.R., Yamada, T.: Genetic Algorithms, Path Re-linking and the Flowshop Sequencing Problem. Evolutionary Computation, Vol. 6, pp. 45-60, 1998

[6] Jones, D.R., Perttunen, C.D. and Stuckman, B.E.: Lipschitzian Optimization without the Lipschitz Constant. Journal of Optimization Theory and Applications, Vol. 79, pp. 157-181, 1993.

[7] Pintér, J.D.: Global Optimization: software, test problems, and applications. Handbook of Global Optimization (Horst, R. and Pardalos, P.M., eds.), Kluwer Academic Publishers, Dordrecht Boston, London, 1995.

Parametrical approach for studying and solving bilinear programming problem

Dmitrii Lozovanu

Institute of Mathematics and Computer Science of Moldovan Academy of Sciences, Academy str., 5, Chisinau, MD–2028
lozovanu@math.md

Abstract The linear parametrical programming approach for studying and solving a bilinear programming problem (BPP) is proposed. On the basis of the duality theory BPP is transformed into a problem of determining compatibility of a system of linear inequalities with a right part that depends on parameters, admissible values of which are determined by an another system of linear inequalities. Some properties of this auxiliary problem are obtained and algorithms for solving BPP are proposed.

Keywords: Bilinear Programming, Linear Parametrical Systems, Duality Principle for Parametrical Systems

1. Introduction and preliminary results

We consider the following bilinear programming problem (BPP) [1,2]:
to minimize the object function

$$z = xCy + d^1x + d^2y \tag{1}$$

on subject

$$A^1x \le b^1,\ x \ge 0; \tag{2}$$

$$A^2y \le b^2,\ y \ge 0, \tag{3}$$

where C, A^1, A^2 are matrices of size $n\times m, m\times n, k\times m$, respectively, and $d^1, x \in R^n;\ d^2, y, b^1 \in R^m;\ b^2 \in R^k$. This problem can be solved by varying the parameter $h \in [-2^L, 2^L]$ in the problem of determining compatibility of the system

$$\left\{\begin{array}{l} A^1x \le b^1,\ x \ge 0; \\ xCy + d^1x + d^2y \le h; \\ A^2y \le b^2,\ y \ge 0, \end{array}\right. \tag{4}$$

where L is the size of the problem (1)-(3).

Furthermore we regard the compatibility problem with respect to x of the system

$$\left\{\begin{array}{l} A^1x \le b^1,\ x \ge 0; \\ xCy + d^1x \le h - d^2y. \end{array}\right. \tag{5}$$

for every y satisfying (3).

The following Theorems are proved in [3] on the basis of the linear duality theory applied to (5).

Theorem 1. *The system (4) has no solutions if and only if the following system of linear inequalities*

$$\begin{cases} -{A^1}^T u \le Cy + d^1; \\ b^1 u < d^2 y - h; \\ u \ge 0 \end{cases}$$

is compatible with respect to u for every y satisfying (3).

Theorem 2. *The minimal value of the object function in the problem (1)-(3) is equal to the maximal value h^* of the parameter h in the system*

$$\begin{cases} -{A^1}^T u \le Cy + d^1; \\ b^1 u \le d^2 y - h; \\ u \ge 0 \end{cases} \tag{6}$$

for which it is compatible with respect to u for every y satisfying (3). An arbitrary solution y^* of the system (3) such that the system

$$\begin{cases} -{A^1}^T u \le Cy^* + d^1; \\ b^1 u < d^2 y^* - h; \\ u \ge 0 \end{cases}$$

has no solution with respect to u corresponds to a solution (x^*, y^*) of the problem (1)-(3).

In this paper we study the problem of determining the compatibility of the system (6) for every y satisfying (3). On the basis of the obtained results we propose algorithms for solving BPP.

Note that the considered problems are NP-hard, but the mentioned above approach allows to argue suitable algorithms for some classes of these problems.

2. Main properties of systems of linear inequalities with a right part that depends on parameters

In order to study BPP we shall use the following results.

2.1 Duality principles for systems of linear inequalities

Let the following system of linear inequalities be given

$$\begin{cases} \sum\limits_{j=1}^{n} a_{ij} u_j \le \sum\limits_{s=1}^{k} b_{is} y_s + b_{i0}, \ i = \overline{1,m}; \\ u_j \ge 0, \ j = \overline{1,p} \ (p \le n) \end{cases} \tag{7}$$

with the right part depending on parameters $y_1, y_2, \dots, y_k$. We consider the problem of determining compatibility of the system (7) for every $y_1, y_2, \dots, y_k$ satisfying the following system

$$\begin{cases} \sum\limits_{s=1}^{k} g_{is} y_s + g_{i0} \le 0, \ i = \overline{1,r}; \\ y_s \ge 0, \ s = \overline{1,q} \ (q \le k). \end{cases} \tag{8}$$

The following Theorem holds [4].

Theorem 3. *The system (7) is compatible with respect to $u_1, u_2, \ldots, u_n$ for every $y_1, y_2, \ldots, y_k$ satisfying (8) if and only if the following system*

$$\begin{cases} -\sum_{i=1}^{r} g_{is} v_i \leq \sum_{i=1}^{m} b_{is} z_i, \ s = \overline{0, q}; \\ -\sum_{i=1}^{r} g_{is} v_i = \sum_{i=1}^{m} b_{is} z_i, \ s = \overline{q+1, k}; \\ v_i \geq 0, \ i = \overline{1, r} \end{cases}$$

is compatible with respect to $v_1, v_2, \ldots, v_r$ for every $z_1, z_2, \ldots, z_m$ satisfying the following system

$$\begin{cases} -\sum_{i=1}^{m} a_{ij} z_i \leq 0, \ j = \overline{1, p}; \\ -\sum_{i=1}^{m} a_{ij} z_i = 0, \ j = \overline{p+1, n}; \\ z_i \geq 0, \ i = \overline{1, m}. \end{cases}$$

2.2 Two special cases of the parametrical problem

Note that if $r = 0$ and $q = k$ in the system (8) then we obtain the problem of determining compatibility of the system (7) for every nonnegative values of parameters $y_1, y_2, \ldots, y_k$. It is easy to observe that in this case the system (7) is compatible for every nonnegative values of parameters $y_1, y_2, \ldots, y_k$ if and only if each of the following systems

$$\begin{cases} \sum_{j=1}^{n} a_{ij} u_j \leq b_{is}, \ s = \overline{0, k}; \\ u_j \geq 0, \ j = \overline{1, p} \end{cases}$$

is compatible.

An another special case of the problem is the one when $n = 0$. This case can be reduced to the previous one using the duality problem for it.

In such a way, our problem can be solved in polynomial time for the mentioned above cases.

2.3 General approach for determining the compatibility property for parametrical systems

It is easy to observe that the compatibility property of the system (7) for all admissible values of parameters $y_1, y_2, \ldots, y_k$ satisfying (8) can be verified by checking compatibility of the system (7) for every basic solution of the system (8). This fact follows from the geometrical interpretation of the problem. The set $Y \subseteq R^k$ of vectors $y = (y_1, y_2, \ldots, y_k)$, for which the system (7) is compatible, corresponds to an orthogonal projection on R^k of the set $UY \subseteq R^{n+k}$ of solutions of the system (7) with respect to variables $u_1, u_2, \ldots, u_n$, $y_1, y_2, \ldots, y_k$.

Another general approach which can be argued on the basis of the mentioned above geometrical interpretation is the following.

We find the system of linear inequalities

$$\sum_{j=1}^{r} b'_{ij} \, y_j + b'_{i0} \leq 0, \ i = \overline{1, m'}, \tag{9}$$

which determines the orthogonal projection Y of the set $UY \subseteq R^{n+k}$ on R^k; then we solve the problem from section 2.2. The system (9) can be found by using method of elimination of variables $u_1, u_2, \ldots, u_n$ from the system (7). The method of elimination of variables from the system of linear inequalities can be found in [5]. Note that in the final system (9) the number of inequalities m' can be too big. Therefore such approach for solving our problem can be used only for a small class of problems.

3. An algorithm for the establishment of the compatibility of a parametrical problem

We propose an algorithm for the establishment of the compatibility of the system (6) for every y satisfying (3). This algorithm works in the case when the sets of solutions of the considered systems are bounded. The case of the problem with unbounded sets of solutions can be easily reduced to the bounded one.

Step 1. Choose an arbitrary basic solution y^0 of the system (3). This solution corresponds to a solution of the system of linear equations

$$\sum_{j=1}^{m} b_{i_s j} y_j + b_{i_s 0} = 0, \ s = \overline{1, m}. \tag{10}$$

The matrix $\overline{B} = (b_{i_s j})$ of this system represents a submatrix of the matrix

$$B' = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1m} \\ b_{21} & b_{22} & \ldots & b_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ b_{k1} & b_{k2} & \ldots & b_{km} \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{pmatrix}$$

and the vector $\overline{b'}^T = (b_{i_1}, b_{i_2}, \ldots, b_{i_k})$ is a "subvector" of $b'^T = (b_1, b_2, \ldots, b_k, 0, 0, \ldots, 0)$. The system of inequalities

$$\sum_{j=1}^{m} b_{i_s j} b_j + b_{i_s 0} \leq 0, \ s = \overline{1, m},$$

which corresponds to the system (10), determines in R^m a cone originated in y^0 with the following generating rays

$$y^s = y^0 + \overline{b}^s t, \ s = \overline{1, m}, \ t \geq 0.$$

Here $\overline{b}^1, \overline{b}^2, \ldots, \overline{b}^m$ represent directing vectors of respective rays originating in y^0. These directing vectors correspond to columns of the matrix $\overline{B}^{-1}$.

Step 2. For each $s = \overline{1, m}$, solve the following problem:

maximize t

on subject

$$\begin{cases} A^2 y \leq b^2; \\ y \geq 0; \\ y = y^0 + \overline{b}^s t, \ t \geq 0 \end{cases}$$

and find m points $\overline{y}^1, \overline{y}^2, \dots, \overline{y}^m$, which correspond to m basic solutions of the system (3), i.e. $\overline{y}^1, \overline{y}^2, \dots, \overline{y}^m$ represent neighboring basic solutions for y^0. If the system (6) is compatible with respect to u for each $y = \overline{y}^1, y = \overline{y}^2, \dots, y = \overline{y}^m$, then go to step 3; otherwise the system (3) is not compatible for every y satisfying (3) and STOP.

Step 3. For each $s = \overline{1, m}$, solve the following problem:

maximize t

on subject

$$\begin{cases} -A^{1^T} u \le Cy + d^1; \\ b^1 u \le d^2 y - h; \\ u \ge 0 \\ y = y^0 + \overline{b}^s t, \ t \ge 0 \end{cases}$$

and find m solutions $t'_1, t'_2, \dots, t'_m$. Then fix m points $y^s = y^0 + \overline{b}^s t'_s, \ s = \overline{1, m}$.

Step 4. Find the hyperplane

$$\sum_{j=1}^{m} a'_j y_j + a'_0 = 0,$$

which passes through the points $y^1, y^2, \dots, y^m$. Consider that the basic solution $y^0 = (y_1^0, y_2^0, \dots, y_m^0)$ satisfies the following condition

$$\sum_{j=1}^{m} a'_j y_j^0 + a'_0 \le 0.$$

Then add to the system (3) the inequality

$$-\sum_{j=1}^{m} a'_j y_j - a'_0 \le 0.$$

If after that the obtained system is not compatible, then conclude that the system (6) is compatible for every y satisfying the initial system (3) and STOP; otherwise change the initial system with the obtained one and go to step 1.

Note that in [3] it is proposed an algorithm for the establishment of the compatibility of the parametrical problem in the case when the system (3) has the following form

$$\begin{cases} y_1 \qquad\qquad \le b_1^2; \\ \qquad y_2 \qquad \le b_2^2; \\ \dots\dots\dots\dots\dots\dots \\ \qquad\qquad y_m \le b_m^2; \\ y_1 \ge 0, y_2 \ge 0, \dots, y_m \ge 0 \end{cases}$$

i.e. the set of solutions of the system (3) is a m-dimensional cube. Even in this case the compatibility problem remains NP-hard, but the approach from [3], based on Theorems 1,2, allows to propose suitable algorithms.

In [3] it is shown that the proposed approach can be used for studying and solving the following concave programming problem:

$$\textit{minimize } z = \sum_{i=1}^{q} \min\{c^{il}x + c_0^{il},\ l = \overline{1, r_i}\}$$

on subject

$$\begin{cases} Ax \leq b; \\ x \geq 0, \end{cases}$$

where $c^{il} \in R^n$, $c_0^{il} \in R^1$, A is a $m \times n$-matrix, $b \in R^n$. This problem can be transformed into BPP:

$$\textit{minimize } z' = \sum_{i=1}^{n} \sum_{l=1}^{r_i} (c^{il}x + c_0^{il}) y_{il}$$

on subject

$$\begin{cases} Ax \leq b,\ x \geq 0; \\ \sum_{l=1}^{r_i} y_{il} = 1,\ i = \overline{1, q}; \\ y_{il} \geq 0,\ l = \overline{1, r_i},\ i = \overline{1, q}. \end{cases}$$

The proposed approach related to this problem is described in [3].

4. Summary

The approach for studying and solving the bilinear programming problem based on linear parametrical programming and duality principle for linear inequalities with a right part that depends on parameters is proposed. This approach allows to study the nonlinear optimization problem by using linear programming methods. Algorithms and general schemes for solving different classes of bilinear programming problems are derived.

References

[1] Altman, M. (1968). *Bilinear Programming*, Bul., Acad. Polan. Sci., Ser. Sci. Math. Astron. et Phis., 16(9), 741–746.

[2] Pardalos, P., Rosen, J. (1986). *Methods for Global Concave Minimization. A bibliographical Survey*, SIAM Review, 28(3), 367–379.

[3] Lozovanu, D. (1991). *Extremal-Combinatorial Problems and Algorithms for their Solving*, Stiinta, Chisinau (in Russian).

[4] Lozovanu, D. (1987). *Duality Principle for Systems of Linear Inequalities with a Right Part that Depends on Parameters*, Izv. AN SSSR, ser. Techn. Cyb., 6, 3–13 (in Russian).

[5] Chernicov, S.N. (1968). *Linear Inequalities*, Moscow, Nauka.

Proceedings of GO 2005, pp. 171 – 176.

An Interval Branch-and-Bound Algorithm based on the General Quadratic Form

Frédéric Messine* and Ahmed Touhami

ENSEEIHT-IRIT, 2 rue C Camichel, 31071 Toulouse Cedex, France {Frederic.Messine, Ahmed.Touhami}@n7.fr

Abstract In this paper, an extension of affine arithmetic introduced by Stolfi et al. [1–5] is proposed. It is based on a general quadratic form which leads to most efficient computations of errors due to non-affine operations. This work recalls the main results published in [10] and emphasizes the interest of such a technique for solving unconstrained global optimization problems of multivariate polynomial functions.

1. Introduction

We focus in this paper on unconstrained bounded minimization problems of polynomial functions, written as follows:

$$\min_{\mathrm{x} \in \mathrm{X} \subseteq \mathbb{R}^{\mathrm{n}}} f(x), \tag{1}$$

where X is a hypercube of $\mathbb{R}^n$ and f is a polynomial function.

The Branch-and-Bound algorithms can be based on interval arithmetic for the computations of the bounds of a function over a box. The outwardly rounded interval arithmetic was then introduced by Moore to deal with numerical errors, [11]. Thus, its utilization inside Branch-and-Bound algorithms makes these methods rigorous [7]; we speak about rigorous global optimization methods when no numerical error can render wrong the computations of the bounds. The principle of a Branch-and-Bound algorithm is to bisect the initial domain where the function is sought for into smaller and smaller boxes, and then to eliminate the boxes where the global optimum cannot occur; i.e. by proving that a current solution is lower (resp. upper for a maximization problem) than a lower (resp. upper) bound of the function over this box. Therefore in such a box, there does not exist a point such that its value is lower (resp. upper) than the current solution already found.

An inclusion function is an interval function such that the interval result encloses the range of the associated real function over the studied box; i.e. if $F(X)$ denotes the inclusion function of a function f over a box X, one has by definition that $[\min_{x\in X} f(x), \max_{x\in X} f(x)] \subseteq F(X)$. The extension of an expression of a function into interval (by replacing all the occurrences of the variables by the corresponding intervals, and all the operations by interval operations) defines an inclusion function which is called the natural extension inclusion function; it is denoted $\mathbf{NE}(X)$. Nevertheless, the direct use of $\mathbf{NE}$ is generally inefficient and then, the computed bounds are not sufficiently accurate to solve both rapidly and with a high numerical precision certain optimization problems of type (1). Therefore, for the past several years, a lot of new techniques which permit to improve the computations of the bounds have been studied. Generally, these techniques are based on the combinations of first or second order

*Work of the first author was also supported by the ENSEEIHT-LEEI-EM^3, 2 rue C Camichel, 31071 Toulouse Cedex

Taylor expansions and on interval computations of an enclosure of the gradient or the Hessian matrix [6, 7, 9]; one uses here an centered inclusion function based on a Taylor expansion at the first order, it is denoted $\mathbf{T_1}(X)$ in the following.

Affine arithmetic was proposed and developed recently by Stolfi et al. [1–5], although a similar tool, the generalized interval arithmetic, have been developed in 1975 by Hansen [6]. Its principle is to keep some affine informations during the computations of bounds. Like generalized interval arithmetic [6], affine arithmetic was developed to take into account the problems of the dependencies of the variables generated by interval computations. This tool permits to limit some negative effects due to interval arithmetic: The links between different occurrences of a same variable. The interest of the bounds computed by using this affine arithmetic was shown in [8]. Furthermore in [8], two new affine forms and a first quadratic form were introduced and have proved their real efficiency solving optimization problems such as defined in (1). The purpose in this paper is to consider the complete quadratic form such as defined in [10] and to show its efficiency for solving unconstrained global optimization problems for multivariate polynomial functions. Affine arithmetic such as defined in [4] is reliable. In this article, one uses a different way to render reliable affine arithmetic by converting all the floating point coefficients into interval ones and by introducing rounded interval computations. This reliable inclusion function is denoted by $\mathbf{AF}(X)$ in the following.

The general quadratic form aims to conserve some affine informations and also some quadratic informations (about the error due to non-affine operations) during the computations. Therefore, these added informations permit to improve the quality of the so-computed bounds (in spite of an expansion of the complexity of such an algorithm). The gain of this technique is validated on some global optimization polynomial problems.

A complete work on the general quadratic form, presenting some properties, are detailed in [10]. Here, one recalls the principle of this technique to compute bounds and we emphasizes the interest of this tool in global optimization.

2. Reliable General Quadratic Form

A general quadratic form is represented by $\widehat{\widehat{x}}$:

$$\left\{\begin{array}{ll} \widehat{\widehat{x}} & = \epsilon^T A\epsilon + b^T\epsilon + c + e^+\epsilon_{n+1} + e^-\epsilon_{n+2} + e\epsilon_{n+3}, \\ & = \displaystyle\sum_{i,j=1}^{n} a_{ij}\epsilon_i\epsilon_j + \sum_{i=1}^{n} b_i\epsilon_i + c + e^+\epsilon_{n+1} + e^-\epsilon_{n+2} + e\epsilon_{n+3}, \end{array}\right. \quad (2)$$

where $A \in \mathrm{I\!R}^{\mathrm{n}\times\mathrm{n}}$, $b \in \mathrm{I\!R}^{\mathrm{n}}$, $c \in \mathrm{I\!R}$, $(e^+, e^-, e) \in (\mathrm{I\!R}^+)^3$ and $\epsilon_i \in [-1, 1]$, $\forall\, i \in \{1, \ldots, n\}$, $\epsilon_{n+1} \in [0, 1]$, $\epsilon_{n+2} \in [-1, 0]$ and $\epsilon_{n+3} \in [-1, 1]$. The symbolic variables ϵ_{n+1}, ϵ_{n+2} and ϵ_{n+3} represent the noise for the errors generated by performing non-affine computations.

The conversions between interval and a general quadratic form are performed as follows:

- **Interval $\longrightarrow$ General Quadratic Form:**

$$\begin{array}{l} X = [x^L, x^U] \\ \longrightarrow \widehat{\widehat{x}} = \epsilon^T A\epsilon + b^T\epsilon + c + e^+\epsilon_{n+1} + e^-\epsilon_{n+2} + e\epsilon_{n+3} \end{array}$$

$$\text{with}\left\{\begin{array}{lcl} A & = & 0\,, \\ b & = & \left(0,\cdots,0,\dfrac{x^L-x^U}{2},0,\cdots,0\right), \\ c & = & \dfrac{x^L+x^U}{2}, \\ e^+ & = & 0, \\ e^- & = & 0, \\ e & = & 0. \end{array}\right.$$

The rank i where $b_i = \dfrac{x^L - x^U}{2}$ is generally determined by the variable x. When a vector is considered $x \in \mathrm{I\!R}^{\mathrm{n}}$, then the rank i corresponds to x_i.

- **General Quadratic Form $\longrightarrow$ Interval:**

$$\begin{array}{rcl} \widehat{\widehat{x}} & = & \epsilon^T A \epsilon + b^T \epsilon + c + e^+ \epsilon_{n+1} + e^- \epsilon_{n+2} + e \epsilon_{n+3} \\ & = & \displaystyle\sum_{i,j=1}^{n} a_{ij}\epsilon_i\epsilon_j + \sum_{i=1}^{n} b_i\epsilon_i + c + e^+\epsilon_{n+1} + e^-\epsilon_{n+2} + e\epsilon_{n+3} \\ & & \\ \longrightarrow X & = & \displaystyle\sum_{\substack{i,j=1\\ i\neq j}}^{n} [-|a_{ij}|,|a_{ij}|] + \sum_{i=1}^{n}[-|b_i|,|b_i|] + \sum_{i=1}^{n} a_{ii}[0,1] + [|c|,|c|] + \\ & & [0,e^+] + [-e^-,0] + [-e,e]. \end{array}$$

Other conversions, such as for example between general quadratic forms and affine forms, are possible, refer to [10].

The operations between general quadratic forms are performed as follows:
let $\widehat{\widehat{x}}$ and $\widehat{\widehat{y}}$ be two general quadratic forms and a a real number, with

$$\begin{array}{rcl} \widehat{\widehat{x}} & = & \epsilon^T A_x \epsilon + b_x^T\,\epsilon + c_x + e_x^+\epsilon_{n+1} + e_x^-\epsilon_{n+2} + e_x\epsilon_{n+3}, \\ \widehat{\widehat{y}} & = & \epsilon^T A_y\,\epsilon + b_y^T\,\epsilon + c_y + e_y^+\epsilon_{n+1} + e_y^-\epsilon_{n+2} + e_y\epsilon_{n+3}. \end{array}$$

One obtains:

$$\begin{array}{rcl} -\widehat{\widehat{x}} & = & -\epsilon^T A_x\epsilon - b_x^T\,\epsilon - c_x + e_x^-\epsilon_{n+1} + e_x^+\epsilon_{n+2} + e_x\epsilon_{n+3}, \\ \widehat{\widehat{x}} + \widehat{\widehat{y}} & = & \epsilon^T(A_x + A_y)\epsilon + (b_x + b_y)^T\epsilon + (c_x + c_y) + (e_x^+ + e_y^+)\epsilon_{n+1} + (e_x^- + e_y^-)\epsilon_{n+2} + (e_x + e_y)\epsilon_{n+3}, \\ \widehat{\widehat{x}} - \widehat{\widehat{y}} & = & \epsilon^T(A_x - A_y)\epsilon + (b_x - b_y)^T\epsilon + (c_x - c_y) + (e_x^+ + e_y^-)\epsilon_{n+1} + (e_x^- + e_y^+)\epsilon_{n+2} - (e_x + e_y)\epsilon_{n+3}, \\ \widehat{\widehat{x}} \pm a & = & \epsilon^T A_x\epsilon + b_x^T\epsilon + (c_x \pm a) + e_x^+\epsilon_{n+1} + e_x^-\epsilon_{n+2} + e_x\epsilon_{n+3}, \end{array}$$

$$a \times \widehat{\widehat{x}} = \left\{\begin{array}{l} \epsilon^T(a\times A_x)\epsilon + (a\times b_x)^T\epsilon + a\times c_x + a\times e_x^+\epsilon_{n+1} + a\times e_x^-\epsilon_{n+2} + a\times e_x\epsilon_{n+3},\ \text{if } a>0, \\ \epsilon^T(a\times A_x)\epsilon + (a\times b_x)^T\epsilon + a\times c_x + |a|\times e_x^-\epsilon_{n+1} + |a|\times e_x^+\epsilon_{n+2} + |a|\times e_x\epsilon_{n+3},\ \text{if } a<0. \end{array}\right.$$

For the multiplication (a non-affine operation), the following computations are performed:

$$\widehat{\widehat{x}} \times \widehat{\widehat{y}} = \epsilon^T A_\times \epsilon + b_\times^T\epsilon + c_\times + e_\times^+\epsilon_{n+1} + e_\times^-\epsilon_{n+2} + e_\times\epsilon_{n+3}, \tag{3}$$

$$\text{with}\left\{\begin{array}{lcl} A_\times & = & c_y A_x + c_x A_y + b_x b_y^T, \\ b_\times & = & c_y\, b_x + c_x\, b_y, \\ c_\times & = & c_x\, c_y. \end{array}\right.$$

The computations of the errors e^+, e^-, e are not so obvious, but are quiet difficult and attention must be paid on it, see [10].

Therefore, this technique allows to construct a new inclusion function based on this general quadratic form; i.e. one converts all interval vector with dimension n into n general quadratic forms which correspond to each occurrence of each variable, the computations are performed using operations between general quadratic form and then, the result is converted into an interval. This resulting interval encloses rigorously the range of the function over the considered box (interval vector). This technique defines a new automatic tool for the computations of bounds for a function over a box. One denotes by $\mathbf{GQF}$ an inclusion function constructed such a way (using general quadratic forms).

Example 1. *The interest of this general quadratic form can be illustrated on this simple example* $f(x_1, x_2) = x_1^2 x_2 - x_1 x_2^2$ *over* $[-1, 3]^2$. *We have:*

$$\begin{aligned} x_1^2 x_2 - x_1 x_2^2 &= (1-2\epsilon_1)^2(1-2\epsilon_2) - (1-2\epsilon_1)(1-2\epsilon_2)^2 \\ &= (1-4\epsilon_1+4\epsilon_1^2)(1-2\epsilon_2) - (1-4\epsilon_2+4\epsilon_2^2)(1-2\epsilon_1) \\ &= \big(1-4\epsilon_1+4\epsilon_1^2-2\epsilon_2+8\epsilon_1\epsilon_2-8\epsilon_1^2\epsilon_2\big) + \big(-1+4\epsilon_2-4\epsilon_2^2+2\epsilon_1-8\epsilon_1\epsilon_2+8\epsilon_1\epsilon_2^2\big) \\ &= -2\epsilon_1+2\epsilon_2+4\epsilon_1^2-4\epsilon_2^2-8\epsilon_1^2\epsilon_2+8\epsilon_1\epsilon_2^2 \end{aligned}$$

- $\mathbf{AF}(X) = [-2\epsilon_1 + 2\epsilon_2 + 40\epsilon_3] = [-44, 44]$, *with* $\epsilon_{n+1} = \epsilon_3 \in [-1, 1]$.
- $\mathbf{GQF}(X) = [-2\epsilon_1 + 2\epsilon_2 + 4\epsilon_1^2 - 4\epsilon_2^2 + 16\epsilon_5] = [-24, 24]$, *with* $\epsilon_{n+3} = \epsilon_5 \in [-1, 1]$.
- $\mathbf{NE}(X) = X_1^2 X_2 - X_1 X_2^2 = [-1, 3]^2[-1, 3] - [-1, 3]^2[-1, 3] = [-9, 27] - [-9, 27] = [-36, 36]$.

In that case, $\mathbf{NE}$ *is better than* $\mathbf{AF}$ *but* $\mathbf{GQF}$ *is the most efficient.*

In [10], some theoretical properties about the efficiency of the bounds of such a technique are reported. Furthermore, the extension into reliable forms are defined by replacing all the floating point coefficients by intervals and by replacing all the operations by rounded interval operations. Thus, the bounds computed by introducing interval components inside the general quadratic form become reliable; no numerical error can occur performing these computations. The bounds are also guaranteed.

3. Application to Rigorous Global Optimization

The main global optimization algorithm is based on a classical Branch-and-Bound technique due to Ichida-Fujii [12]. To show the efficiency of the use of general quadratic forms, the computations of the bounds are replaced inside this algorithm. One can speak here about rigorous global optimization because the bounds are numerically guaranteed; all the computed bounds are reliable using the four techniques presented above $\mathbf{NE}, \mathbf{T_1}, \mathbf{AF}$ and $\mathbf{GQF}$. In order to show the efficiency of such an inclusion function based on the general quadratic form some multivariate polynomial problems are taken into account, see Table 1. All these problems are solved by a basic interval Branch-and-Bound algorithm due to Ichida-Fujii, see [12]. This algorithm is modified in order to determine the global minimum f^* with a maximal accuracy (by solving the problem with an accuracy divided by 10), see [10]. In fact, the algorithm stops when the new precision (divided by 10) of the global solution cannot be improved after a consequent computational effort (here 100000 iterations).

In Table 1, we summarize for each polynomial problem, the initial domain of research and the global minimum f^*. Generally, these functions came from the literature [8,9,12]; f_2 is the well known Golstein Price function and f_5 a function due to Ratschek.

All these numerical tests have been performed on a HP-UX, 9000/800, 4 GB memory, quadriprocessor 64-bit processor, computer from the `Laboratoire d'Electrotechnique et d'Electronique Industrielle du CNRS/UMR 5828 ENSEEIHT-INPT Toulouse`. The codes have been developed in Fortran 90. The algorithm uses iteratively the following inclusion functions: natural extension into interval $\mathbf{NE}$, a technique based on Taylor expansions at the

Functions	Initial Domain of Research	Global minimum
$f_1(x) = x_1^3x_2 + x_2^2x_3x_4^2 - 2x_5^2x_1 + 3x_2x_4^2x_5$	$X = [-10, 10]^5$	$f_1^* = -1042000$
$f_2(x) = [1 + (x_1 + x_2 + 1)^2$	$X = [-2, 2]^2$	$f_2^* = 3$
$(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$		
$\times[30 + (2x_1 - 3x_2)^2$		
$(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$		
$f_3(x) = 4x_1^2 - 2x_1x_2 + 4x_2^2 - 2x_2x_3 + 4x_3^2 - 2x_3x_4$	$X = [-1, 3] \times [-10, 10]\times$	$f_3^* = 5.77$
$+4x_4^2 + 2x_1 - x_2 + 3x_3 + 5x_4$	$[1, 4] \times [-1, 5]$	
$f_4(x) = x_1^3x_2 + x_2^2x_3x_4^2 - 2x_5^2x_1 + 3x_2x_4^2x_5$	$X = [-10, 10]^5$	$f_4^* = -1667708716.3372$
$-\frac{1}{6}x_5^5x_4^3x_3^2$		
$f_5(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$X = [-1000, 1000]^2$	$f_8^* = -1.03162845348366$

Table 1. Test Functions.

Pbs	**NE**			**T₁**			**AF**			**GQF**		
	CPU	ϵ_p	#its	CPU	ϵ_p	#its	CPU	ϵ_p	#its	CPU	ϵ_p	#its
f_1	0.61	10^{-8}	248	146.1	10^{-8}	9148	1.01	10^{-8}	314	1.97	10^{-8}	269
f_2	612.0	1	117552	9.88	10^{-12}	6536	6.12	10^{-12}	1849	3.23	10^{-12}	1071
f_3	307.6	10^{-1}	47738	135.8	10^{-12}	10980	13.41	10^{-13}	5495	15.76	10^{-13}	4121
f_4	0.60	10^{-5}	257	—	—	—	17.1	10^{-4}	3278	7.55	10^{-5}	626
f_5	3.93	10^{-2}	13125	2.61	10^{-14}	4431	8.52	10^{-14}	3009	4.24	10^{-14}	1553
avg	184.8s		35784	73.59s		7774	9.2s		2789	6.5s		1528

Table 2. Comparative Tests between Different Reliable Inclusion Functions

first order $\mathbf{T_1}$, **AF** and **GQF**. All these inclusion functions use the double precision floating point representation.

The first thing that we must compare is the accuracy (ϵ_p) obtained by each method. Therefore, considering equal accuracy, the CPU-times (CPU) and the number of iterations (# its) can then be compared.

Table 2 shows that, the obtained accuracy (ϵ_p) are quiet similar for all the considered inclusion functions (**GQF** is always the most efficient) except for the natural extension **NE** which is clearly less efficient (for f_2, f_3 and f_5); this remark is not true when the functions f_1 and f_4 are considered, because those functions are particular examples constructed to show that sometimes the inclusion function $\mathbf{T_1}$ can be inefficient (here the global optimum is on a side of the initial domain). — in Table 2 means that the inclusion function $\mathbf{T_1}$ cannot produce a result with an accuracy 1 in 100000 iterations; this case is not taken into account for the average of $\mathbf{T_1}$. Furthermore, the inclusion function $\mathbf{T_1}$ is less efficient when the problems have a relatively high number of variables (4 or 5) because the computations of enclosures of the gradient become very expensive, considering the CPU-time, by using an interval automatic differentiation code, [7].

Considering the average in Table 2, the interval Branch-and-Bound algorithm associated with **GQF** proves is own efficiency on these 5 examples providing the best accuracy for all cases, the best CPU-time average and the best average for the number of iterations. A gain around 30% is denoted comparing both CPU-time and number of iterations between **GQF** and **AF**. The gains induced by the **GQF** method are important even if the polynomial functions are quiet simple in the sense that the so-generated quadratic forms are not dense (a lot of elements of the matrix are equal to 0).

To summarize, these numerical results show that interval Branch-and-Bound algorithm based on the **GQF** inclusion function are the most efficient. Furthermore, this seems to be emphasized when a critical size of the problem is reached (4 or 5 variables) and also if the degree of the considered polynomial function is relatively high (≥ 4).

4. Conclusion

In this paper, one recalls the interest to construct new inclusion functions for solving global optimization problems by using an interval Branch-and-Bound algorithm. On five numerical tests, one shows the efficiency of the bounds computed with our general quadratic forms leading to solve difficult global optimization problems in only a few seconds for a high required precision.

References

[1] J. Comba and J. Stolfi, *Affine Arithmetic and its Applications to Computer Graphics*, in Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing), 1993, pp. 9–18.

[2] L. de Figueiredo, *Surface Intersection using Affine Arithmetic*, in Proceedings of Graphics Interface'96, 1996, pp. 168–175.

[3] L. de Figueiredo, R. V. Iwaarden, and J. Stolfi, *Fast Interval Branch and Bound Methods for Unconstrained Global Optimization*, Technical Report 97-08, Institute of Computing, UNICAMP, Campinas, SP - Brazil, 1997.

[4] L. de Figueiredo and J. Stolfi, *Self-Validated Numerical Methods and Applications*, Brazilian Mathematics Colloquium monographs, IMPA/CNPq, Rio de Janeiro, Brazil, 1997.

[5] L. de Figueiredo and J. Stolfi, *Affine Arithmetic: Concepts and Applications*, Numerical Algorithms, 37 (2004), pp. 147–158.

[6] E. Hansen, *A generalized interval arithmetic*, in Lecture Notes in Computer Science no. 29. Interval Mathematics, K. Nickel, ed., Springer Verlag, 1975, pp. 7–18.

[7] R. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, Boston, London, 1996.

[8] F. Messine, *Extension of Affine Arithmetic: Application to Unconstraineded Global Optimisation*, Journal of Universal Computer Science, 8 (2002), pp. 992–1015.

[9] F. Messine and J.-L. Lagouanelle, *Enclosure Methods for Multivariate Differentiable Functions and Application to Global Optimization*, Journal of Universal Computer Science, 4 (1998), pp. 589–603.

[10] F. Messine and A. Touhami, *A General Reliable Quadratic Form: an Extension of Affine Arithmetic*, Technical Report TR/TLSE/05/01, ENSEEIHT-IRIT, Toulouse, France, 2005, to appear in Journal of Reliable Computing.

[11] R. Moore, *Interval Analysis*, Prentice Hall, 1966.

[12] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimisation*, Ellis Horwood Ltd., 1988.

Proceedings of GO 2005, pp. 177 – 181.

A New approach to the Study of the Smith + Smith Conjecture

R. P. Mondaini and N. V. Oliveira

Alberto Luiz Coimbra Institute
for Graduate Studies and Research in Enginerring, COPPE/UFRJ - Technology Centre,
P.O. Box 68511, Rio de Janeiro, RJ, Brazil, rpmondaini@gmail.com

Abstract The search for a point set configurations of the $\mathbb{R}^3$ space which contains the smallest value of the Euclidean Steiner Ratio is almost finished. In the present work we introduce some analytical methods which aim to support a famous conjecture of Discrete Mathematics literature. The relations of this problem with that of molecular architecture in terms of motivation as well as application of the formulae obtained is also emphasized.

Keywords: Euclidean Full Steiner Trees, Steiner Ratio Function, Macromolecular structure modelling.

1. Introduction

One of motivations for modelling the macromolecular structure is the careful observation of the protein data banks. In the present stage of macromolecular evolution, the site of the atoms in a biomacromolecule can be modelled by Steiner points of an usual Steiner Problem. This is the Steiner Tree modelling with three edges converging at a Steiner Point and each pair of edges making angles of 120° therein. This looks as a last stage in the Nature's plans for modelling these structures since it can be shown that this tree conformation is the most stable one [1]. After an exhaustive series of computational experiments by working with evenly spaced given points along a right circular helix of unit radius or,

$$P_j = (\cos(j\omega),\, \sin(j\omega),\, \alpha j\omega); \quad 0 \le j \le n-1 \tag{1}$$

where ω is the angular distance between consecutive points and $2\pi\alpha$ is the helix pitch, we get a sequence of Steiner points in a helix of smallest radius and the same pitch $2\pi\alpha$, *i. e.*, belonging to the same helicoidal surface. All the experimental results led to write the formula for angular distance of consecutive Steiner points along the internal helix as [2]

$$\omega_k = \arctan\left(\frac{y_k}{x_k}\right) + 2\pi\left[\frac{z_k}{2\pi\alpha}\right] + \pi\left[\frac{m}{2}\right], \quad 1 \le k \le n-2 \tag{2}$$

and $\omega_{k+1} - \omega_k \approx b\omega + c$, with $b \approx 1$, $c \approx 0$. The squared brackets stand for the greatest integer value and $m = 1,\, 2,\, 3,\, 4$, according to the quadrant of the angle $\arctan\left(\frac{y_k}{x_k}\right)$.

We have worked with a modified version of a famous algorithm [3]. This reduced search space version of the algorithm was developed in 1996 and is available at *http://www.biomat.org /apollonius*.

From eq. (2), we write the Ansatz for the Steiner Points $S_k(x_k, y_k, z_k)$ as

$$S_k(r(\omega,\alpha)\cos(k\omega), r(\omega,\alpha)\sin(k\omega), \alpha k\omega), \quad 1 \le k \le n-2. \tag{3}$$

By forming a Steiner tree with points P_j and S_k with a 3-Sausage's topology [4], we can obtain from the condition of edges intersecting at $120°$, an expression for the radius function $r(\omega, \alpha)$ or:

$$r(\omega, \alpha) = \frac{\alpha\omega}{\sqrt{A_1(1 + A_1)}} \tag{4}$$

where

$$A_1 = 1 - 2\cos\omega. \tag{5}$$

The restriction to full Steiner trees can be obtained from eq. (1) alone. We then have for the angle of consecutive edges formed by the given points

$$-\frac{1}{2} \le \cos\theta(\omega, \alpha) = -1 + \frac{(1 + A_1)^2}{2(\alpha^2\omega^2 + 1 + A_1)}. \tag{6}$$

2. The generalization of the formulae to sequences of non-consecutive points

The generalization of the formulae derived in section 1 to non-consecutive points [5] along a right circular helix is done by taking into consideration the possibility of skipping points systematically to define subsequences. Let us consider the subsequences of fixed and Steiner points respectively.

$$(P_j)_{m,\, l^j_{max}} \,:\, P_j,\, P_{j+m},\, P_{j+2m},\, ...,\, P_{j+l^j_{max}\cdot m} \tag{7}$$

$$(S_k)_{m,\, l^k_{max}} \,:\, S_k,\, S_{k+m},\, S_{k+2m},\, ...,\, S_{k+l^k_{max}\cdot m} \tag{8}$$

with $0 \le j \le m - 1 \le n - 1, \quad 0 \le k \le m - 1 \le n - 2$ and

$$l^j_{max} = \left[\frac{n - j - 1}{m}\right]; \quad l^k_{max} = \left[\frac{n - k - 2}{m}\right] \tag{9}$$

where $(m - 1)$ is the number of skipped points and the square brackets stand for the greatest integer value. It is worth to say that the sequences (1) and (3) correspond to $(P_0)_{1,n-1}$ and $(S_1)_{1,n-2}$ respectively.

The n points of the helical point set are then grouped into m subsequences and we consider a new sequence of n given points which is written as

$$P_j = \bigcup_{j=0}^{m-1} (P_j)_{m,\, l^j_{max}} \tag{10}$$

Analogously, a new sequence for the union of the subsequences of the Steiner points is introduced in the form

$$S_k = \bigcup_{k=1}^{m-1} (S_k)_{m,\, l^k_{max}} \tag{11}$$

If the points P_{j+lm}, S_{k+lm} are evenly spaced along the helices, their coordinates are given analogously to eqs. (1) and (3) of section 1. We can write,

$$P_{j+lm}\,(\cos(j + lm)\omega, \sin(j + lm)\omega, \alpha(j + lm)\omega) \tag{12}$$

$$S_{k+lm}\,(r_m(\omega, \alpha)\cos(k + lm)\omega, r_m(\omega, \alpha)\sin(k + lm)\omega, \alpha(k + lm)\omega)\,. \tag{13}$$

We can now form Steiner trees with the points P_{j+lm} and S_{k+lm} with the 3-Sausage's topology. We get from the condition of an angle of 120° between each pair of edges meeting at a Steiner point, the expression of radius $r_m(\omega, \alpha)$,

$$r_m(\omega, \alpha) = \frac{m\alpha\omega}{\sqrt{A_m(1 + A_m)}} \tag{14}$$

where

$$A_m = 1 - 2\cos(m\omega). \tag{15}$$

The restriction to full Steiner trees is now obtained from the points P_{j+lm} only and we have

$$-\frac{1}{2} \leq \cos\theta_m(\omega, \alpha) = -1 + \frac{(1 + A_m)^2}{2(m^2\alpha^2\omega^2 + 1 + A_m)} \tag{16}$$

3. A Proposal for a Steiner Ratio Function

After a straightforward but tedious calculation, we get that the Euclidean lengths of the m-spanning tree and of the m-Steiner tree for a great number of fixed points, $n >> 1$, are given by

$$l^m_{SP}(\omega, \alpha) = n\sqrt{m^2\alpha^2\omega^2 + 1 + A_m} \tag{17}$$

$$l^m_{ST} = n\left(1 + m\alpha\omega\sqrt{\frac{A_m}{1 + A_m}}\right). \tag{18}$$

The usual prescription for the Steiner Ratio Function (SRF) lead us to write

$$\rho(\omega, \alpha) = \frac{\min\limits_m(1 + m\alpha\omega\sqrt{\frac{A_m}{1 + A_m}})}{\min\limits_m\sqrt{m^2\alpha^2\omega^2 + 1 + A_m}} \tag{19}$$

The "$\min\limits_m$" in eq. (19) should be understood in the sense of a piecewise function formed by the functions corresponding to the values $m = 1,\ 2,\ ...,\ n - 1$.

The restriction to full trees is applied by remembering that for 3-dimensional macromolecular structure there is not tree built from partial full trees. The tree which represents the "scaffold" of the structure is itself a full tree or it is completely degenerate. It is seen that the surfaces corresponding to eq. (16) for $m \geq 2$ violate the inequality there for a large part of their domain. The surface for $m = 1$ has the largest feasible domain. Our proposal for the SRF function of a helical point set should be written then as

$$\rho(\omega, \alpha) = \frac{1 + \alpha\omega\sqrt{\frac{A_1}{1+A_1}}}{\min\limits_m\ \sqrt{m^2\alpha^2\omega^2 + 1 + A_m}}. \tag{20}$$

We can also suppose necessary bounds on $\rho(\omega, \alpha)$, or

$$\frac{\sqrt{3}}{3} \leq \rho(\omega, \alpha) \leq 1 \tag{21}$$

where the first inequality stands for the Graham-Hwangs's greatest lower bound [6] for the Euclidean Steiner Ratio.

The corresponding ω-region is given by

$$\arccos(1/4) \leq \omega \leq 2\pi - \arccos(1/4). \tag{22}$$

For the region defined by eqs. (21) and (22), we can define the unconstrained optimization problem of eq. (20) in the form

$$\rho(\omega, \alpha) = \max_{m} \rho_m(\omega, \alpha) \tag{23}$$

where the surfaces $\rho_m(\omega, \alpha)$ are

$$\rho_m(\omega, \alpha) = \frac{1 + \alpha\omega\sqrt{\frac{A_1}{1+A_1}}}{\sqrt{m^2\alpha^2\omega^2 + 1 + A_m}}. \tag{24}$$

The first three surfaces meet at a possible global minimum of the surface (20) or (23). Furthermore, in this point, beyond the 3-Sausage's topology for the Steiner tree structure we also have a 3-Sausage's configuration for the fixed points or the familiar set of vertices of regular tetrahedra glued together at common faces [7]. This non trivial solution can be written

$$\omega_R = \pi - \arccos(2/3); \quad \alpha_R = \sqrt{30}\,[9(\pi - \arccos(2/3))] \tag{25}$$

and

$$\rho(\omega_R, \alpha_R) = \frac{1}{10}(3\sqrt{3} + \sqrt{7}) = 0.78419037337... \tag{26}$$

which is the value assumed by the main conjecture of authors of ref. [4] for the Steiner Ratio in $\mathbb{R}^3$ with Euclidean distance.

It is worth to notice that there is always a pair of values which give the same value of ρ, namely (ω, α) and $\left(2\pi N - \omega, \frac{\omega\alpha}{2\pi N - \omega}\right)$, $N \in \mathbb{Z}$. This means that

$$\bar{\omega} = \pi + \arccos(2/3), \quad \bar{\alpha} = \sqrt{30}\,[9(\pi + \arccos(2/3))]$$

correspond to the same ρ-value, eq. (26).

4. The Existence of a Global Minimum. The Weierstrass Theorem

The use of the Weierstrass Theorem has as a requirement the definition of a compact domain. We shall define it in the diagram below where we have depicted the curves $\rho_m = 1$, $m = 1, 2, 3, 4, 5$.

The compact region can be defined as

$$(H\rho_1 - \bigcup_{k \geq 2} H\rho_k) \bigcap \{(\omega, \alpha)| \arccos(1/4) \leq 2\pi - \arccos(1/4)\} \tag{27}$$

where $H\rho_1$, $H\rho_k$ are the hypographs of the functions $\rho_1 = 1$, $\rho_k = 1$, respectively. Since the functions $\rho_k(\omega, \alpha)$ are continuously decreasing from the boundaries of this compact domain towards its interior, the Weierstrass Theorem guarantees the existence of a global minimum inside it.

5. Concluding Remarks

This work is one of the last steps in our effort to proof the Smith+Smith Conjecture. There are also nice results to be reported about the study of configurations in the neighbourhood of this value of the Euclidean Steiner Ratio. They are associated to a continuous description of molecular chirality [8, 9] and their influence in the strengthening of molecular interaction. Studies of this subject are now in progress and will be published elsewhere.

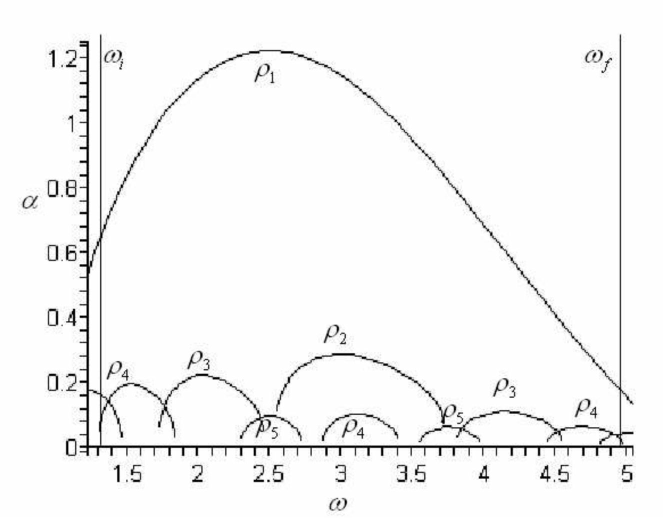


Figure 1. The compact region to be used in the Weierstrass Theorem $\omega_i = \arccos(1/4)$, $\omega_f = 2\pi - \arccos(1/4)$ according the Graham-Hwang's lower bound.

References

[1] R.P.Mondaini. (2004). "The Geometry of Macromolecular Structure: Steiner's Points and Trees", Proc. Symp. Math. Comp. Biology IV, 347-356.

[2] R.P.Mondaini. (2001). Private communication to A. Ochoa.

[3] W. D. Smith. (1992). "How to find minimal Steiner trees in Euclidean d-Space", Algorithmica 7, 137-177.

[4] W. D. Smith, J. Mac Gregor Smith. (1995). "The Steiner Ratio in 3D Space", Journ. Comb. Theory A69, 301-332.

[5] R.P.Mondaini. (2005). "Modelling the Biomacromolecular Structures with Selected Combinatorial Optimization Techniques", arχv: math-ph/0502051 v 1 - 25 Feb 2005.

[6] R.L.Graham, F. K. Hwang. (1976). "Remarks on Steiner Minimal Trees I", Bull. Inst. Acad. Sinica 4, 177-182.

[7] J. M. Smith, B. Toppur (1996). "Euclidean Steiner Minimal Trees, Minimal Energy Configurations and the Embedding Problem of Weighted Graphs in E^3",Discret Appl. Math. 71, 187-215.

[8] R.P.Mondaini. (2003). "Proposal for Chirality Measure as the Constraint of a Constrained Optimization Problem", Proc. Symp. Math. Comp. Biology III, 65-74.

[9] R.P.Mondaini. (2004). "The Euclidean Steiner Ratio and the Measure of Chirality of Biomacromolecules", Gen. Mol. Biol. 4, 658-664.

Fitting separable nonlinear spectrotemporal models

Katharine M. Mullen[1], Mikas Vengris[2], and Ivo H.M. van Stokkum[1]

[1] *Department of Physics and Astronomy, Faculty of Sciences, Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands,* {kate}{ivo}@nat.vu.nl

[2] *Department of Quantum Electronics, Faculty of Physics, Laser Research Center, Vilnius University, Sauletekio 10, LT10223 Vilnius, Lithuania,* Mikas.Vengris@ff.vu.lt

Abstract According to the superposition principle time-resolved spectra may be described by a product of concentration profiles and spectra of components. We review here methods for fitting the resulting separable nonlinear spectrotemporal model based on alternating least squares and variable projection algorithms. Comparison of the algorithms on simulated spectrotemporal data reveals that the variable projection-based techniques are advantageous in terms of iterations and function evaluations required to convergence, sensitivity of computational efficiency to starting values, and accuracy of linear approximation standard error estimates of parameter values.

Keywords: Time-resolved spectra, global analysis, separable nonlinear models, variable projection, alternating least squares.

1. Introduction

In many applications (an overview of which is found in [5]) it is possible to separate the parameters x of the unconstrained minimization problem $Minimize\ \varphi(x)$, $x \in R^n$ into $x = (y, z)$ where $y \in R^p$, $z \in R^q$, $p + q = n$, and the subproblem

$$Minimize\ \varphi(y, z), \tag{1}$$

is an overdetermined linear system with the parameters y occurring linearly (the parameters y are termed *conditionally linear*, since they may be found by solving a linear optimization problem given fixed nonlinear parameters z). Then the n-dimensional unconstrained problem may be replaced by the q-dimensional problem

$$Minimize\ \varphi(y(z), z), \tag{2}$$

where $y(z)$ denotes a solution of (1) which must be computed each time φ is evaluated. A problem in which the variables separate in this sense is

$$Minimize \parallel vec(F(z)H - G) \parallel_2 \tag{3}$$

where $F \in R^{m\times q}$ is a matrix determined by nonlinear parameters z, $H \in R^{n\times p}$ is a matrix of conditionally linear parameters, $G \in R^{m\times n}$ is a response matrix, and $vec(.)$ denotes transformation to a vector representation. A problem of the form contained in (3) arises in fitting spectrotemporal models to time-resolved spectral measurements collected in a matrix Ψ of m time points and n wavelengths, the columns of which represent decay traces and the rows of which represent time-gated spectra. By the superposition principle, application of a kinetic model describing the evolution of the concentrations $c_l(\phi)$ of components contributing to Ψ

results in a model of general form

$$\Psi = C(\phi)E^T + \Xi = \sum_{l=1}^{n_{comp}} c_l(\phi)e_l^T + \Xi \qquad (4)$$

where C is an m by n_{comp} matrix dependent on intrinsically nonlinear parameters ϕ, with n_{comp} representing the number of distinct components contributing to Ψ, E is an n by n_{comp} matrix of conditionally linear parameters, and Ξ is an m by n residual matrix with spherical Gaussian distribution [13]. The c_l (columns of C) represent concentration profiles of components contributing to Ψ. The e_l (columns of E) represent spectra of these components. Data of this form arises in the investigation of photophysical systems by means of time-resolved optical spectroscopy; typical experiments result in Ψ containing on the order of 1,000-100,000 measurements. Estimates for nonlinear parameters ϕ describing the evolution of component concentrations and conditionally linear parameters E describing the component spectra provide a concise description of Ψ and yield insight into the underlying system dynamics.

2. Optimization methods

In seeking ϕ that solve

$$Minimize \parallel vec(C(\phi)E^T - \Psi) \parallel_2 \qquad (5)$$

both the alternating least squares method (e.g., [3], [14]), and the variable projection method ([9], [13]) are often applied. [14] states "alternating least squares regression is rapidly becoming the most commonly used method to solve spectroscopic image analysis problems based on either a bilinear or trilinear model", despite "slow processing speed due to convergence problems". Here alternating least squares and several variants of variable projection are presented in terms of their gradient in ϕ-space.

We consider two gradients derived from variable projection: the original method introduced by Golub and Pereyra [6], [4] (GP), and the approximation introduced by Kaufman [7] (KAUF). The alternating least squares gradient (ALS) gradient we consider was introduced by Wold [15] as NIPALS. We also consider a finite-difference approximation of the gradient (NUM).

Using the notation of [1], let the derivative of C with respect to the nonlinear parameters be denoted $C_\phi = \frac{dC}{d\phi^T}$. Applying the QR decomposition, $C = QR = [Q_1\ Q_2][R_{11}\ 0]^T$, where Q is $m \times m$ and orthogonal and R is $m \times n_{comp}$. Assuming C is of full column rank, $C^+ = R_{11}^{-1}Q_1^T$. Using $\hat{E}^T(\phi) = C^+\Psi$, the matrix of residuals $\Psi - C(\phi)\hat{E}^T = (I - C(\phi)C^+(\phi))\Psi$. The algorithms ALS (both the ALS-GN and ALS-LS variants), KAUF, GP, NUM may then be defined as follows, where "convergence" is some appropriate stopping criterion. For compactness of representation the iteration subscript s is suppressed.

```
ALGORITHMS ALS, KAUF, GP, NUM:
    1. Choose starting $\phi$ approximately
    2. For $s$ := 1, 2 . . . until convergence do
        Determine the gradient in $\phi$-space according to:
            ALS := $-C_\phi C^+\Psi$
            KAUF := $-Q_2Q_2^TC_\phi C^+\Psi$
            GP := $-Q_2Q_2^TC_\phi C^+\Psi - Q_1R_{11}^{-T}C_\phi^TQ_2Q_2^T\Psi$
            NUM := finite difference approximation of $\frac{d(I-CC^+)}{d\phi}\Psi$
        $\phi_{s+1}$ := STEP($\phi_s$, gradient, . . .)
```

By basing our discussion and implementation of the ALS, KAUF, GP, NUM algorithms on the direction in ϕ-space that ϕ is incremented each iteration, we leave undetermined the method (referred to as STEP in the description of the algorithms) of determining the step-size in this direction, allowing the question of an optimal method to remain separate from the differences in the gradient proscribed by each algorithm. We introduce two varieties of ALS differing in the STEP method. The first (ALS-GN) makes a Gauss-Newton step given the ALS gradient. The second (ALS-LS) makes a Gauss-Newton step augmented by a line search in the gradient direction until the sum-square error (SSE) is seen to increase. For KAUF, GP, and NUM, we use a Gauss-Newton step. Implementation of algorithms was based on the *nls* package from the R language and environment for statistical computing [10]. Simulation studies (utilizing the *minpack* package from [10]) have indicated that for all algorithms under consideration, replacement of the Gauss-Newton step with a Levenberg-Marquardt step does not alter performance.

We now summarize some prior results comparing subsets of the methods under consideration. Ruhe and Wedin [11] have shown that for starting ϕ close to the solution, the asymptotic convergence rates of KAUF and GP are superlinear whenever application of Gauss-Newton to the unseparated parameter set ($\phi + E$) has a superlinear rate of convergence, and that ALS always has only a linear rate of convergence. Bates and Lindstrom demonstrated that for a simple model having a single nonlinear parameter the performance of KAUF and GP was similar [1]. Gay and Kaufman also performed a comparison of KAUF and GP on models inappropriate for time-resolved spectra using datasets containing < 70 data points [8]. We have not found in the literature a clear comparison of alternating least squares and variable projection methods for fitting time-resolved spectral data to appropriate models, despite widespread application of both classes of algorithms.

3. Simulation

For a simulation study we generated data Ψ with $n_{comp} = 2$, $\phi = \{k_1, k_2\}$, $c_l = \exp(-k_l t)$, where c_l is the column of C describing the lth concentration profile. The spectral shapes are described by a Gaussian in the energy domain, with $\bar{\nu} = \lambda^{-1}$, so that

$$e_l(\mu_{\bar{\nu}}, \sigma_{\bar{\nu}}) = a_l \bar{\nu}^5 \exp(-ln(2)(2(\bar{\nu} - \mu_{\bar{\nu}})/\sigma_{\bar{\nu}})^2), \tag{6}$$

where e_l is the column l of E describing the lth spectrum, with parameters $\mu_{\bar{\nu}}$ and $\sigma_{\bar{\nu}}$ for the location and width of e_l, and amplitude parameter a_l.

Table 1. Rate constants, spectral parameters, and amplitudes for simulated Ψ

component	k	$\mu_{\bar{\nu}}$	$\sigma_{\bar{\nu}}$	a
1	.5	22	9	1
2	.6	18	8	2

The simulated data were inspired by real datasets [12], [13] representing a mixture of two emitting states with close kinetic rate constants (lifetimes) decaying exponentially and with overlapping spectra. The dataset was made to contain $m = 51$ time points equidistant in the interval 0-2 ns and $m = 51$ wavelengths equidistant in the interval 350-550 nm. The simulation parameters are collated in Table 1. Gaussian noise with zero mean was was added, with the width σ of the noise distribution equal to 3×10^{-3} of the maximum of the data.

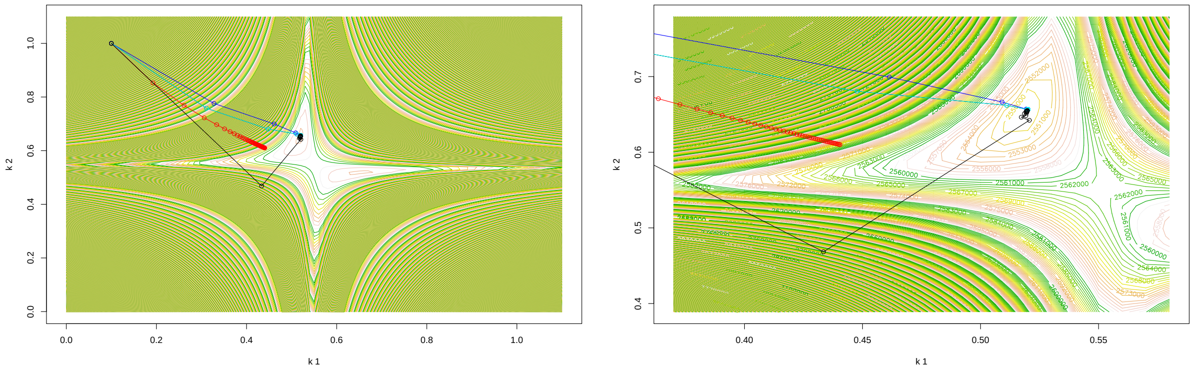


Figure 1. Contour map of the sum square of residuals $||vec(\Psi - CE^T)||^2$ as rate constants k_1, k_2 vary, at a relatively large (Left) and relatively small (Right) scale. The progress of ALS-GN (red), ALS-LS (black), KAUF (cyan), and GP/NUM (both blue) is depicted from starting values $k_1 = .1, k_2 = 1$; rate constant estimates after each iteration are marked with a circle.

4. Results

For a simulated dataset, we have evaluated the SSE $||vec(\Psi - CE^T)||^2$ as rate constants k_1, k_2 vary, with the result being the surface shown in Figure 1. Figure 1 also shows the values found by each algorithm under consideration for each of 50 iterations from the starting values $k_1 = .1, k_2 = 1$. KAUF, GP, ALS-LS and NUM converge on the same (globally optimal) solution in 4 iterations, (note however that the performance of ALS-LS depends on the choice of an efficient line-search method). ALS-GN is not convergent, and from this case study and others we conclude that the Gauss-Newton step coupled with the ALS-gradient is not sufficient for the solution of typical estimation problems in time-resolved spectral modelling.

Performance of the algorithms from a range of starting values and on variants of the dataset under different noise realizations was examined. For cases in which the globally optimal parameter values are located at the end of a valley with respect to the starting values, we observe that the performance of ALS-LS is very much hampered in terms of iterations required to convergence in comparison to KAUF, GP, and NUM. A plot of the SSE surface (as in Figure 1) in this case shows that ALS-LS follows a zig-zagging path between the walls of the valley, toward the globally optimal solution to which it eventually converges.

In order to examine the properties of standard error estimates as returned by the fitting algorithms under consideration, 1000 realizations of the dataset were simulated. For each realization, the $deviation(k) = \hat{k} - k$, where $\hat{k}$ is the estimated rate constant value, and k is the value used in simulation, the (linear approximation) standard error ($\hat{\sigma}_{\hat{k}}$), derived from $cov(\hat{\phi}) = \hat{\varsigma}^2(J^TJ)^{-1}$, where $\hat{\varsigma}^2$ denotes the estimated variance and J is the Jacobian (the gradient of the residual vector) evaluated at $\hat{\phi}$, and the ratio of these two calculations (the studentized parameter deviation [2], [12]), were calculated. Table 2 reports root mean square (RMS) results.

At the level of precision collated in Table 2, results for NUM, KAUF and GP are identical. NUM and GP only differ from KAUF in the 3rd decimal place of deviation/$\hat{\sigma}_{\hat{k}}$, and from each other in the 6th. RMS $deviation/\hat{\sigma}_{\hat{k}}$ is expected to be 1 in linear models, and hence the degree to which this ratio approximates 1 can be used as a measure of the applicability of the standard error returned by the respective algorithms. Under the ALS gradient, $\hat{\sigma}_{\hat{k}}$ is much too small, and not useful as a measure of confidence in parameter value estimates.

Likelihood-based confidence regions may be constructed around parameter estimates based on the likelihood ratio between the sum square of residuals $S(\hat{\phi}) = ||vec(\Psi - CE^T)||^2$ at the solution and at values $S(\phi)$ around the solution as $\phi = \{k_1, k_2\}$ is varied. The confidence level

Table 2. Root mean square deviation and standard error of nonlinear parameters

		ALS-LS	KAUF/GP/NUM
RMS deviation ($\hat{k} - k$)	k_1	0.022	0.022
	k_2	0.025	0.025
RMS $\hat{\sigma}_{\hat{k}}$	k_1	0.00033	0.021
	k_2	0.00048	0.027
RMS (deviation/$\hat{\sigma}_{\hat{k}}$)	k_1	55	1.3
	k_2	37	1.2

$1 - \alpha$ is determined as

$$1 - \alpha = F\left(P, N - P, (N - P)/P\frac{S(\phi) - S(\hat{\phi})}{S(\hat{\phi})}\right) \tag{7}$$

where F is the cumulative F-distribution, $N = (\text{times} - n_{comp})(\text{wavelengths}) = (51 - 2)(51)$, and $P = n_{comp} = 2$. The resulting contour plot of confidence regions about the parameter estimates is shown in in the left panel of Figure 2. For comparison, the linear approximation confidence regions calculated from $cov(\phi)$ for KAUF, GP, or NUM are shown in the right panel of Figure 2. Note that the linear approximation confidence regions are slightly too small as compared to the likelihood-based confidence regions, which is consistent with the slight underestimation of $\hat{\sigma}_{\hat{k}}$ in Table 2, as measured by the overshoot of deviation/$\hat{\sigma}_{\hat{k}}$ to 1.

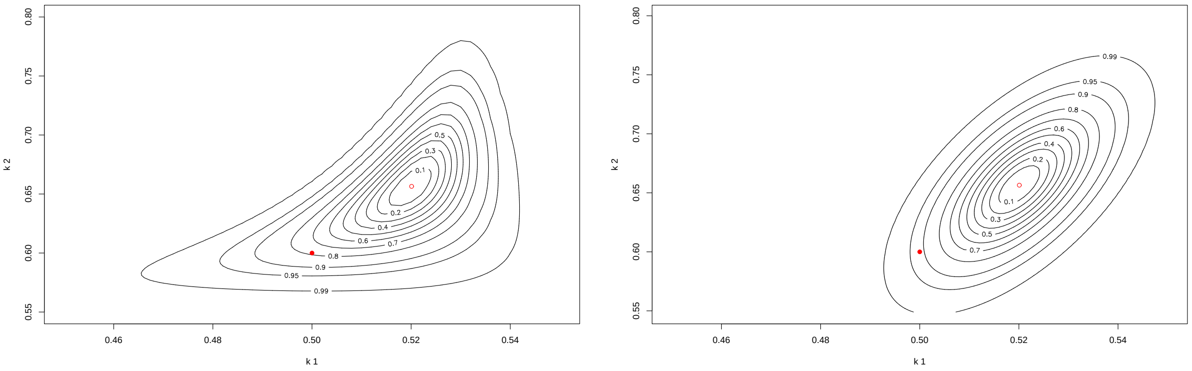


Figure 2. For the dataset depicted in Figure 1, (Left) contour map of confidence levels $1 - \alpha$ as determined by Equation 7 as rate constants k_1, k_2 vary, (Right) linear approximation confidence regions as found using KAUF, GP, or NUM for the same levels as at left. In both (Left) and (Right) the closed red circle marks the rate constant values used in simulation, and the open red circle marks the globally optimal values found by KAUF, GP, NUM, and ALS-LS.

5. Conclusions

We observe that the performance of GP, KAUF, NUM, and ALS-LS is equal or near-equal in terms of the accuracy of returned parameter estimates after convergence. In terms of iterations and function evaluations required to convergence, ALS-LS is not favorable in comparison to GP, KAUF, and NUM. The lack of computational efficiency under ALS-LS is due to the additional function evaluations necessary to perform the line-search each iteration, and to sensitivity to the choice of starting values that results in estimates that zig-zag through parameter space over iterations from some starting values. The standard error estimates returned by KAUF, GP, and NUM are usable as a measure of confidence in the associated parameter values, and allow, e.g., the construction of confidence regions about parameter estimates. The standard error estimates returned by ALS-LS do not allow inference regarding the associated parameter estimates, which is a significant drawback in practical application. Under the criteria of computational efficiency and goodness of standard error estimates, we thus conclude that variable projection based algorithms are advantageous over alternating least-squares based techniques, and are to be preferred for application to this problem domain. Of the variable projection techniques, the KAUF algorithm is the least computationally expensive as evident from comparison of the equations determining the gradient. Hence we recommend that KAUF be considered the optimal variable projection-based technique for fitting spectrotemporal models to time-resolved spectra.

6. Acknowledgments

This research was funded by Computational Science grant #635.000.014 from the Netherlands Organization for Scientific Research (NWO).

References

[1] Douglas M. Bates and Mary J. Lindstrom. Nonlinear least squares with conditionally linear parameters. In *Proceedings of the Statistical Computing Section*, pages 152–157, New York, 1986. American Statistical Association.

[2] Douglas M. Bates and Donald G. Watts. *Nonlinear regression analysis and its applications*. John Wiley & Sons, 1988.

[3] M. Blanco, A.C. Peinado, and J. Mas. Elucidating the composition profiles of alcoholic fermentions by use of ALS methodology. *Analytica Chimica Acta*, in press, 2005.

[4] G. H. Golub and V. Pereyra. The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate. *SIAM J. Num. Anal.*, 10:413–432, 1973.

[5] Gene Golub and Victor Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:R1–R26, April 2003.

[6] G.H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. Technical report, Stanford University, Department of Computer Science, 1972.

[7] L. Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975.

[8] L. Kaufman and D. Gay. Tradeoffs in algorithms for separable and block separable nonlinear least squares. In R. Vichnevetsky and J. J. H. Miller, editors, *IMACS '91, Proceedings of the 13th World Congress on Computational and Applied Mathematics*, pages 157–158, Dublin, 1991. Criterion Press.

[9] John F. Nagle. Solving complex photocycle kinetics - theory and direct method. *Biophysical Journal*, 59:476–487, 1991.

[10] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0.

[11] Axel Ruhe and Per Ake Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):318–337, July 1980.

[12] Ivo H.M. van Stokkum. Parameter precision in global analysis of time-resolved spectra. *IEEE Transactions on Instrumentation and Measurement*, 46(4):764–768, 1997.

[13] Ivo H.M. van Stokkum, Delmar S. Larsen, and Rienk van Grondelle. Global and target analysis of time-resolved spectra, and erratum. *Biochimica et Biophysica Acta*, 1657, 1658:82–104, 262, 2004.

[14] Ji-Hong Wang, Philip K. Hopke, Thomas M. Hancewicz, and Shuliang L. Zhang. Application of modified alternating least squares regression to spectroscopic image analysis. *Analytica Chimica Acta*, 476:93–109, 2003.

[15] H. Wold and E. Lyttkens. Nonlinear iterative partial least squares (NIPALS) estimation procedures. *Bull. ISI*, 43:29–51, 1969.

Global Optimisation Challenges in Robustness Programming Bounding Methods

Niels J. Olieman and Eligius M.T. Hendrix

Wageningen University, Hollandsweg 1, 6706 KN, Wageningen, The Netherlands, {Niels.Olieman, Eligius.Hendrix}@wur.nl

Abstract Robustness Programming is introduced as a framework for expressing Robustness optimisation problems. Robustness bounding techniques can be used for feasible domain range reduction. Robustness bounding techniques are based on finding largest enclosed and enclosing shapes. Global optimisation techniques are essential for finding Robustness bounds.

Keywords: Robustness Programming, Probability Theory, Global Optimisation.

1. Introduction

Consider a wooden table with 4 table legs, each with a diameter of 20cm and a tabletop that is 10cm thick. There are 10 persons sitting at the table, telling jokes and having a good time with beer and wine. Suppose that the *uncontrolable factors* such as bumping and pushing against the table, hardly moves the table. Since the desirable table properties such as *stability* and *maximum weight support* appear not sensitive to the considered uncertainties, the table can be called robust. A similar table, where the mentioned dimensions are twice as big, is probably *more* robust. Apparently an object can have some level of *robustness*, since some objects can be more robust than others.

As another example, consider a container of margarine. Typically the container is stored in the refrigerator, but is warming up towards room temperature, during breakfast and lunch. The daily cycle of cooling down and warming up can have an effect on the structure of the margarine, such that over time the margarine does not look appealing anymore. A margarine product which appears not to be effected by the temperature fluctuations can be called robust. A margarine product which maintains the desirable properties for the widest range of temperature fluctuations has the highest robustness.

Robustness is related to uncertain uncontrolable factors. There exist alternative modelling paradigms for modelling uncertainty and each gives rise to alternative measures for robustness. For instance, Taguchi [8] and Markowitz [6] implicitly model robustness using the *mean* and *variance* statistics. Ben-Tal [1] uses a binary approach: an object is either robust or not, i.e. an object either maintains its desired properties under all possible conditions or not.

Focus is on the computation and optimisation of a robustness measure. The basic assumptions are, (i) that object requirements can be quantified and (ii) uncertain object properties can be modelled mathematically as a function of controllable and uncontrollable factors. The vector x represents the controllable factors, v the uncontrollable factors and S object properties are represented by the functions $u_s(x, v)$ for $s = 1, .., S$. Non-negative values of $u_s(x, v)$ represent the *desired* object properties.

As a simplified food product design example, consider a product which is a mix of $i = 1, ..., I$ raw materials, where x_i represents the proportion of raw material i in the mix. Let for instance x_8 represent the proportion of orange juice in the food product. It is likely that the concentration of vitamin C in the orange juice will vary, depending on the country of origin, weather conditions during growth and harvest period and storage time. Likewise, there is uncertainty in the vitamin C concentration of the other raw materials. The vector element v_i represents the uncontrollable vitamin C concentration in raw material i. If food scientists determined that there should be at least 0.05% vitamin C in the food product, then the following condition should be fulfilled:

$$u_1(x, v) = v'x - 0.05 \geq 0 \tag{1}$$

For the general case, a set of uncontrollable factors can be defined for which it is certain that the object properties fulfill the requirements:

$$\mathbb{H}(x) := \{v \in \mathbb{V} | u_s(x, v) \geq 0, s = 1, .., S\}$$

where $\mathbb{V}$ is the set of all possible realisations of v. The set $\mathbb{H}(x)$ is called the *Happy set*, since elements of this set correspond to the favourable situation that all uncertain object properties are as intended. A design x is called *robust*, if and only if the design will perform as intended for *all* possible variations in $\mathbb{V}$:

$$\mathbb{H}(x) \equiv \mathbb{V} \Longleftrightarrow \text{"}x \text{ is robust"}$$

Consequently, we will say a design has *some* level of *robustness*, if the design only performs as intended for a *subset* $\mathbb{S} \subset \mathbb{V}$ of all possible realisations of v:

$$\mathbb{H}(x) \subset \mathbb{V} \Longleftrightarrow \text{"}x \text{ is \textit{less than} robust"}$$

Perhaps the most common approach in science for modelling uncertainty, is studied in the science field called *probability theory* [4] [5]. In probability theory, the notions *random vector* and *probability space* are formally introduced, which can be used to model a collection of uncertain events. The following notation is used:

1. The probability space is defined as $(\mathbb{V}, \mathcal{V}, \Pr)$, with *sample space* or *support set* $\mathbb{V} \subseteq \mathbb{R}^N$; the σ-field $\mathcal{V}$ of all subsets of $\mathbb{V}$; a probability measure $\Pr : \mathcal{V} \to [0, 1]$ and by definition $\Pr(\emptyset) = 0$ and $\Pr(\mathbb{V}) = 1$
2. The stochastic vector (in **bold**) $\mathbf{v}$ has *realisations* v in the support set: $v \in \mathbb{V}$.
3. The *probability* that $\mathbf{v}$ will have realisations in a subset $\mathbb{S} \in \mathcal{V}$, $\mathbb{S} \subseteq \mathbb{V}$ is formally denoted with $\Pr(\mathbb{S})$. Also an informal notation is used, which makes reference to the connected random vector: $\Pr\{\mathbf{v} \in \mathbb{S}\} = \Pr(\mathbb{S})$.

Under the assumption that it is appropriate to model the uncontrollable factors as stochastic variates, authors such as Du and Chen [3], Bjerager [2] and Nie et al. [7] studied probabilistic robustness computation and optimisation approaches with encouraging results.

Robustness $R(x)$ can be expressed in a probabilistic way: the probabilistic robustness of an object is the probability that an object will have desired properties:

$$R(x) = \Pr\{\mathbf{v} \in \mathbb{H}(x)\} \tag{2}$$

The framework for robustness optimisation problems is here defined as Robustness Programming (RP) and can be formulated as:

$$R^* = \max_{x \in \mathbb{R}^I} [\Pr\{\mathbf{v} \in \mathbb{H}(x)\}] \tag{3}$$

In practice, there is often not a closed form expression to compute (2). Alternatively, one can *estimate* or find (exact) upper and lower *bounds* for (2). Typically estimation algorithms can be *slow* and can be inefficient to start with, in the context of robustness optimisation. An improvement would be to explore efficient $R(x)$-bounding approaches to reduce the feasible domain of x. The notation for upper bound is $^+R(x)$ and for lower bound is $^-R(x)$. An approach can be, to start with computing a lower bound for the optimum $^-R^* = \max\limits_{x\in\mathbb{R}^I} [^-R(x)]$, followed by reducing the feasible domain

$$\mathbb{X} = \left\{x \in \mathbb{R}^n |^+R(x) \geq^- R^*\right\} \tag{4}$$

and reduce RP roblem (3) to

$$R^* = \max_{x\in\mathbb{X}} \left[\Pr\left\{\boldsymbol{v} \in \mathbb{H}(x)\right\}\right] \tag{5}$$

The optimal bound solution $^-x^*$ corresponding $^-R^*$, can be used as a starting point when using an iterative optimisation method.

Feasible domain range reduction is an application example of $R(x)$-bounding approaches. From a practical point of view, only knowing the optimal $R(x)$ bounds can already be valuable information and sufficient for decision making. One can conclude that $R(x)$-bounding methods can be relevant in the context of robustness optimisation.

The $R(x)$-bounding methods are based on the following basic principle: Consider an *enclosed* shape $^-\mathbb{H}(x) \subseteq \mathbb{H}(x)$ and an *enclosing* shape $^+\mathbb{H}(x) \supseteq \mathbb{H}(x)$. Based on these shapes, for any two points x^a and x^b for which it holds that

$$^-\mathbb{H}(x^a) \supseteq^+ \mathbb{H}(x^b) \left(\supseteq \mathbb{H}(x^b)\right)$$

it follows that

$$R(x^a) \geq R(x^b) \tag{6}$$

Even without knowing the actual probability space of v. Of course, this method only works if $\mathbb{H}(x)$ is bounded. In this study, three types of shapes in the v-space are considered:

$$\begin{array}{ll} \text{1-norm (diamond):} & \mathbb{D}(c, r_d) = \left\{v \in \mathbb{R}^N \left| \sum\limits_{n=1}^{N} |v_n - c_n| \leq r_d \right.\right\} \\ \text{2-norm (ball):} & \mathbb{B}(c, r_b) = \left\{v \in \mathbb{R}^N |(v-c)'(v-c) \leq r_b^2\right\} \\ \infty\text{-norm (cube):} & \mathbb{C}(c, r_c) = \left\{v \in \mathbb{R}^N |\forall_{1\leq n\leq N} : -r_c \leq v_n - c_n \leq r_c\right\} \end{array} \tag{7}$$

where vector c is the centre of the shape and r_d, r_b and r_c the radius. By definition it holds that

$$\mathbb{D}(c, r_d) \subset \mathbb{B}(c, r_b) \subset \mathbb{C}(c, r_c) \tag{8}$$

for $r_d = r_b = r_c$. For simplicity reasons, the centre of the shapes is fixed at $c = E[\mathbf{v}]$. For fixed x, the smallest shapes enclosing $\mathbb{H}(x)$, are defined by

$$\begin{array}{ll} \text{1-norm (diamond):} & ^+\mathbb{D}\left(c, \max\limits_{v\in\mathbb{H}(x)} \sum\limits_{n=1}^{N} |v_n - c_n|\right) \\ \text{2-norm (ball):} & ^+\mathbb{B}\left(c, \max\limits_{v\in\mathbb{H}(x)} \sqrt{(v-c)'(v-c)}\right) \\ \infty\text{-norm (cube):} & ^+\mathbb{C}\left(c, \max\limits_{n} \max\limits_{v\in\mathbb{H}(x)} |v_n - c_n|\right) \end{array} \tag{9}$$

For fixed x, the biggest shapes enclosed by $\mathbb{H}(x)$, can be determined by identifying that requirement s, that is in distance closest to nominal value c. Alternatively, when blowing up the shape, one should find out which constraint s, $s = 1, .., S$ is touched first:

$$\begin{array}{ll} \text{1-norm (diamond):} & ^-\mathbb{D}\left(c, \min\limits_{s} \min\limits_{v} \left\{ \sum\limits_{n=1}^{N} |v_n - c_n| \middle| u_s(x, v) = 0\right\}\right) \\ \text{2-norm (ball):} & ^-\mathbb{B}\left(c, \min\limits_{s} \min\limits_{v} \left\{ \sqrt{(v-c)'(v-c)} \middle| u_s(x, v) = 0\right\}\right) \\ \infty\text{-norm (cube):} & ^-\mathbb{C}\left(c, \min\limits_{n} \min\limits_{s} \min\limits_{v} \left\{ |v_n - c_n| \middle| u_s(x, v) = 0\right\}\right) \end{array} \tag{10}$$

Any of the enclosed or enclosing shapes can be used for the computation of respectively a robustness lower bound or upper bound. Since c is fixed, the radius of any of the shapes, is proportional to the corresponding probability bound. Thus the comparison of the bounds, can be based on the radius in combination with property (8).

The challenges for GO are two-fold:

1. For the maximisation and minimisation problems inside the definitions (9) and (10), it is crucial to find global optima, since only in that situation a bound can be guaranteed.
2. The $R(x)$-bounding approach is competing with $R(x)$-estimation. In the situation a Monte Carlo sampling approach for estimating $R(x)$ and N samples are sufficient for estimating $R(x)$ at a satisfying accuracy level, then it only makes sense if the bounding approach can find a bound within $< N$ function evaluations. In practice this means fast global optimisation routines are needed.

As an illustration see figure 1 where

$$u(x,v) = 2 - (x_1 * (v_1 - p_1))^2 + (x_2 * (v_2 - p_2))^2$$

and

$$\mathbb{H}(x) = \left\{ v \in \mathbb{R}^2 | u(x,v) \geq 0 \right\}$$

and exogenous restrictions for the initial feasible domain are given by

$$\mathbb{X} = \left\{ x \in \mathbb{R}^2 | 2x_1 = x_2 \right\}$$

In the illustration $E(\mathbf{v}) = c \neq p$, such that graphically speaking: p is little above and to the left from c. Let

$$^{-}r_c(x) = \min_n \min_s \min_v \left\{ |v_n - c_n| \middle| u_s(x,v) = 0 \right\}$$

be the radius of the cube enclosed by $\mathbb{H}(x)$ and

$$^{+}r_c(x) = \max_n \max_{v \in \mathbb{H}(x)} |v_n - c_n|$$

be the radius of the cube enclosing $\mathbb{H}(x)$. The left part of figure 1 illustrates the v-space and the right part the x space. The lower left part of figure 1, illustrates the search for the largest enclosed cube. The inner cube is a subset of $\mathbb{H}(x^{\mathrm{I}})$ which intersects with the closure of $\mathbb{H}(x^{\mathrm{I}})$ and thus corresponds to the global solution for the optimisation problem in the definition of $^{-}r_c(x^{\mathrm{I}})$. The other three larger cubes are partially outside $\mathbb{H}(x^{\mathrm{I}})$ and correspond to local solutions for the optimisation problems in the definition of $^{-}r_c(x^{\mathrm{I}})$. In figure 1, the happy set corresponding point x^{III} is enclosed by a cube with a radius identical to the radius of the cube belonging to point x^{I}: $^{-}r_c(x^{\mathrm{I}}) =^{+} r_c(x^{\mathrm{III}})$. Without knowing the details of the probability space $(\mathbb{V}, \mathcal{V}, \mathrm{Pr})$, the bounds guarantee that the probability mass of $\mathbb{H}(x^{\mathrm{III}})$ is smaller than or equal to the probability mass of $\mathbb{H}(x^{\mathrm{I}})$. Thus $R(x^{\mathrm{I}}) \geq R(x^{\mathrm{III}})$. Therefore the feasible domain $\mathbb{X}$ can be reduced to

$$\mathbb{X}' := \left\{ x \in \mathbb{X} | x_1 \leq x_1^{\mathrm{III}} \right\}$$

A second illustration of the R(x)-bounding methods, is the application in the context of conditional MC sampling for estimating $R(x)$. In the situation that the probability mass of shape $^{-}\mathbb{H}(x)$ inside $\mathbb{H}(x)$ is known, then only samples outside $^{-}\mathbb{H}(x)$ are needed to estimate $R(x)$. The largest *diamond* is an example of such an enclosed shape: $^{-}\mathbb{D}(c,^{-} r_d^*(x)) \subseteq \mathbb{H}(x)$, with $^{-}r_d^*(x) = \min_s \min_v \left\{ \sum_{n=1}^{N} |v_n - c_n| \middle| u_s(x,v) = 0 \right\}$. Consider that Robustness Programming techniques exist which can compute $^{-}R(x) = \Pr\left\{ \mathbf{v} \in^{-} \mathbb{D}(c,^{-} r_d^*(x)) \right\}$ efficiently and can generate

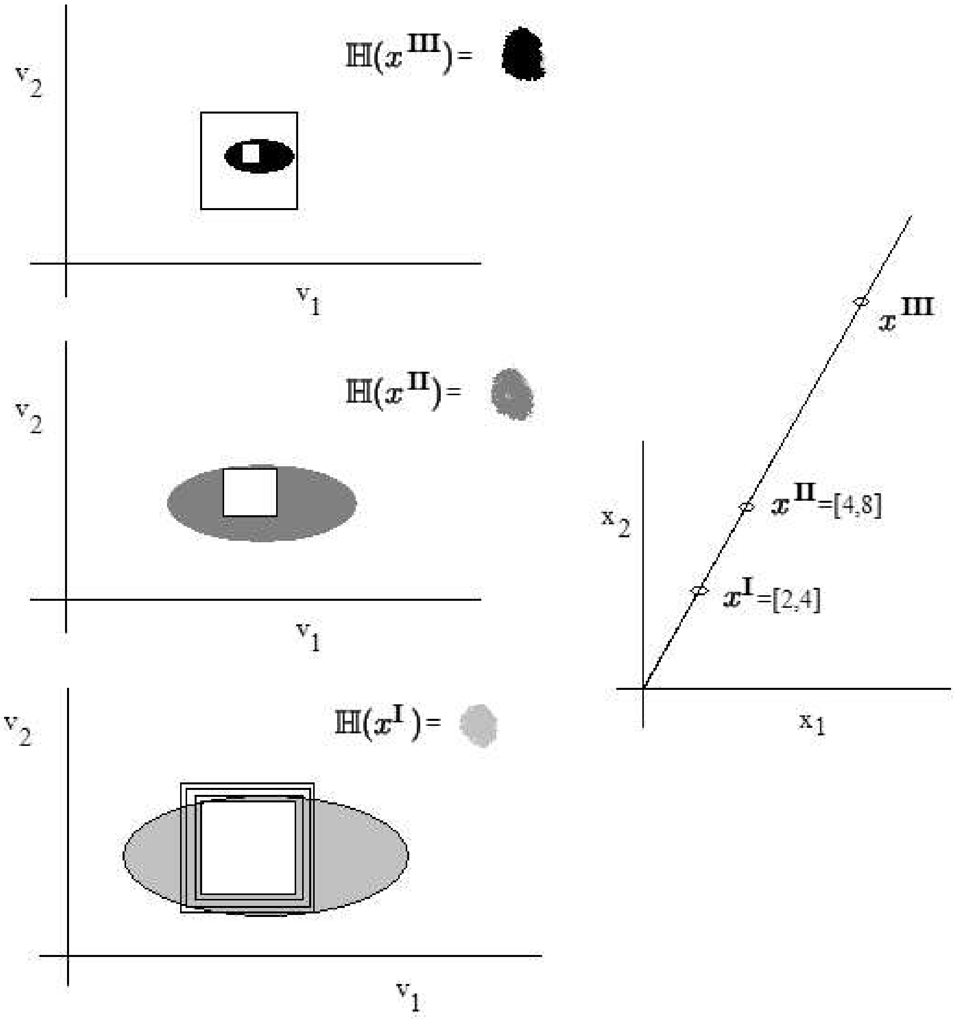


Figure 1. Illustration of $R(x)$-bounding by cube shape

conditional samples from $\mathbf{v}$, outside ${}^-\mathbb{D}(c, {}^- r_d^*(x))$. These conditional samples can be used to estimate $\Pr\left\{\mathbf{v} \in \mathbb{H}(x) | \mathbf{v} \notin^- \mathbb{D}(c, {}^- r_d^*(x))\right\}$, such that an estimate for

$$R(x) = {}^-R(x) + (1 - {}^-R(x)) \Pr\left\{\mathbf{v} \in \mathbb{H}(x) | \mathbf{v} \notin^- \mathbb{D}(c, {}^- r_d^*(x))\right\}$$

can be computed.

The standard error of the conditional MC estimate will be a factor $(1 - {}^-R(x))$ smaller, in comparison to the standard error of an ordinary MC estimate. As a concequence, in expectation, the conditional MC approach requires less samples. Less samples means shorter computation time. However, the conditional MC approach is more efficient, only in the situation that the shorter computation time in the sampling stage, compensates the extra computation time for finding the largest diamond.

2. Summary

Robustness of an object can be defined as the probability that uncertain factors have a favourable effect on the object. Favourable realisations of the uncertain factors can be collected in a set, which is called the Happy set.

Robustness bounds can be defined, which are proportional to the radius of shapes enclosed by the Happy set and shapes enclosing the happy set. Finding the radius of the largest enclosed and smallest enclosing shapes, are global optimisation problems, where it is essential to find global optima to guarantee a lower bound or upper bound respectively.

From an end-user perspective, the extreme values of robustness bounds, can be useful for decision support. From a technical perspective, robustness bounding methods are useful in the context of feasible domain reduction and efficient conditional sampling techniques.

References

[1] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization − methodology and applications. *Mathematical Programming series B*, 92:453–480, 2002.

[2] P. Bjerager. Probability integration by directional simulation. *ASCE Journal of Engineering Mechanics*, 8(114):1285–1302, 1988.

[3] X. Du and W. Chen. Towards a better understanding of modeling feasibility robustness in engineering design. *ASME Journal of Mechanical Design*, 122(4):385–394, 2000.

[4] Saeed Ghahramani. *Fundamentals of Probability Theory*. Prentice Hall, Upper Saddle River, New Jersey 07458, 2000.

[5] G.R. Grimmett and D.R. Stirzaker. *Probability and Random Processes 3ed*. Oxford University Press, Great Clarendon Street, Oxford ox2 6dp, 2001.

[6] H.M. Markowitz. Portfolio selection. *Jounal of Finance*, 7:77–91, 1952.

[7] Jinsuo Nie and Bruce R. Ellingwood. Directional methods for structural reliability analysis. *Structural Safety*, 22(3):233–249, 2000.

[8] G. Taguchi. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. Asian Productivity Organization, Tokyo, 1986.

On solving of bilinear programming problems*

Andrey V. Orlov

Institute of System Dynamics and Control Theory, Laboratory of Global Optimization Methods, Lermontov st. 134, Irkutsk, Russia, anor@icc.ru

Abstract In this paper we suggest a new approach for solving disjoint bilinear programming problems. This approach is based on the Global Search Strategy for d.c. maximization problems, developed by A.S.Strekalovsky. Two local search methods and outline of global search algorithm for such problems are presented.

Keywords: bilinear programming, d.c. functions, special methods of local search, bimatrix games, global search

1. Introduction

A number problems in engineernig design, decision theory, operation research and economy can be described by the bilinear programming problems. For example, finding the Nash equilibrium in bimatrix games [3, 4] and bilinear separation problem in $I\!R^n$ [5] can be formulated as specific bilinear problems.

In spite of external simplicity such problems are nonconvex. As is well known in nonconvex problems there exist a lot of local extremums or even stationary (critical) points, that are different from global solutions. And classical methods of convex optimization [1] are not applicable in such problems.

There are two types of bilinear programs: with joint constraints and with disjoint constraints. The former is more hard problem than the latter. But even for problems with disjoint constraints development of fast algorithm is complicated problem. Several methods have been proposed in literature to solve jisjoint bilinear problems [6]). The aim of this paper is to study the usefullness of Global Optimality Conditions approach, developed by A.S. Strekalovsky [2], for disjoint bilinear programs.

2. Problem statement and d.c. decomposition of goal function

Let us consider the bilinear function

$$F(x,y) = \langle c,x\rangle + \langle x,Qy\rangle + \langle d,y\rangle, \tag{1}$$

where $c,\ x \in I\!R^m;\ \ d,\ y \in I\!R^n$; Q is an $(m \times n)$ matrix.

The disjoint bilinear programming problem can be written as follows:

$$\left.\begin{array}{c} F(x,y) \uparrow \max\limits_{(x,y)}, \\ s.t.\ \ x \in X \overset{\triangle}{=} \{x \in I\!R^m \mid Ax \leq a,\ x \geq 0\}, \\ y \in Y \overset{\triangle}{=} \{y \in I\!R^n \mid By \leq b,\ y \geq 0\}, \end{array}\right\} \qquad (BLP)$$

*This work was supported by RFBR Grant No. 05-01-00110

where A is $(m_1 \times m)$ matrix, B is $(n_1 \times n)$ matrix, $a \in I\!R^{m_1}$, $\ b \in I\!R^{n_1}$. We will assume that X and Y are bounded polytopes.

It is readily seen that goal function of (BLP) can be represented as difference of two convex functions:

$$F(x,y) = f(x,y) - g(x,y), \tag{2}$$

where $f(x,y) = \dfrac{1}{4}\|x + Qy\|^2 + \langle c, x\rangle,\ \ g(x,y) = \dfrac{1}{4}\|x - Qy\|^2 - \langle d, y\rangle.$

Therefore Global Search Strategy for d.c. maximization problems may be using for solving (BLP) [2].

3. Local Search

A searching of critical (stationary) points in nonconvex problem under consideration is one of significant part of Global Search Strategy above mentioned. In this case definition of a critical point depends on properties of the problem and the corresponding local search method.

Further we presented two local search methods in (BLP).

Let (x_0, y_0) be a starting point.

X-procedure.

Step 0. Let $s := 0,\ \ x^s := x_0$.

Step 1. Find $\rho_s/2$-solution y^{s+1} of Linear Program

$$\langle d + x^sQ, y\rangle \uparrow \max_y,\ \ y \in Y, \qquad (LP_y).$$

Here

$$\langle d + x^sQ, y^{s+1}\rangle + \rho_s/2 \geq \sup_y\{\langle d + x^sQ, y\rangle \mid y \in Y\}. \tag{3}$$

Step 2. Find $\rho_s/2$-solution x^{s+1} of Linear Program

$$\langle c + Qy^{s+1}, x\rangle \uparrow \max_x,\ \ x \in X, \qquad (LP_x).$$

Here

$$\langle c + Qy^{s+1}, x^{s+1}\rangle + \rho_s/2 \geq \sup_x\{\langle c + Qy^{s+1}, x\rangle \mid x \in X\}. \tag{4}$$

Step 3. If

$$F(x^{s+1}, y^{s+1}) - F(x^s, y^{s+1}) \leq \tau, \tag{5}$$

where τ is accuracy of solution, then Stop, else let $s := s + 1$ and go to step 1.

The sense of this algorithm is to approximate solution of the linear programming problems in x and y. This method is modification of algorithm from [3].

Further we formulate the convergence theorem for this method.

Theorem 1. *Let $F(\cdot)$ be bounded above function on $X \times Y$ and suppose $\rho_s > 0,\ s = 1, 2, \ldots \sum\limits_{s=1}^{\infty} \rho_s < +\infty$. Then sequence of points (x^s, y^s) from X-procedure converge to $(\widehat{x}, \widehat{y})$:*

$$F(\widehat{x}, \widehat{y}) \geq F(\widehat{x}, y) \qquad \forall y \in Y, \tag{6}$$

$$F(\widehat{x}, \widehat{y}) \geq F(x, \widehat{y}) \qquad \forall x \in X. \tag{7}$$

The pair $(\widehat{x}, \widehat{y})$ is called the critical point of (BLP) if (6) and (7) takes place. This point is global solution of (BLP) in x and y separately.

When the algorithm stop (i.e. inequality (5) hold), we have approximate critical point of (BLP).

Further notice features of the algorithm. First, algorithm is needed only x_0 from (x_0, y_0) for starting. Second, this point may be not feasible. In spite of the last fact, the convergence of this method was proven.

Below we presented another "symmetric" method for local search in (BLP). This algorithm is needed only y_0 from (x_0, y_0) for starting.

Y-procedure.

Step 0. Let $s := 0, \;\; y^s := y_0$.

Step 1. Find $\rho_s/2$-solution x^{s+1} of Linear Program

$$\langle c + Qy^s, x\rangle \uparrow \max_x, \;\; x \in X, \qquad (LP_x).$$

Here

$$\langle c + Qy^s, x^{s+1}\rangle + \rho_s/2 \geq \sup_x\{\langle c + Qy^s, x\rangle \mid x \in X\}. \tag{8}$$

Step 2. Find $\rho_s/2$-solution x^{s+1} of Linear Program

$$\langle d + x^{s+1}Q, y\rangle \uparrow \max_y, \;\; y \in Y, \qquad (LP_y).$$

Here

$$\langle d + x^{s+1}Q, y^{s+1}\rangle + \rho_s/2 \geq \sup_y\{\langle d + x^{s+1}Q, y\rangle \mid y \in Y\}. \tag{9}$$

Step 3. If

$$F(x^{s+1}, y^{s+1}) - F(x^{s+1}, y^s) \leq \tau, \tag{10}$$

where τ is accuracy of solution, Then Stop, else let $s := s + 1$ and go to step 1.

Theorem 2. *In conditions of theorem 1 sequence (x^s, y^s) from Y-procedure converge to $(\widehat{x}, \widehat{y})$, such as (6) and (7) takes place.*

Notice, that both above methods converges to the point with identical properties.

4. Global Search Algorithm

Since we not obtain the global solution by local search, further the global search algorithm is presented.

Suppose $(x^0, y^0) \in D \overset{\triangle}{=} X \times Y$ is a starting point, $\{\tau_k\}, \;\; \{\delta_k\}$ are sequences of numbers, $\tau_k, \delta_k > 0, k = 0, 1, 2, ..., \;\;\; \tau_k \downarrow 0, \;\; \delta_k \downarrow 0, \;\; (k \to \infty)$, $Dir = \{(u^1, v^1), ..., (u^N, v^N) \in I\!R^{m+n} | (u^s, v^s) \neq 0, \;\; s = 1, ..., N\}$ is set of vectors, $\gamma_- \overset{\triangle}{=} \inf(g, D)$, $\gamma_+ \overset{\triangle}{=} \sup(g, D)$, ν and q are scalars.

Global search algorithm

Step 0. Let $k := 1, \;\; (\bar{x}^k, \bar{y}^k) := (x^0, y^0), \;\;\; s := 1, \;\;\; p := 1, \;\;\; \gamma := \gamma_-, \;\;\; \triangle\gamma = (\gamma_+ - \gamma_-)/q$.

Step 1. Beginning from $(\bar{x}^k, \bar{y}^k) \in D$ by X-procedure or Y-procedure obtain τ_k-critical point $(x^k, y^k) \in D$ in (P). Here $F(x^k, y^k) \geq F(\bar{x}^k, \bar{y}^k)$. Let $\zeta_k := F(x^k, y^k)$.

Step 2. Compute point $(\bar{u}^s, \bar{v}^s)$ of level surface approximation of $f(\cdot)$ by $(u^s, v^s) \in Dir$. Here $f(\bar{u}^s, \bar{v}^s) = \gamma + \zeta_k$.

Step 3. If $g(\bar{u}^s, \bar{v}^s) > \gamma + \nu\gamma$, then let $s := s + 1$ and go to step 2, else go to step 4.

Step 4. Beginning from $(\bar{u}^s, \bar{v}^s)$, by X-procedure or Y-procedure obtain δ_k-critical point $(\hat{x}^s, \hat{y}^s) \in D$ in (P).

Step 5. Find δ_k-solution (x_0^s, y_0^s), $f(x_0^s, y_0^s) = \gamma + \zeta_k$, of level problem

$$\begin{gathered}\langle \nabla_x f(x_0^s, y_0^s), \hat{x}^s - x_0^s \rangle + \langle \nabla_y f(x_0^s, y_0^s), \hat{y}^s - y_0^s \rangle + \delta_k \geq \\ \geq \sup_{x,y} \{ \langle \nabla_x f(x,y), \hat{x}^s - x \rangle + \langle \nabla_y f(x,y), \hat{y}^s - y \rangle : f(x,y) = \gamma + \zeta_k \}. \end{gathered} \tag{11}$$

Step 6. Compute

$$\eta_k(\gamma) = \gamma - g(\hat{x}^s, \hat{y}^s) + \langle \nabla_x f(x_0^s, y_0^s), \hat{x}^s - x_0^s \rangle + \langle \nabla_y f(x_0^s, y_0^s), \hat{y}^s - y_0^s \rangle. \tag{12}$$

Step 7. If $\eta_k(\gamma) \leq 0,\ s < N$, then let $s := s + 1$ and go to step 2.

Step 8. If $\eta_k(\gamma) \leq 0,\ s = N$, then let $\gamma := \gamma + \triangle\gamma$, $s := p$ and go to step 2.

Step 9. If $\eta_k(\gamma) > 0$, then let $(\bar{x}^{k+1}, \bar{y}^{k+1}) := (\hat{x}^s, \hat{y}^s)$, $k := k + 1$, $s := s + 1$, $p := s$ and go to step 1.

Step 10. If $s = N,\ \eta_k(\gamma) \leq 0,\ \ \forall \gamma \in [\gamma_-, \gamma_+]$ then Stop.

Let us explain some steps and parameters of above algorithm.

On step 2 we construct the level surface approximation by the set of vectors Dir. On step 3 we examine the point of level surface approximation on applicability by the Global Oplimality Conditions inequality (see [2]). Scalar ν is a first parameter of algorithm. By this parameter we can change accuracy of inequality on step 3. On step 4 we implement additional local search instead of linearized problem solving (see [2]).

On step 6 we compute the quality assessment of algorithm's iteration. On steps 7–10 we check on stopping criterion and perform returns on internal (on points of level surface approximation and on parts of segment $[\gamma_-, \gamma_+]$) and external (on critical points) cycles. Scalar q is a second parameter of algorithm. By this parameter we can change number of above mentioned segment parts.

The testing of the developed algorithm performed on the special disjoint bilinear program. This program is equivalent to finding Nash equilibrium point in bimatrix games [4].

Acknowledgments

The author wish to thank prof. A.S. Strekalovsky for their encouragement and support.

References

[1] Vasil'ev, F.P. (1988). *Numerical methods of extremal problems solving*. Nauka, Moscow (in russian).

[2] Strekalovsky, A.S. (2003). *Elements of Nonconvex Optimization*. Nauka, Novosibirsk (in russian).

[3] Mukhamediev, B.M. (1978). *The solution of bilinear problems and finding the equilibrium situations in bimatrix games*. Computational mathematics and Mathematical Physics, Vol.18, 60–66.

[4] Orlov, A.V., Strekalovsky, A.S. (2004). *Seeking the Equilibrium Situations in Bimatrix Games*. Automation and remote control, Vol.65, 204–217.

[5] Mangasaryan, O.L. (1993). *Bilinear Separation of Two Sets in n-Space*. Computational Optimization and Applications, No.2, 207–227.

[6] Floudas, C.A., Visweswaran, V. (1995). *Quadratic optimization*. In Handbook of Global Optimization/Ed. by Horst. R., Pardalos P. Dordrecht: Kluwer Academic Publishers, 224–228.

GASUB: A genetic-like algorithm for discrete location problems*

Blas Pelegrín[1], Pascual Fernández[1], Juani L. Redondo[2], Pilar M. Ortigosa[2], and I. García[2]

[1] *Dpt. Statistics and Operational Research, University of Murcia, Spain,* {pfdez,pelegrin}@um.es

[2] *Dpt. Computer Architecture and Electronics, University of Almería, Spain,* {inma, pilar, juani}@ace.ual.es

Abstract We propose a new genetic-like algorithm, *GASUB*, for finding solutions to different facility location problems which are hard to solve. We deal with the *p-median problem*, the *maximum capture* , and the *location-price problem with delivered prices*. In each one of them, a given number s of facility locations must be selected in a set of potential locations, containing m location points, so as to optimize a predetermined fitness function. Although these problems can be formulated as integer linear optimization problems, they can only be solved by standard optimizers for moderately small values of s and m. *GASUB* is compared with *MSH*, a multi-start substitution method widely used for location problems. Computational experiments with location-price problems (the ones with the more complex fitness function) show that *GASUB* obtains better solutions than *MSH*. Furthermore, the proposed algorithm finds global optima in all tested problems, which is shown by solving those problems by *Xpress-MP*, an integer linear programming optimizer [11].

Keywords: Combinatorial optimization, Discrete location problems, Stochastic algorithms.

1. Introduction

Location models have been developed to help decision makers in a variety of situations in which new facilities have to be located to serve a set of users which are aggregated in a given number of demand points (see [5]). These models can be classified in two big groups: Non-competitive models and Competitive models, depending on a single player in the market is considered or more than one. A detailed taxonomy can be found in the survey papers [3, 8].

In a non-competitive environment, the most important objective is *minimization of transportation costs*, which are due to the interaction between users and facilities, being the *p-MEDIAN* problem the basic model for many distribution systems (see [2]). In a competitive environment, firms compete for users and the usual objective is *profit maximization*. Since users are supposed to buy at the cheapest facility, strategic decisions on location and price have to be made. Firms normally use either a mill price or a delivered price policy. Under *mill pricing* (the seller sets a factory price, equal for all the customers in the market, and the buyer takes care of carriage). A basic model is *MAXCAP* (see [10]) in which a common mill price is set by all competing firms; there is a finite set of possible facility locations; and the objective is market share maximization (which is equivalent to profit maximization in this setting). Under *delivered pricing* (the seller charges a specific price in each market area, which includes the freight cost, and takes care of transport), there exist equilibrium prices that are determined by

*This research has been supported by the Ministry of Science and Technology of Spain under the research projects BEC2002-01026 and CICYT-TIC2002-00228, in part financed by the European Regional Development Fund (ERDF). We are very grateful with *Dashoptimization* for providing a*Xpress-MP* license for testing.

the facilities location (see [6]), thus the location-price problem becomes a location problem when firms charge equilibrium prices. To our knowledge, the resultant location problem only has been studied in discrete space and it will be referred here as the *MAXPROFIT* problem (see [4]).

The aim of this paper is to show that the above mentioned problems can be formulated in the same framework and solved by using the same type of algorithms. By using integer linear programming formulations of the problems, standard optimizers can be applied to find optimal solutions to moderated size problems. By determining its fitness functions, heuristics algorithms can be used to get good solutions for large size problems. We propose a new genetic algorithm with subpopulation support, *GASUB* , which is related to a previous algorithm given in [9]. The new algorithm is compared with the multi-start substitution heuristic, *MSH*, a procedure widely used for many combinatorial location problems (see [2]).

2. The location problems formulation

We will use the following notation when necessary:

$i, I = 1, 2, ..., n$	Index and collection of demand points
$j, J = 1, 2, ..., m$	Index and potential sites for facility location
$k, K = 1, 2, ..., q$	Index and pre-existing facility locations
w_i	Demand (or buying power) at point i
d_{ij}	Distance between demand point i and point j
$D_i = min\{d_{ik} : k \in K\}$	Distance from demand point i to the closest pre-existing facility
$N_i^< = \{j \in J : d_{ij} < D_i\}$	Collection of potential locations for servers that are closer to point i than the closest pre-existing facility
$N_i^= = \{j \in J : d_{ij} = D_i\}$	Collection of potential locations for servers that are at the same distance to point i than the closest pre-existing facility
$I^* = \{i \in I : N_i^< \cup N_i^= \neq \emptyset\}$	Collection of demand points that have at least one potential location for server closer or at the same distance than the pre-existing facilities
t	Unit transportation cost
s	Number of new facilities to be built

In the *p-MEDIAN* problem, there is no pre-existing facility in the market ($K = \emptyset$) and the objective is to minimize total transportation cost between demand points and their closest facilities. The following decision variables are defined:

$$\begin{aligned} y_j &= \begin{cases} 1 & \text{if a new facility is opened in } j \\ 0 & \text{otherwise} \end{cases} \\ x_{ij} &= \text{proportion of demand at } i \text{ served from site } j \end{aligned}$$

Then the problem is formulated as follows:

$$(P_1) \left\{ \begin{array}{ll} \min & \sum_{i \in I} \sum_{j \in J} t w_i d_{ij} x_{ij} \\ s.t. & \sum_{j \in J} x_{ij} = 1, \quad i \in I \\ & x_{ij} \leq y_j, \quad i \in I, j \in J \\ & \sum_{j \in J} y_j = s \\ & x_{ij} \geq 0 \\ & y_j \in \{0,1\} \end{array} \right.$$

In *MAXCAP*, there already exist some competing pre-existing facilities. Customer pays for transportation and buys at its closest facility. If a new facility and a pre-existing facility are the closest to a demand point i, then demand in i is divided so that a fixed proportion θ_i of customers buy at the new closest facilities, $0 \leq \theta_i \leq 1$. The following decision variables are defined:

$$
\begin{aligned}
y_j &= \begin{cases} 1 & \text{if a new facility is opened in } j \\ 0 & \text{otherwise} \end{cases} \\
x_i &= \begin{cases} 1 & \text{if the new facilities capture demand point } i \\ 0 & \text{otherwise} \end{cases} \\
z_i &= \begin{cases} 1 & \text{if demand point } i \text{ is divided} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

then the problem is formulated as follows:

$$
(P_2)\left\{
\begin{array}{ll}
\max & \sum_{i\in I} w_i x_i + \sum_{i\in I} \theta_i w_i z_i \\
s.t. & x_i \leq \sum_{j\in N_i^<} y_j, \quad i \in I^* \\
 & z_i \leq \sum_{j\in N_i^=} y_j, \quad i \in I^* \\
 & x_i + z_i \leq 1, \quad i \in I^* \\
 & \sum_{j\in J} y_j = s \\
 & x_i \geq 0, z_i \geq 0 \\
 & y_j \in \{0,1\}
\end{array}
\right.
$$

In *MAXPROFIT*, facilities take charge of transportation and deliver the product to customers. Each facility offers a specific price at each demand point. As result of price competition, the optimal price a new facility can offer at demand point i is the equilibrium price, which is given by $p_{min} + tD_i$, where p_{min} is the minimum selling price at the firm's door. With equilibrium prices, only the closest facility to a demand point i can offer the lowest price in i. Then, the same rule as in *MAXCAP* is used for tie breaking when two or more facilities are the closest to a demand point. Let $p_{net} = p_{min} - p_{prod}$, where p_{prod} is the production cost, which is supposed not to depend on site location. The following decision variables are defined:

$$
\begin{aligned}
y_j &= \begin{cases} 1 & \text{if a new facility is opened in } j \\ 0 & \text{otherwise} \end{cases} \\
x_{ij} &= \text{proportion of demand at } i \text{ served from site } j \\
z_i &= \begin{cases} 1 & \text{if demand point } i \text{ is divided} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

then the problem is formulated as follows:

$$
(P_3)\left\{
\begin{array}{ll}
\max & \sum_{i\in I^*} \sum_{j\in N_i^<} [p_{net} + t(D_i - d_{ij})] w_i x_{ij} + \sum_{i\in I^*} p_{net} \theta_i w_i z_i \\
s.a. & \sum_{j\in N_i^<} x_{ij} + z_i \leq 1, \quad i \in I^* \\
 & x_{ij} \leq y_j, \quad i \in I^*, j \in N_i^< \\
 & z_i \leq \sum_{j\in N_i^=} y_j, \quad i \in I^* \\
 & \sum_{j\in J} y_j = s \\
 & x_{ij} \geq 0, z_i \geq 0 \\
 & y_j \in \{0,1\}
\end{array}
\right.
$$

Problems (P_1), (P_2) and (P_3) can be exactly solved by standard integer linear programming optimizers only for data of moderated size, which is due to the optimizer can not manage the corresponding matrices when the cardinalities of I and/or J are very large. In fact, (P_1) and (P_2) have been proved to be NP-Hard (see [1, 7]) and (P_3) is also NP-Hard since it becomes (P_2) by taking $p_{net} = 1, t = 0$ and $x_i = \sum_{j \in N_i^<} x_{ij}$.

3. The GASUB algorithm

In this section the basic concepts, the algorithm, and the setting of the parameters are outlined. First we will determine the fitness function for the above problems (notice that (P_1) will be considered as a maximization problem).

For each problem, once the y-variables are fixed (which means that s elements in the set J are selected), the optimal values of non y-variables can easily be determined. Let $S \subset J$ be any subset of s elements, we denote with $d_i(S)$ the distance from demand point i to the closest location in S, $d_i(S) = min\{d_{ij} : j \in S\}$. If we consider the sets $I_S^1 = \{i : d_i(S) < D_i\}$ and $I_S^2 = \{i : d_i(S) = D_i\}$, then the fitness functions are given as follows:

$$\text{\textit{p-MEDIAN}:} \quad \Pi_1(S) = -\sum_{i \in I} w_i d_i(S)$$

$$\text{\textit{MAXCAP}:} \quad \Pi_2(S) = \sum_{i \in I_S^1} w_i + \sum_{i \in I_S^2} \theta_i w_i$$

$$\text{\textit{MAXPROFIT}:} \quad \Pi_3(S) = p_{net}(\sum_{i \in I_S^1} w_i + \sum_{i \in I_S^2} \theta_i w_i) + t \sum_{i \in I_S^1} (D_i - d_i(S)) w_i$$

3.1 Problem encoding

A point (individual in terms of genetic algorithms) consists of a single string that is a collection of m bits. The position of a bit in the string coincides with the index of the associated facility. Because of the set S of selected facilities is predetermined for every problem, the number of bits to 1 value must be fixed to the number of new facilities to be selected (cardinal of S). We must consider this constraint when generating any search point.

3.2 Basic concepts

A key notion in GASUB is that of a **subpopulation**. A subpopulation would be equivalent to a single individual, which is defined by a center, a fitness function and a radius value. The center is a solution and the radius indicates the attraction area of this subpopulation.

Of course, this definition assumes a *distance* defined over the search space. For our combinatorial problem we define the Hamming distance. Because of the constraint of the problem, where the number of chosen facilities and hence the number of bits (genes) to 1 is fixed, the Hamming distance between any two feasible points (individuals) must be always multiple of 2.

The radius of a subpopulation is not arbitrary; it is taken from a list of decreasing radii that follows a **cooling schedule**. The first element of this list is the diameter of the search space. If the radius of a subpopulation is the *i*th element of the list, then the level of the subpopulation is said to be *i*. Given the largest radius and the smallest one (r_1 and r_{levels}, respectively) the radii in the list are expressed by the exponential function

$$r_i = 2r_1 \left(\frac{r_{levels}}{r_1} \right)^{\frac{i-1}{levels-1}} \qquad (i = 2, \ldots, levels). \tag{1}$$

The parameter $levels$ indicates the maximal number of levels in the algorithm, i.e. the number of different 'cooling' stages. Every level i (i.e. for levels from $[1, levels]$) has a radius value (r_i) and two maxima on the number of function evaluations (f.e.) namely new_i (maximum f.e. allowed when creating new subpopulations) and n_i (maximum f.e. allowed when mutating individuals).

During the optimization process, a list of subpopulations is kept by GASUB and this $subp - list$ defines the whole **population**.

3.3 Input parameters

In GASUB the most important parameters are those defined at each level: the radii (r_i) and the numbers of function evaluations for subpopulations creation (new_i) and optimization (n_i). These parameters are computed from some user-given parameters that are easier to understand:

$evals$: The maximal number of function evaluations the user allows for the whole optimization process.It could be called as *Whole Budget*. Note that the actual number of function evaluations may be less than this value.

$levels$: The maximum number of levels, i.e. the number of cooling stages.

max_subp_num: The maximum length of the $subp - list$.

r_{levels}: The radius associated with the maximum level, i.e. $levels$.

3.4 The Algorithm

The GASUB algorithm has the following structure:

```
Begin GASUB
    Initializing population
    Mutation($n_1$)
    for $i = 1$ to $levels$
        Determine $r_i$, $new_i$, $n_i$
        Generation and crossover($new_i$/length($subp\_list$))
        Selection($r_i$, $max\_subp\_num$)
        Mutation($n_i / max\_subp\_num$)
        Selection($r_i$, $max\_subp\_num$)
    end for
End GASUB
```

In the following, the different key stages in the algorithm are described:

Initializing population: A new subpopulations list consisting of one subpopulation with a random center at level one is created. The center must have as many genes to 1 value as the number of new facilities are being chosen.

Generation and crossover: For every individual in the $subp - list$, new random individuals are generated, and for every pair of new individuals the objective function is evaluated at the middle of the *section* connecting the pair (see Figure 1). All generated individuals must satisfy the constraint of having a fixed number of genes to 1 value. If the fitness value of the middle point is better than the fitness value of the center of the subpopulation, then this individual will be the new center, keeping the same level value. If the value in the middle point is worse than the values of the pair, then the members of the pair are inserted in the $subp - list$. Every newly inserted subpopulation is assigned the actual level value (i).

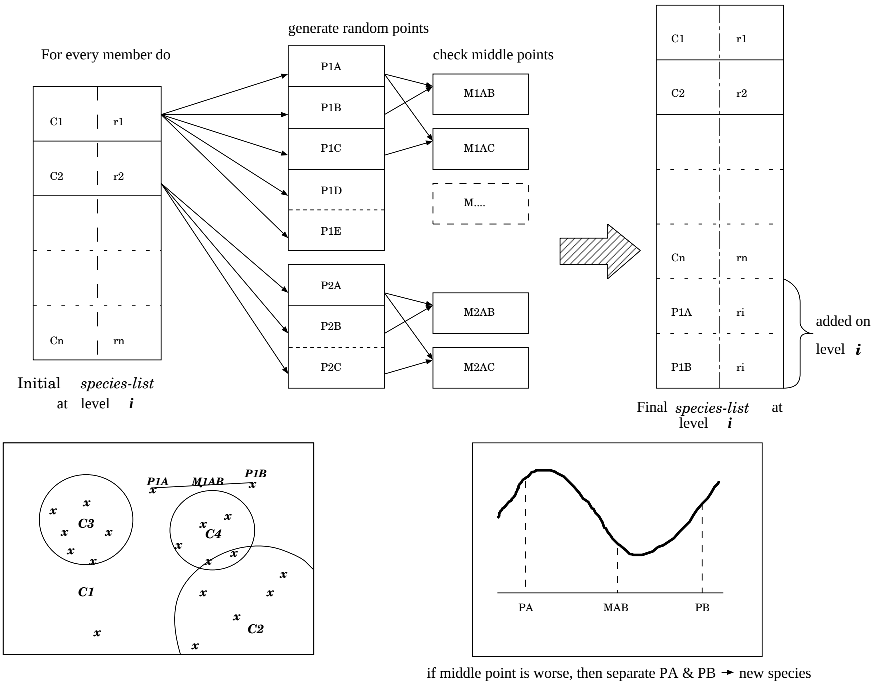


Figure 1. Generation and crossover

As a result of this procedure the $subp - list$ will eventually contain several subpopulations with different levels (hence different radii). The motivation behind this method is to create subpopulations that are on different 'hills' so ensuring that there is a valley between the new species.

Selection: This procedure has two mechanisms, the first tries to fuse subpopulations that are too close and the second selects the subpopulations that will be maintained in the $subp - list$.

Fuse_species: If the centers of any pair of subpopulations from the $subp - list$ are closer to each other than the given radius, the two subpopulations are fused. The center of the new subpopulation will be the one with the better function value while the level will be the minimum of the levels of the fused subpopulations (so the radius will be the largest one).

Select_species: It deletes subpopulations to reduce the list length to the given value. Higher level subpopulations are deleted first; therefore subpopulations with larger radii are always kept. For this reason one subpopulation at level 1 whose radius is equal to the diameter of the search domain always exists, making it possible to escape from local optima.

Mutation: It applies consecutive mutations to every center of every subpopulation. A mutation means an interchange of one facility, so one gene of the center is changed from 0 to 1 and other gene is changed from 1 to 0, according to our problem encoding.

If when mutating an individual, its function value is better than the value of the center, then this new individual will replace the center. At level i, the maximum number of consecutive mutations applied to each center is n_i/max_subp_num.

Note that the fact that GASUB may terminate simply because it has executed all its levels. The final number of function evaluations thus depends on the complexity of the objective function. This behavior is qualitatively different from genetic algorithms which typically run until a maximum number of function evaluations.

4. Computational experiments

For computational experiments we selected 1046 cities in Spain as nodes, and their populations as demand. The geographic coordinates and population of each city was obtained from *http://www.terra.es/personal/GPS.2000* and *http://www.ine.es*, respectively. Distances were taken as the Euclidean distances between cities and each city demand was taken proportional to its population. All cities were chosen as location candidates (the set J) in all test problems.

In order to have an overall view on the performance of *MSH* and *GASUB*, we fixed $p_{net} = 1$ and generated different types of problems, varying the number of pre-existing facilities ($q = 2, 4, 6, 8, 10$), the number of new facilities ($s = 2, 4, 6, 8, 10$) and the unit transportation cost ($t = 0.001, 0.005, 0.01, 0.05, 0.1$). These 125 problems were optimally solved by using *Xpress-MP* and their optimal values were used to evaluate the performance of both heuristics. All the computational results have been obtained under Linux on a Pentium IV with 3GHz CPU and 2GB memory. The algorithms were implemented in C++.

Results show that the parameter t practically doesn't affect to the computational time, in such a way that given fixed values for q and s, the required times to solve the problem are similar for different values of t. For this reason, graphics in Figure 2 showing the computational times are average values from the computational times obtained for the different t values ($0.001, 0.005, 0.01, 0.05, 0.1$). Anyway it is interesting to remark that *MSH* do not get optimal locations for some small values of t and high values of s. On the other hand, *GASUB* not only finds the global solution for all runs of every problem but also finds more than one existing global optima. For instance, for $q = 4$ two global optima were found by *GASUB*.

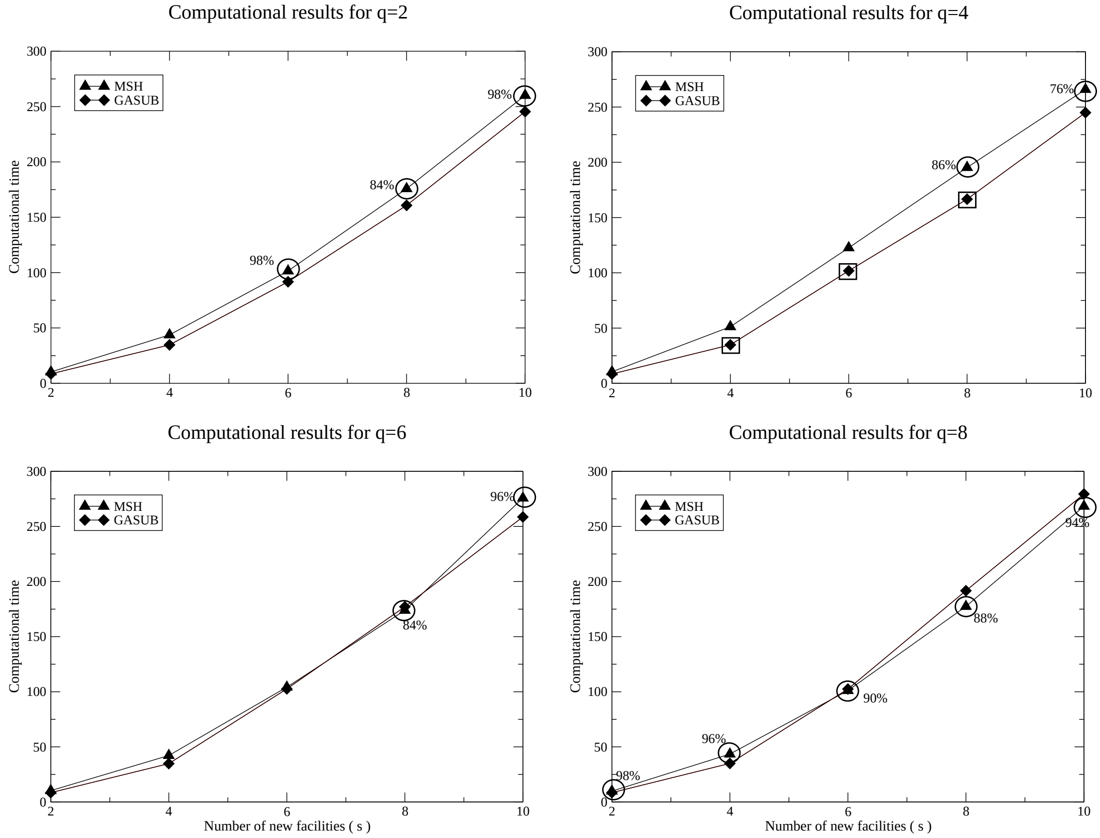


Figure 2. Computational results for q=2, 4, 6, 8

Figure 2 shows four comparative graphics with computational results of the problems for different values of the number of pre-existing facilities ($q = 2, 4, 6, 8$). Each graphic shows the average value of computational time (varying t) required for *MSH* and *GASUB*, for the five values of s $(2, 4, 6, 8, 10)$. For cases where *MSH* doesn't find the solution at all runs, we have drawn a circle and the corresponding percentage of success on finding the global solution (below 100%). Note that these are average values for different values of t. For those cases where *GASUB* finds the two optima we have drawn a square.

5. Conclusions and future research

We have shown some basic discrete location problems which can be solved, exactly and approximately, by the same type of algorithms. Then we have proposed a genetic-like heuristic, *GASUB*, to deal with large instances of these problems. This algorithm has been compared with an optimizer, *Xpress-MP*, and a well known heuristic used for many location problems, *MSH*. For this, 125 instances of *MAXPROFIT* have been solved by the three algorithms. Computational experiments show that optimal locations can also be obtained by the heuristics. *MSH* obtained optimal locations in 20% of the solved problems, while *GASUB* always found optimal locations in the tested problems. An additional advantage of *GASUB* is that this algorithm can generate more than one global optimum (in 8 out of the 125 solved problems), the contrary happens with *MSH* and *Xpress-MP* which obtain only one optimum. Computational times for both heuristics increase significantly when the s value increases, which is explained by the fact that the number of location combination exponentially increases when the parameter s increases.

As future research we will conduct more experiments with instances of the three mentioned models and we will compare *GASUB* with another genetic algorithm. Furthermore, a parallelization of *GASUB* will be investigated in order to reduce computational times.

References

[1] G. Cornuejols, M.L. Fisher, and G.L. Nemhauser. Locations of bank accounts to optimize float: An analytical study of exact and approximate algorithms. *Management Science*, 23:789–810, 1997.

[2] M.S. Daskin. *Network and Discrete location: Models, Algorithms and Applications*. Dis Wiley, 1995.

[3] H.A. Eiselt, G. Laporte, and J.F. Thisse. Competitive location models: a framework and bibliography. *Transportation Science*, 27:44–54, 1993.

[4] P. Fernández, B. Pelegrín, M.D. García, and P. Peeters. A discrete long-term location-price problem under the assumption of discriminatory pricing: Formulations and parametric analysis. *European Journal of Operations Research*, pages –, 2005. forthcoming.

[5] R.L. Francis, T.J. Lowe, and A. Tamir. Demand point aggregation for location models. In Z. Drezner and H. Hamacher, editors, *Facility Location: Application and Theory*, pages 207–232. Springer, 2002.

[6] M.D. García, P. Fernández, and B. Pelegrín. On price competition in location-price models with spatially separated markets. *TOP*, 12(2):351–374, 2004.

[7] S.L. Hakimi. On locating new facilities in a competitive environment. *European Journal of Operations Research*, 12:29–35, 1983.

[8] H.W. Hamacher and S. Nickel. Classification of location models. *Location Science*, 13:229–242, 1998.

[9] P.M. Ortigosa, I. García, and M. Jelasity. Reliability and performance of UEGO, a clustering-based global optimizer. *Journal of Global Optimization*, 19(3):265–289, 2001.

[10] D. Serra and C. ReVelle. Competitive location in discrete space. In Z. Drezner, editor, *Facility Location: A Survey of Applications and Methods*, pages 367–386. Springer, 1995.

[11] Xpress-MP. *Dashoptimization*, 2004.

Dynamic Stochastic Optimal Path

Deolinda M. L. D. Rasteiro[1] and António J. B. Anjo[2]

[1] *Fisics/Mathemathics Department, Coimbra Engineering Superior Institute, Coimbra, Portugal,* dml@isec.pt

[2] *Mathematics Department, Aveiro University, Aveiro, Portugal,* batel@ua.pt

Abstract In this paper we propose an algorithm which determines the path that minimizes the expected value of an utility function over a dynamic probabilistic network with discrete real random variables (parameters) associated to each emerging arc. To obtain the optimal dynamic path from a source to sink node, we use a generalization of Bellman first-in-first-out labelling-correcting algorithm used to the determine the shortest path in directed networks with deterministic parameters associated to each arc. Additionally some initialization techniques that improve the running times without jeopardizing memory are also considered.

The topology of the networks is not known in advance, which means that we only have knowledge of the incoming (outgoing) arcs, and their parameters, of some specific node once we reach it. Thus the optimal path is determined in a dynamic way.

We also present computational results for networks with 100 up to 10000 nodes and densities 2, 5 and 10.

Keywords: Probability Networks, Expected Value of an Optimal Path, Dynamic paths, Utility Functions

1. Introduction

Suppose we want to obtain de optimal path in a directed random network, where the parameters associated to the arcs are real random variables following discrete distributions. The criteria that has been chosen to decide which path is optimal is the one that minimizes the expected value of an utility function over the considered network.

This methodology can be used in different applications, as energy network or data network, where real on time optimal solutions are necessary.

Different types of optimal path problems over random probabilistic networks are considered in the literature. Considering networks with random parameters that are all realized at once, the first publication belongs to Frank, [6] who determined the shortest path on a random graph and presented a process for obtaining the probability distribution of the shortest path.

In 1991 Bard and Bennett, [2] developed heuristic methods based on Monte-Carlo simulations for the stochastic optimal path with non-increasing utility function. Their computational results regarded networks with 20 up to 60 nodes.

For the special case of acyclic networks Cheung and Muralidharan [4] developed a polynomial time algorithm (in terms of the number of realizations per arc cost and the number of emerging arcs per node) to compute the expected cost of the dynamic stochastic shortest path.

In 2004 Rasteiro and Anjo, [11] developed an algorithm to solve the stochastic optimal path considering random continuous parameters associated to the network arcs.

In this paper we propose an algorithm which determines the path that minimizes the expected value of an utility function over a dynamic probabilistic network with discrete real random variables (parameters) associated to each emerging arc. Our algorithm is a generalization of Bellman first-in-first-out labelling-correcting algorithm used to the determine the shortest path in directed networks with deterministic parameters associated to each arc.

The most common methods used to solve the classical deterministic shortest path problem are the label-correcting methods. In this approach to each node $i \in \mathcal{N}$, ($(\mathcal{N}, \mathcal{A})$ is a probabilistic network where $\mathcal{N} = \{v_1, \ldots, v_n\}$ is the set of nodes and $\mathcal{A} = \{a_1, \ldots, a_m\} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of arcs) is assigned a label (usually the distance or cost to the destination node) and put into a queue. Then we scan through the nodes in the queue and update the labels, if necessary. After a node is scanned it will be removed from the queue, and then some nodes may be inserted or repositioned in the queue. The process is repeated until the queue is empty or all labels are correct. There are several implementations of this methods including the first-in-first-out algorithm of Bellman [3]. A review of these methods, which basic difference is the way how the queue is manipulated, can be found in Ahuja et al [1].

We present computational results, for networks with 100 up to 10000 nodes and densities 2, 5 and 10, which prove that our approach is very efficient in terms of memory and time.

2. Problem Definition

In the stochastic shortest path problem a directed probabilistic network $(\mathcal{N}, \mathcal{A})$ is given where each arc $(i, j) \in \mathcal{A}$ is associated to the real random variable X_{ij} which is called the random parameter of the arc $(i, j) \in \mathcal{A}$. We assume that the real random variables X_{ij} have discrete distributions and are independent. The variables X_{ij} are sometimes referred as cost, time or distance.

The set of outcomes of X_{ij} will be denoted by $S_{X_{ij}} = \left\{d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^r\right\}$. We will assume that the dimension of $S_{X_{ij}}$, i.e, r is always a finite value. The probability of X_{ij} assume the value d_{ij}^l is denoted by p_{ij}^l.

If an appropriate utility measure is assigned to each possible consequence and the expected value of the utility measure of each alternative is calculated, then the best action is to consider the alternative with the highest expected utility (which can be the smallest expected value). Different axioms that imply the existence of utilities with the property that expected utility is an appropriate guide to consistent decision making are presented in [5, 7–10]. The choice of an adequate utility function for a specific type of problem can be taken using direct methods presented in Keeney and Raiffa's book.

The utility of the arc $(i, j) \in \mathcal{A}$ in the optimal path is measured calculating the minimum of the real random variables X_{iw} where w is such that the arc $(i, w) \in \mathcal{A}$, i.e, w belongs to the forward star[1] of i. Thus $\mathcal{U}((i, j)) = \min\limits_{w \in \mathcal{F}(i)} X_{iw}$.

Associated to the path p, we define the real random variable

$$X_p = \sum_{(i,j) \in p} \min_{w \in \mathcal{F}(i)} X_{iw}$$

representing the random parameter of the loopless path $p \in \mathcal{P}$.

With the objective of determine the optimal path, we consider a real function $\mathcal{U} : \mathcal{P} \longrightarrow \mathrm{I\!R}$, called *utility function*, such that for each loopless path p, $\mathcal{U}(p)$ depends on the random variables associated to the arcs of p and is defined as

$$\mathcal{U}(p) = \mathrm{E}\left(\sum_{(i,j) \in p} \min_{w \in \mathcal{F}(i)} X_{iw}\right).$$

[1] The forward star of node i is the set formed by the terminal nodes of its outgoing arcs.

In the dynamic stochastic shortest path, we want to determine the loopless path $p^\star \in \mathcal{P}$ that minimizes the expected value of the utility function. The loopless path $p^\star$ is called *optimal solution* of the referred problem.

The problem can then be mathematically defined as

$$\begin{aligned}
\min_{p\in\mathcal{P}} U(p) &= \min_{p\in\mathcal{P}} \ \mathrm{E}\left(\sum_{(i,j)\in\mathcal{A}} \min_{w\in\mathcal{F}(i)} X_{iw}\right) \\
&= \min \ \mathrm{E}\left(\sum_{(i,j)\in\mathcal{A}} \min_{w\in\mathcal{F}(i)} X_{iw}\right) Y_{ij} \qquad (1)\\
&\quad s.t \\
&\quad \sum_{(i,j)\in\mathcal{A}} Y_{ij} - Y_{ji} = \begin{cases} 1 & , \quad i = s \\ 0 & , \quad i \notin \{s,t\} \\ -1 & , \quad i = t \end{cases} \\
&\quad Y_{ij} \in \{0,1\}
\end{aligned}$$

Since the constraint matrix is totally unimodular, by the integrality property, the solution of the previous problem is equal to the solution of its linear relaxation.

In order to exemplify the problem consider the following network

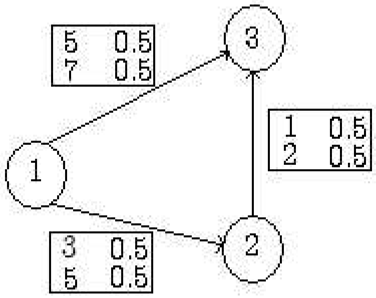


Figure 1. Network Example

The solution for the referred example is

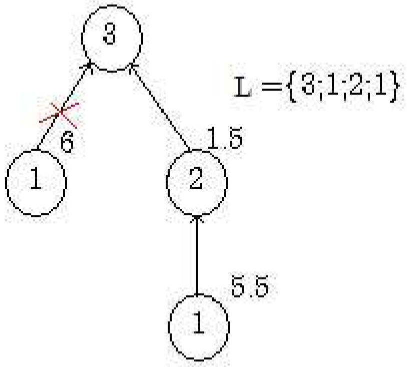


Figure 2. Solution

which means that the dynamic stochastic optimal loopless path is the following

$$< 1, (1,2), 2, (2,3), 3 >$$

with value 5.5.

3. An algorithm

In order to build the algorithm consider the variables $l_1, \ldots, l_n$ associated to the linear relaxation of problem (1) constraints. Its dual is

$$\begin{array}{ll} \max & l_1 - l_n \\ s.t & \\ & l_i - l_w \leq \mathrm{E}\left(\min_{w \in \mathcal{F}(i)} X_{iw}\right) \\ & \\ & l \lesseqqgtr 0 \end{array} \tag{2}$$

and thus the condition to obtain a dual feasible (which is the generalization of Bellman's equation) solution will be the main step of our algorithm.

We will denote by $l[i]$ the actual label of node i, i.e, the actual minimum expected value of the utility function from node i to the terminal node; L is the set of nodes already labelled which will work as a first-in-first-out list; $\mathcal{B}(j)$ is the set of predecessor nodes of node j; $\mathcal{F}(i)$ is the set of successor nodes of node i.

The solution to problem (2) is obtained using the following algorithm

```
Algorithm 1 {
    Initialize $l$[terminal] $= 0$;
    Initialize all label nodes (except the terminal) to "infinity";
    Initialize the set $L$ with the terminal node;
    While $L \neq \emptyset$ do
    {
      Remove node $j$ from $L$;
      For each $i \in B[j]$ do
      {
        If $(l[i] > \mathrm{E}(\min_{w \in \mathcal{F}[i]}\{X_{iw} + l[w]\}))$
        {  $l[i] = \mathrm{E}\left(\min_{w \in \mathcal{F}[i]}\{X_{iw} + l[w]\}\right)$};
          If $(i \notin L)$
            $\{L = L \cup \{i\}\}$
      }
    }
    }
```

When applying the above algorithm to the network example we observe that node 1 was first labelled with 6 and inserted in L to be removed in order to label other nodes. Fortunately, in this case, there were no predecessors of node 1 otherwise we would have had unnecessary work since its label was in a future iteration improved! Thus improvements must be done! One that is clear and solves this type of "wrong" labelling can be done by using the following property.

Theorem 1. *Let X and Y be two random variables with finite mean value. Then the following inequality holds $E(\min(X,Y)) \leq \min(E(X), E(Y))$.*

The improved algorithm differs from the previous one only in the first step, since instead of initializing the labels to infinity we initialize them to the labels obtained when running an algorithm to obtain the shortest deterministic path from t to s, considering the means of the random variables associated to the arcs. The improved algorithm is

```
Algorithm 2 {
    Initialize $l$[terminal] = 0;
    If (network is acyclic)
      {Initialize all label nodes (except the terminal) using the
      shortest path considering the mean of the arcs parameters};
    else{network is cyclic}
      {Initialize all label nodes (except the terminal) using the
      maximum of the labels obtained when determining the shortest
      path considering the mean of the arcs parameters};
    Apply Algorithm 1 }
```

The result of applying the improved algorithm to the example network presented above is

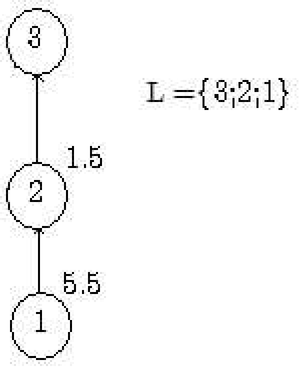


Figure 3. Solution with improvements

4. Computational Results and Conclusions

The computational effort of the proposed algorithms is highest in the calculation of the minimum of real random variables since we have to compare the set of outcomes for each random variable and then compute the resulting probability mass function. The time consumed on this step of the algorithm is highly dependent of the density of the network and of the dimension of the set of outcomes of each random variable.

The algorithm was implemented in C++. The experiments were done on a Pentium IV 2.8 GHz with 256 Mbyte memory.

For each dimension 30 network topologies were generated. For each topology 30 arc probability mass functions were generated. The number of outcomes of each arc random variable is 5.

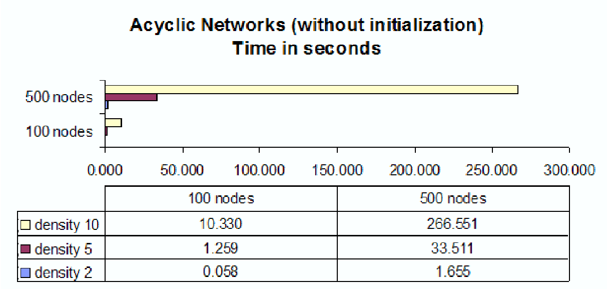
The execution times presented, in seconds, are the mean of the 30 times obtained for each topology.

In this paper we presented two algorithms, for acyclic and cyclic directed networks and also two variants - one without initialization and other with initialization.

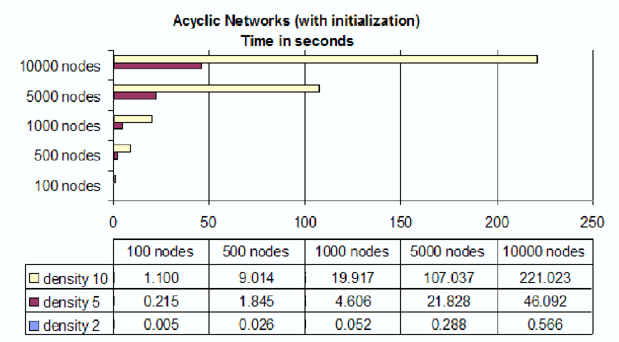
As it was referred the computational effort is highest when we want to determine the minimum of the random variables, and this is one of the features that has to be considered in future theoretical work.

For the networks examples generated the obtained results are strongly encouraging either in terms of time consuming or memory occupation.

To summarize the developed algorithms are very efficient to the proposed problems and we think that they will also perform very well for real problems that can be modulated to fit on our problem definition.



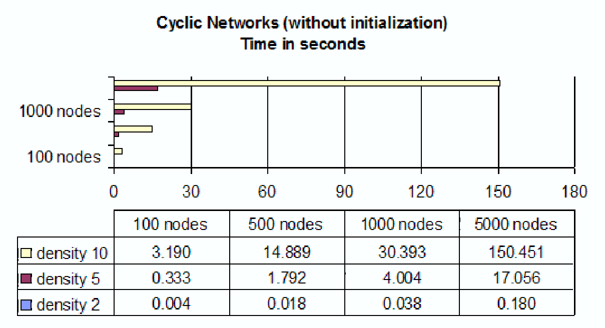
	100 nodes	500 nodes
□ density 10	10.330	266.551
■ density 5	1.259	33.511
■ density 2	0.058	1.655



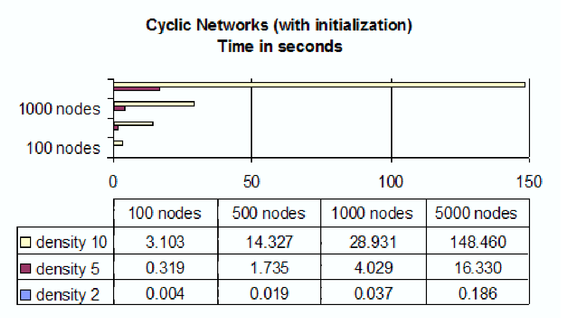
	100 nodes	500 nodes	1000 nodes	5000 nodes	10000 nodes
□ density 10	1.100	9.014	19.917	107.037	221.023
■ density 5	0.215	1.845	4.606	21.828	46.092
■ density 2	0.005	0.026	0.052	0.288	0.566

Figure 4. Running times (seconds) for acyclic networks without initialization.

Figure 5. Running times (seconds) for acyclic networks with initialization.



	100 nodes	500 nodes	1000 nodes	5000 nodes
□ density 10	3.190	14.889	30.393	150.451
■ density 5	0.333	1.792	4.004	17.056
■ density 2	0.004	0.018	0.038	0.180



	100 nodes	500 nodes	1000 nodes	5000 nodes
□ density 10	3.103	14.327	28.931	148.460
■ density 5	0.319	1.735	4.029	16.330
■ density 2	0.004	0.019	0.037	0.186

Figure 6. Running times (seconds) for cyclic networks without initialization.

Figure 7. Running times (seconds) for cyclic networks with initialization.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows theory, algorithms, and applications*, Uper Saddle River, Prentice-Hall, 1993.

[2] J. F. Bard and J. E. Bennett, *Arc reduction and path preference in stochastic acyclic networks*, Management Science **31/7** (1991), 198–215.

[3] R. Bellman, *On a routing problem*, Quarterly of Applied Mathematics **16** (1958), 88–90.

[4] R. K. Cheung and B. Muralidharan, *Dynamic routing of priority shipments on a less-thantruckload service network*, Tech. report, Department of Industrial and Manufacturing Systems Engineering, Iowa State University, 1995.

[5] P. C Fishburn, *Utility theory for decision making*, Wiley, New York, 1970.

[6] H. Frank, *Shortest paths in probabilistic graphs*, Operations Research **17** (1969), 583–599.

[7] R. D. Luce and H. Raiffa, *Games and decisions*, Wiley, New York, 1957.

[8] J. W. Pratt, H. Raiffa, and R. O. Schlaifer, *Introduction to statistical decision theory*, McGraw-Hill, New York, 1965.

[9] L. J. Savage, *The foundations of statistics*, Wiley, New York, 1954.

[10] J. von Neumann and O. Mogenstern, *Theory of games and economic behavior*, Princeton University Press, Princeton, N.J., 1947.

[11] Rasteiro, D.M.L.D. and Anjo, A.J.B., *Optimal Paths in Probabilistic Networks* , Journal of Mathematical Sciences **120**/1 (2004), 974–987.

A Multi-Objective Evolution Strategy for the Optimization of Nonlinear Dynamic Systems

José-Oscar H. Sendín, Antonio A. Alonso, and Julio R. Banga

Process Engineering Group (IIM-CSIC), Vigo, Spain.
osendin@iim.csic.es; antonio@iim.csic.es; julio@iim.csic.es

Abstract In this contribution, we consider the multicriteria optimization of nonlinear dynamic systems in the field of biochemical process engineering. For these problems, computing the set of optimal solutions can be a challenging task due to their highly constrained and non-linear nature. Thus, global optimization methods are needed to find suitable solutions.

We present a new multi-objective evolution strategy for dealing with this class of problems. The proposed approach combines a well-known multicriteria optimization technique with a recent stochastic global optimization method. The usefulness and efficiency of this novel approach are illustrated by considering the integrated design and control of a wastewater treatment plant.

Keywords: Multiobjective optimization, evolution strategy, bioprocess engineering, nonlinear dynamic systems.

1. Introduction

For most real world applications, the optimization problem involves multiple performance criteria (often conflicting) which must be optimized simultaneously. In general, there does not exist a single solution which is simultaneously optimal for all the objectives. Instead, the solution of a multi-objective optimization problem is a set of optimal trade-offs between the different criteria, the so-called Pareto-optimal set (or Pareto front). All points in this set are optimal in the sense that an improvement in one objective can only be achieved by degrading one or more of the others. In the absence of any further information, no solution can be considered better than another and, ideally, the entire Pareto-optimal set should be found.

In this work, we consider the multicriteria optimization of nonlinear dynamic systems in the field of biochemical process engineering. Particularly, the integrated design and control of bioprocesses is formulated as a nonlinear multi-objective optimization problem subject to dynamic (differential-algebraic) constraints. As in the single-objective case, these problems can be very challenging to solve due to the highly constrained, non-linear and sometimes non-smooth nature of most bioprocess models [1]. Furthermore, the optimization problems associated with the integration of design and control are frequently multimodal (non-convex) as described by e.g. Schweiger and Floudas [7]. Thus, computing the Pareto-optimal set is far from trivial and global optimization methods are needed to find suitable solutions.

Many methods have been suggested for finding Pareto-optimal solutions. Full reviews can be found in the books by Deb [3] and Miettinen [4]. Traditionally, the most common strategy is to combine multiple criteria into one single objective function (e.g. a weighted sum of the objectives), or to optimize one of the objectives while the others are converting to inequality constraints. In order to obtain different solutions, these approaches require solving repeatedly a set of single non-linear programming (NLP) problems (e.g. by changing the weights), and the solution depends largely on the chosen parameters.

On the other hand, an increasing number of evolutionary, population-based, algorithms have been developed in the last decade for handling multi-objective optimization problems. Since a set of candidate solutions is used in each iteration, these methods are capable of finding multiple Pareto-optimal solutions in one single optimization run.

In this contribution, we present an alternative multi-objective optimization approach which is ultimately based on an extension of a recent Evolution Strategy (ES) for single-objective NLPs. The proposed approach makes use of a well known multicriteria optimization method to generate an even spread of points in the Pareto front. The usefulness and efficiency of this novel strategy are illustrated by solving a wastewater treatment plant case study.

2. Problem Statement

Given a process dynamic model, the multi-objective optimization problem can be mathematically stated, without loss of generality, as follows:

Find $\mathbf{v}$ to minimize simultaneously

$$\mathbf{J}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}) = \left[\; J_1(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}), \; J_2(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}), \; \ldots, \; J_m(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}) \; \right]^T \tag{1}$$

subject to:

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{v}) = 0 \tag{2}$$

$$\dot{\mathbf{x}}(t_0) = \mathbf{x_0} \tag{3}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{v}) = 0 \tag{4}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{v}) \leq 0 \tag{5}$$

$$\mathbf{v}^L \leq \mathbf{v} \leq \mathbf{v}^U \tag{6}$$

where $\mathbf{J}$ is the vector of objective functions, $\mathbf{v}$ is the vector of decision variables, $\mathbf{f}$ is the set of differential and algebraic equality constraints describing the system dynamics (i.e. the nonlinear process model), $\mathbf{x}$ is the vector of state variables, $\mathbf{h}$ and $\mathbf{g}$ are possible equality and inequality path and point constraints which express additional requirements for the process performance, and $\mathbf{v}^L$ and $\mathbf{v}^U$ are the lower and upper bounds for the decision variables.

The solution to this problem is a set of points known as Pareto-optimal. A feasible solution $\mathbf{v}^*$ is said to be *Pareto-optimal* solution if there is no $\mathbf{v}$ such that $J_i(\mathbf{v}) \leq J_i(\mathbf{v}^*)$, for all $i = 1, ..., m$, with at least one strict inequality. The vector $\mathbf{J}(\mathbf{v}^*)$ is said to be *non-dominated*.

3. Multi-Objective Optimization Methods

Computing the Pareto-optimal set can be a very challenging task due to the highly constrained and non-linear nature of most biochemical systems. In this regard, the ability of evolutionary, population-based, algorithms to deal with problems involving non-convex Pareto fronts makes them attractive to solve highly nonlinear multi-objective problems. As a drawback, a very large population size is usually required, which is translated into a large number of Pareto-optimal solutions. Besides the associated rise in computational effort, such a large set can be very difficult to handle, especially as the number of objectives increases. In order to facilitate the selection of a suitable compromise, from a practical point of view it would be desirable to generate only a small number of optimal solutions capturing the complete trade-off among the objectives.

3.1 NBI-based Evolution Strategy

Here we propose a new multi-objective evolutionary algorithm which combines the recent SRES (Stochastic Ranking Evolution Strategy) by Runarsson and Yao [6] with the Normal Boundary Intersection (NBI) method developed by Das & Dennis [2].

NBI transforms the original multi-objective optimization problem into a set of NLPs which are solved sequentially. Mathematically, the so-called NBI subproblem is formulated as:

$$\min_{\mathbf{v},\gamma} \quad -\gamma \tag{7}$$

subject to:

$$\Phi\mathbf{w} + \gamma\mathbf{n} = \mathbf{J}(\mathbf{v}) - \mathbf{J}^* \tag{8}$$

Φ is the $m \times m$ *pay-off* matrix in which the i^{th} column is $\mathbf{J}(\mathbf{v}_i^*) - \mathbf{J}^*$, where $\mathbf{J}^*$ is the vector containing the individual minima of objectives (i.e., the *Utopia* point) and $\mathbf{v}_i^*$ is the minimizer of the i^{th} objective; $\mathbf{w}$ is a vector of weights such that $\sum_{i=1}^{m} w_i = 1$, $w_i \geq 0$ and $\mathbf{n}$ is the unit normal to the *Convex Hull of Individual Minima* (CHIM) pointing towards the origin. Thus, $\Phi\mathbf{w}$ defines a point in the CHIM and $\Phi\mathbf{w} + \gamma\mathbf{n}$, $\gamma \in \Re$, represents the set of points on the normal. In practice, the method uses a *quasi-normal* direction given by an equally-weighted linear combination of the columns of Φ multiplied by -1.

The solution to the above NLP is expected to be Pareto-optimal. If the NBI-subproblem is systematically solved for an equally distributed set of weights $\mathbf{w}$, an even spread of non-dominated points can be obtained (Fig. 1). It should be noted that this technique requires minimizing each objective function individually to find the *utopia* vector.

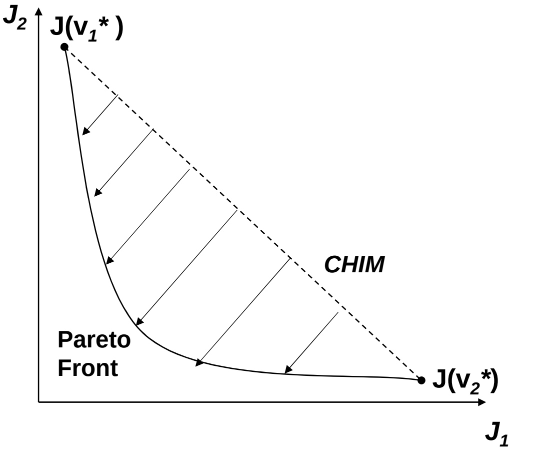


Figure 1. How NBI works.

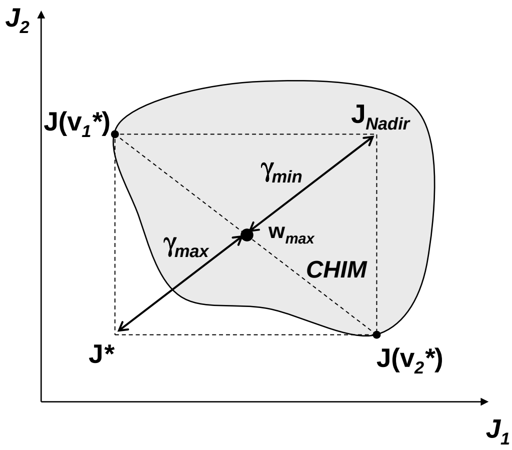


Figure 2. Bounds for parameter γ.

The original MATLAB implementation works by solving the associated NLPs by means of a Sequential Quadratic Programming (SQP) method. Thus, it can fail with non-convex problems. In order to surmount this difficulty, the SQP solver can be replaced by a suitable stochastic algorithm like SRES, which has been found to be efficient and robust even for large nonlinear problems [1]. SRES is based on a (μ,λ)-ES where each member in the population λ is sorted according to the stochastic ranking scheme proposed for handling the constraints (see Ref. [6] for details). The best μ individuals are selected to create a new population of size λ.

The new approach proposed here takes advantage of the population-based component of SRES to *simultaneously* solve the set of NBI-subproblems associated to an equally distributed set of K weights vectors. This allows to generate an even spread of Pareto-optimal solutions in one single optimization run instead of solving repeatedly the NBI-subproblem with different weight vectors. This is done in the following way:

For each weight vector $\mathbf{w}_k$, $k \in \{1, \ldots, K\}$, a fitness function is defined as:

$$\varphi_k = \bar{\gamma}_k + R_P \cdot \Sigma\bar{h}^2_{NBI,k} \tag{9}$$

where $\bar{h}_{NBI,k}$ are the normalized equality constraints introduced by NBI (Eq. 8), R_P is a penalty coefficient, and $\bar{\gamma_k}$ is:

$$\bar{\gamma_k} = \left(1 - \frac{\gamma - \gamma_{min}}{\gamma_{max} - \gamma_{min}}\right).$$

For a bi-objective optimization problem, the upper bound γ_{max} is defined as the euclidean distance to the *Utopia* vector from the point in the CHIM given by $\mathbf{w}_{max} = [0.5, 0.5]^T$ (Fig. 2). Similarly, the lower bound γ_{min} is the euclidean distance to the *Nadir* point, $\mathbf{J_{Nadir}}$ (i.e., the vector of upper bounds on each objective in the entire Pareto set), multiplied by -1.

In each iteration (generation), the basic procedure of the algorithm is as follows:

```
1. For each individual $j \in \{1, \quad \ldots, \lambda\}$ evaluate the objective vector and the constraints;
2. For each weight vector $k \in \{1, \quad \ldots, K\}$:
3.     For each individual $j \in \{1, \quad \ldots, \lambda\}$:
4.         Evaluate the fitness function $\varphi_k$ (Eq. 9);
5.         Perform Stochastic Ranking;
6.         Select the best $\mu_k$ individuals ($\mu_k \approx \lambda/(3K \quad))$;
7. Update the mutation strength (non-isotropic self-adaptation);
8. Mutate the $(K \cdot \mu_k)$ parents to create the new population of size $\lambda$.
```

The algorithm has been implemented in Matlab version 6.5. For the sake of comparison, we have made an implementation of the standard NBI in which the set of NLPs are solved *sequentially* by means of SRES. Also, the original Matlab 5.0 implementation of NBI has been rewritten for running on version 6.5 (with the FMINCON code, which is part of the Matlab Optimization Toolbox).

4. Case Study: A Wastewater Treatment Plant

This case study, solved previously by Moles et al. [5] using a weighted sum approach, represents an alternative configuration of a real wastewater treatment plant located in Manresa (Spain). The process flowsheet (Fig. 3) consists of two aerated bioreactors, where the biodegradable substrate is transformed by a microbial population (biomass), and two settlers, where the activated sludge is separated from the clean water and recycled to the corresponding reactor. The activated sludge in excess is eliminated via a purge stream (q_p). The aim of the control system is to keep the substrate concentration at the output (s_2) under a certain admissible value. We consider a step disturbance to the input substrate concentration (s_i), taking the flowrate of the sludge recirculation to the first tank (qr_1) as the manipulated variable.

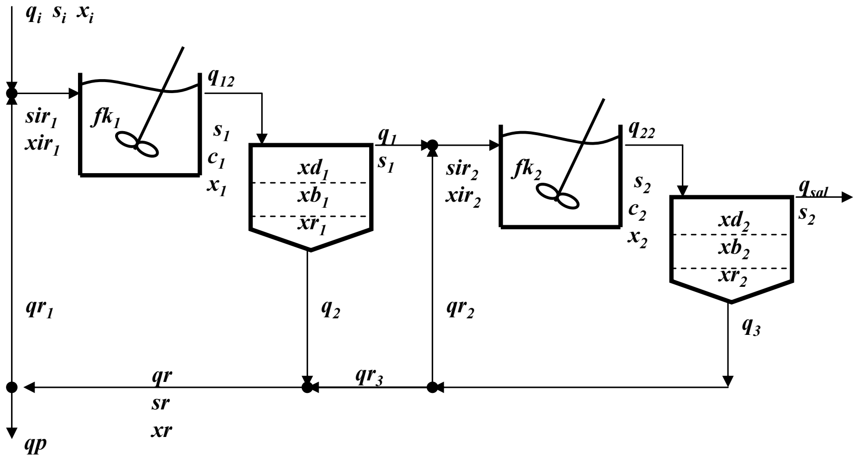


Figure 3. Process flowsheet of the wastewater treatment plant.

The dynamic model consists of a set of 33 DAEs (14 of them are ODEs) and 44 variables. Three flow rates (q_{r2}, q_{r3} and q_p) are fixed at their steady-state values corresponding to a given nominal operational conditions. Therefore, this leaves 8 design variables, namely the volume of the aeration tanks (v_1 and v_2), the areas of the settlers (ad_1 and ad_2), the aeration factors (fk_1 and fk_2), and the gain and the integral time of a PI controller.

The integrated design and control problem is formulated to minimize a weighted sum of capital and operation costs (J_1) and simultaneously a controllability measure (J_2), taken here as the Integral Square Error (ISE):

$$\mathbf{J} = \begin{bmatrix} J_1 = (c_1 v_1^2) + (c_2 v_2^2) + (c_3 ad_1^2) + (c_4 ad_2^2) + (c_5 fk_1^2) + (c_6 fk_2^2) \\ J_2 = \int_0^\infty e^2(t)dt \end{bmatrix} \tag{10}$$

subject to:

- the 33 model DAEs (system dynamics), acting as equality constraints;
- a set of 152 inequality constraints which impose limits on the state variables, residence times and biomass loads in the bioreactors, the hydraulic capacity in the settlers, the sludge ages in the decanters, and the recycle and purge flow rates.

This problem was handled by solving the DAEs using an initial value problem (IVP) solver (e.g. RADAU5) for each evaluation of the objective vector and the constraints. The IVP solver implemented in Fortran was called from Matlab using a mex-file interface.

5. Results and Discussion

First of all, the *utopia* vector was found by minimizing each objective function separately. These NLPs were solved by means of SRES with a population size $\lambda = 100$ and 100 generations. For the minimization of the economic function, SRES arrived at $J_1^* = 1023.01$, with an ISE of $J_2 = 25.7412$. For the minimization of the ISE, the best value found was $J_2^* = 0.2828$, with a cost of $J_1 = 2596.33$.

The NBI-based Evolution Strategy suggested here was applied with a set of weights $K = 15$ (the uniform spacing between two consecutive w_i is 1/16). Population sizes λ of 150 and 250 were used, and the algorithm was run for 500 generations in order to assure convergence. For both population sizes, the best Pareto fronts obtained are quite similar (Fig. 4) and present a very good distribution of points. Computation times[1] of 12 and 17 hours were required, respectively, but it is important to mention that these times can be reduced since the majority of points could be found with a lesser number of generations.

For the problem under consideration, an obvious conclusion is that a considerable improvement of the economic objective can be obtained while maintaining a very good controllability (low values of the ISE), but for a cost of about 1470, more economic designs can only be achieved at the expense of a large increase in the ISE.

This problem was also solved using the standard NBI with the gradient-based solver, taking as initial guess the optimal solution of the previous NLP. Fig. 5 shows the results obtained for $K = 15, 20$ and 30. For the majority of NLPs, FMINCON did not converge to a solution and, although better points can be found as K increases, not all the solutions were Pareto-optimal.

When SRES is used as solver in the standard NBI, the initial population is created from the solution of the previous NLP, including also the solutions of the NBI-subproblems that have just been solved. For the chosen $K = 15$, the Pareto front was obtained after a computation time of 43 hours, with $\lambda = 100$ and 200 generations (Fig. 5). Although the Pareto-optimal set is practically equal to that obtained using the NBI-based ES, the computational effort required was about 3 times more.

[1] Computation times are referred to a Pentium IV-2.4 GHz

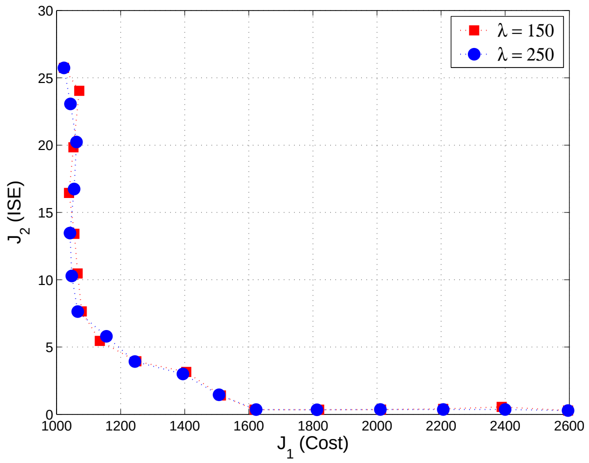


Figure 4. Pareto-optimal sets obtained with the NBI-based ES.

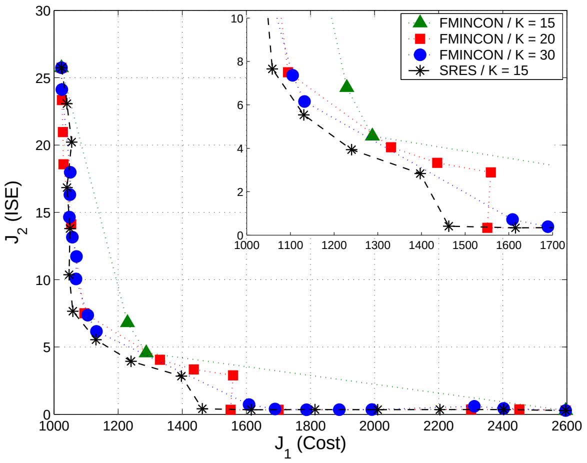


Figure 5. Pareto-optimal sets obtained with the standard NBI.

6. Conclusions

In this work, we have presented an alternative method for the multi-objective optimization of nonlinear dynamic systems. The NBI-based Evolution Strategy has been specially designed to produce an even spread of Pareto-optimal solutions in one single optimization run, instead of solving repeatedly a set of NLPs. This novel approach has been successfully applied to the integrated design and control of a wastewater treatment plant. A very good approximation of the optimal trade-offs between cost and controllability has been generated with less computational effort than the standard NBI (using also SRES as solver). It should be noted that this problem could not be solved satisfactorily with traditional gradient-based methods.

In the near future, we will consider selected case studies where the NBI-based ES will be compared with other techniques. Also, new ideas will be suggested to increase its efficiency.

Acknowledgments

Author José-Oscar H. Sendín acknowledges a pre-doctoral grant from the I3P programme of the Spanish Council for Scientific Research (CSIC).

References

[1] Banga, J.R., Moles, C.G. and Alonso, A.A. (2003). “Global Optimization of Bioprocesses using Stochastic and Hybrid Methods,” In *Frontiers in Global Optimization*, (Floudas, C.A. and Pardalos, P.M., eds.). Kluwer Academic Publishers, Dordrecht.

[2] Das, I and Dennis, J.E. (1998). “Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems,” *SIAM J. Optimization, Vol 8*, 631–657.

[3] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester.

[4] Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht.

[5] Moles, C.G., Gutierrez, G., Alonso, A.A. and Banga, J.R. (2003). “Integrated design and control via global optimization: A wastewater treatment plant case study,” *Chemical Engineering Research & Design*, 81, 507–517.

[6] Runarsson T.P. and Yao X. (2000), “Stochastic Ranking for Constrained Evolutionary Optimization,” *IEEE Transac. Evol. Comp., Vol. 4*, 284–294.

[7] Schweiger C.A. and Floudas A. (1997), “Interaction of Design and Control: Optimization with Dyamic Models,” In *Optimal Control Theory, Algorithms and Applications*, (Hager, W.W. and Pardalos, P.M., eds.), Kluwer Academic Publishers, Dordrecht.

Proceedings of GO 2005, pp. 219 – 224.

Diagonal global search based on a set of possible Lipschitz constants*

Yaroslav D. Sergeyev[1,3] and Dmitri E. Kvasov[2,3]

[1] *Dipartimento di Elettronica, Informatica e Sistemistica, Universitá della Calabria, Via P.Bucci, Cubo 41C – 87036 Rende (CS), Italy,* yaro@si.deis.unical.it

[2] *Dipartimento di Statistica, Universita' di Roma "La Sapienza", P.le A. Moro 5 – 00185 Roma, Italy,* kvadim@si.deis.unical.it

[3] *N. I. Lobachevski State University of Nizhni Novgorod, Russia*

Keywords: Global optimization, black-box functions, derivative-free methods, partition strategies, diagonal approach

In this presentation, the problem of global minimization of a multidimensional multiextremal "black-box" function satisfying the Lipschitz condition over a hyperinterval $D \subset \mathbb{R}^N$ with an unknown Lipschitz constant L is considered:

$$f^* = f(x^*) = \min_{x \in D} f(x), \tag{1}$$

$$|f(x') - f(x'')| \le L\|x' - x''\|, \quad x', x'' \in D, \quad 0 < L < \infty, \tag{2}$$

where

$$D = [a, b] = \{x \in \mathbb{R}^N : a(j) \le x(j) \le b(j),\ 1 \le j \le N\}, \tag{3}$$

a, b are given vectors in $\mathbb{R}^N$, and $\|\cdot\|$ denotes the Euclidean norm.

The function $f(x)$ is supposed to be non-differentiable. Hence, optimization methods using derivatives cannot be used for solving problem (1)–(3). It is also assumed that evaluation of the objective function at a point is a time-consuming operation.

Numerous algorithms have been proposed (see, e.g., [1, 2, 4–10, 12]) for solving problem (1)–(3). In these algorithms, several approaches for specifying the Lipschitz constant can be considered.

First, it can be given a priory (algorithms of this kind were surveyed in [4, 5]). This case is very important from the theoretical viewpoint but is not frequently encountered in practice. The more promising and practical approaches are based on an adaptive estimation of L in the course of the search. In such a way, algorithms can use either a global estimate of the Lipschitz constant (see, e.g., [8, 9, 12]) valid for the whole region D from (3), or local estimates L_i valid only for some subregions $D_i \subseteq D$ (see, e.g., [7, 10, 12]).

An interesting approach for solving problem (1)–(3) has been proposed in [6]. At each iteration of this algorithm, called DIRECT, instead of only one estimate of the Lipschitz constant a set of possible values of L is used. Due to its simplicity and efficiency, DIRECT has been widely adopted in practical applications (see references in [1]). However, some aspects limit the applications of DIRECT, especially when multidimensional multiextremal "black-box" functions are to be minimized.

*This work has been partially supported by the following grants: FIRB RBAU01JYPN, FIRB RBNE01WBBB, and RFBR 04-01-00455-a.

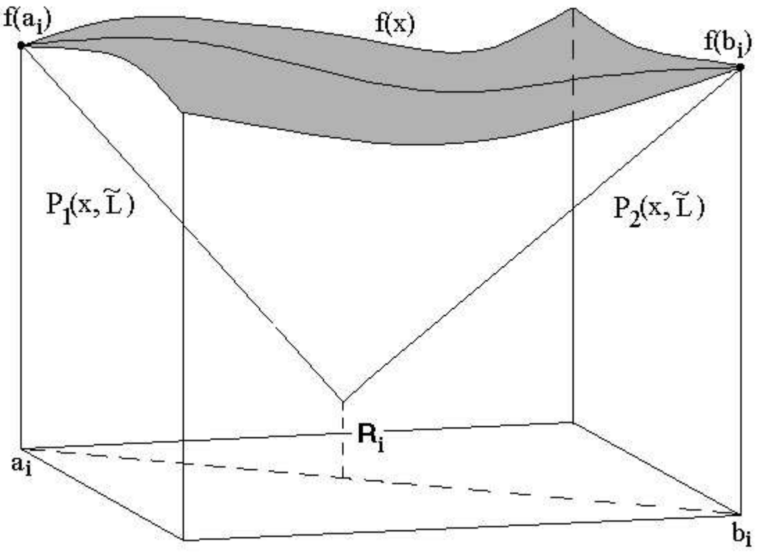


Figure 1. Estimation of the lower bound of $f(x)$ over an interval $D_i = [a_i, b_i]$.

The goal of this work is to present a new algorithm oriented to solving difficult multidimensional multiextremal "black-box" problems (1)–(3). In the algorithm, partition of the admissible region into a set of smaller hyperintervals is performed by a new efficient diagonal partition strategy (see [8, 11]). This strategy allows one to accelerate significantly the search procedure in terms of function evaluations with respect to the traditional diagonal partition strategies (see, e.g., [9]), especially in high dimensions. A new technique balancing usage of the local and global information has been also incorporated in the new method.

Following the diagonal approach (see [9]), the objective function is evaluated at two vertices of each hyperinterval (see Fig. 1). The procedure of estimating the Lipschitz constant evolves the ideas of the center-sampling method DIRECT from [6] to the case of diagonal algorithms.

Particularly, in order to calculate the lower bounds of $f(x)$ over hyperintervals, possible estimates of the Lipschitz constant varying from zero to infinity are considered at each iteration of the proposed diagonal algorithm. An auxiliary function is considered over the main diagonal of each hyperinterval $D_i = [a_i, b_i]$. This function is constructed as maximum of two linear functions $P_1(x, \tilde{L})$ and $P_2(x, \tilde{L})$ passing with the slopes $\pm\tilde{L}$ through the vertices a_i and b_i (see Fig. 1). An estimate of the lower bound of $f(x)$ over the main diagonal of D_i is calculated at the intersection of these two lines and is given by the following formula (see [7, 9])

$$R_i = R_i(\tilde{L}) = \frac{1}{2}(f(a_i) + f(b_i) - \tilde{L}\|b_i - a_i\|), \quad 0 < L \le \tilde{L} < \infty. \tag{4}$$

For any $\tilde{L} \ge L$, the value R_i is the lower bound of $f(x)$ over the diagonal $[a_i, b_i]$, but not over the whole hyperinterval D_i. It is shown, that inequality

$$\tilde{L} \ge \sqrt{2}L$$

guarantees that the value R_i from (4) is a valid estimate of the lower bound of $f(x)$ over the whole hyperinterval D_i, i.e.,

$$R_i(\tilde{L}) \le f(x), \quad x \in D_i.$$

Numerical results performed to compare the new algorithm with two algorithms belonging to the same class of methods for solving problem (1)–(3) – the original DIRECT algorithm from [6] and its locally-biased modification DIRECTl from [1, 2] – are presented. In the following, we briefly describe results of wide numerical experiments performed.

Table 1. Number of trial points for GKLS test functions.

N	Δ	Class			50%			100%	
		r^*	ρ^*	DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	0.90	0.20	111	152	166	1159	2318	403
2	10^{-4}	0.90	0.10	1062	1328	613	3201	3414	1809
3	10^{-6}	0.66	0.20	386	591	615	12507	13309	2506
3	10^{-6}	0.90	0.20	1749	1967	1743	>1000000 (4)	29233	6006
4	10^{-6}	0.66	0.20	4805	7194	4098	>1000000 (4)	118744	14520
4	10^{-6}	0.90	0.20	16114	33147	15064	>1000000 (7)	287857	42649
5	10^{-7}	0.66	0.30	1660	9246	3854	>1000000 (1)	178217	33533
5	10^{-7}	0.66	0.20	55092	126304	24616	>1000000 (16)	>1000000 (4)	93745

Table 2. Number of hyperintervals for GKLS test functions.

N	Δ	Class			50%			100%	
		r^*	ρ^*	DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	0.90	0.20	111	152	269	1159	2318	685
2	10^{-4}	0.90	0.10	1062	1328	1075	3201	3414	3307
3	10^{-6}	0.66	0.20	386	591	1545	12507	13309	6815
3	10^{-6}	0.90	0.20	1749	1967	5005	>1000000	29233	17555
4	10^{-6}	0.66	0.20	4805	7194	15145	>1000000	118744	73037
4	10^{-6}	0.90	0.20	16114	33147	68111	>1000000	287857	211973
5	10^{-7}	0.66	0.30	1660	9246	21377	>1000000	178217	206323
5	10^{-7}	0.66	0.20	55092	126304	177927	>1000000	>1000000	735945

In all the experiments, the GKLS-generator described in [3] (and free-downloadable from *http://wwwinfo.deis.unical.it/~yaro/GKLS.html*) has been used. It generates several classes of multidimensional and multiextremal test functions with known local and global minima. The procedure of generation consists of defining a convex quadratic function (paraboloid) systematically distorted by polynomials. Each test class provided by the generator includes 100 functions and is defined by the following parameters:

N – problem dimension;

M – number of local minima;

f^* – value of the global minimum;

ρ^* – radius of the attraction region of the global minimizer;

r^* – distance from the global minimizer to the vertex of the paraboloid.

The other necessary parameters are chosen randomly by the generator for each test function of the class. Note, that the generator produces always the same test classes for a given set of the user-defined parameters.

By changing the user-defined parameters, classes with different properties can be created. For example, fixed dimension of the functions and number of local minima, a more difficult class can be created either by shrinking the attraction region of the global minimizer, or by moving the global minimizer far away from the paraboloid vertex.

In executed numerical experiments, eight GKLS classes of continuously differentiable test functions of dimensions $N = 2$, 3, 4, and 5 have been used. The number of local minima

Table 3. Number of trial points for shifted GKLS test functions.

N	Δ	Class		50%			100%		
		r^*	ρ^*	DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	0.90	0.20	111	146	165	1087	1567	403
2	10^{-4}	.090	0.10	911	1140	508	2973	2547	1767
3	10^{-6}	0.66	0.20	364	458	606	6292	10202	1912
3	10^{-6}	0.90	0.20	1485	1268	1515	14807	28759	4190
4	10^{-6}	0.66	0.20	4193	4197	3462	37036	95887	14514
4	10^{-6}	0.90	0.20	14042	24948	11357	251801	281013	32822
5	10^{-7}	0.66	0.30	1568	3818	3011	102869	170709	15343
5	10^{-7}	0.66	0.20	32926	116025	15071	454925	> 1000000(1)	77981

Table 4. Number of hyperintervals for shifted GKLS test functions.

N	Δ	Class		50%			100%		
		r^*	ρ^*	DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	0.90	0.20	111	146	281	1087	1567	685
2	10^{-4}	0.90	0.10	911	1140	905	2973	2547	3227
3	10^{-6}	0.66	0.20	364	458	1585	6292	10202	5337
3	10^{-6}	0.90	0.20	1485	1268	4431	14807	28759	12949
4	10^{-6}	0.66	0.20	4193	4197	14961	37036	95887	73049
4	10^{-6}	0.90	0.20	14042	24948	57111	251801	281013	181631
5	10^{-7}	0.66	0.30	1568	3818	17541	102869	170709	106359
5	10^{-7}	0.66	0.20	32926	116025	108939	454925	> 1000000	685173

M was equal to 10 and the global minimum value f^* was equal to -1.0 for all classes (these values are default settings of the generator). For each dimension N we considered two test classes: a simple class and a difficult one. The difficulty of a class was increased either by decreasing the radius ρ^* of the attraction region of the global minimizer (as for two- and five-dimensional classes), or by increasing the distance r^* from the global minimizer x^* to the paraboloid vertex (three- and four-dimensional classes).

The global minimizer $x^* \in D$ was considered to be found when the algorithm generated a trial point x' inside a hypercube with a vertex x^* and the volume smaller than the volume of the initial hypercube $D = [a, b]$ multiplied by an accuracy coefficient Δ, $0 < \Delta \leq 1$, i.e.,

$$|x'(i) - x^*(i)| \leq \sqrt[N]{\Delta}(b(i) - a(i)) \qquad (5)$$

for all i, $1 \leq i \leq N$, where N is from (3).

The algorithm stopped either when the maximal number of trials equal to 1 000 000 was reached, or when condition (5) was satisfied.

In view of the high computational complexity of each trial of the objective function, the methods were compared in terms of the number of evaluations of $f(x)$ required to satisfy condition (5). The number of hyperintervals generated until condition (5) is satisfied, was taken as the second criterion for comparison of the methods. This number reflects indirectly degree of qualitative examination of D during the search for a global minimum.

Results of numerical experiments with eight GKLS tests classes are reported in Tables 1 and 2. These tables show, respectively, the maximal number of trials and the corresponding number of generated hyperintervals required for satisfying condition (5) for a half of the

Table 5. Improvement obtained by the new algorithm with respect to DIRECT and DIRECT*l*.

N	Δ	Class		GKLS		Shifted GKLS	
		r	ρ	DIRECT/New	DIRECTl/New	DIRECT/New	DIRECTl/New
2	10^{-4}	0.90	0.20	2.88	5.75	2.70	3.89
2	10^{-4}	0.90	0.10	1.77	1.89	1.68	1.44
3	10^{-6}	0.66	0.20	4.99	5.31	3.29	5.34
3	10^{-6}	0.90	0.20	>166.50	4.87	3.53	6.86
4	10^{-6}	0.66	0.20	>68.87	8.18	2.55	6.61
4	10^{-6}	0.90	0.20	>23.45	6.75	7.67	8.56
5	10^{-7}	0.66	0.30	>29.82	5.31	6.70	11.13
5	10^{-7}	0.66	0.20	>10.67	>10.67	5.83	>12.82

functions of a particular class (columns "50%") and for all 100 function of the class (columns "100%"). The notation "$>$ 1 000 000 (k)" means that after 1 000 000 trials the method under consideration was not able to solve k problems.

In order to avoid a redundant accumulation of trial points generated by DIRECT near the paraboloid vertex (with the function value equal to 0) and to put DIRECT and DIRECT*l* in a more advantageous situation, we shifted all generated functions, adding to their values the constant 2. In such a way, the value of each function at the paraboloid vertex became equal to 2.0 and the global minimum value f^* was increased by 2, i.e., became equal to 1.0. Results of numerical experiments with shifted GKLS classes (defined in the rest by the same parameters) are reported in Tables 3 and 4.

Note that on a half of test functions from each class (which were simple for each method with respect to the other functions of the class) the new algorithm manifested a good performance with respect to DIRECT and DIRECT*l* in terms of the number of generated trial points (Tables 1, 3). When all functions were taken in consideration, the number of trials produced by the new algorithm was significantly fewer in comparison with two other methods (see columns "100%" of Tables 1, 3), providing at the same time a good examination of the admissible region (see Tables 2, 4).

Table 5 summarizes (based on the data from Tables 1 – 4) results of numerical experiments performed on 1600 test functions from GKLS and shifted GKLS continuously differentiable classes. It represents the ratio between the maximal number of trials performed by DIRECT and DIRECT*l* with respect to the corresponding number of trials performed by the new algorithm.

As it can be seen from Tables 1 – 5, the new method demonstrates a quite satisfactory performance with respect to DIRECT [6] and DIRECT*l* [1, 2] when multidimensional functions with a complex structure are minimized.

References

[1] J. M. Gablonsky. *Modifications of the DIRECT algorithm*. PhD thesis, North Carolina State University, Raleigh, North Carolina, 2001.

[2] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *J. Global Optim.*, 21(1):27–37, 2001.

[3] M. Gaviano, D. Lera, D. E. Kvasov, and Ya. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Software*, 29(4):469–480, 2003.

[4] R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1995.

[5] R. Horst and H. Tuy. *Global Optimization – Deterministic Approaches*. Springer, Berlin, 1993.

[6] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. Optim. Theory Appl.*, 79(1):157–181, 1993.

[7] D. E. Kvasov, C. Pizzuti, and Ya. D. Sergeyev. Local tuning and partition strategies for diagonal GO methods. *Numer. Math.*, 94(1):93–106, 2003.

[8] D. E. Kvasov and Ya. D. Sergeyev. Multidimensional global optimization algorithm based on adaptive diagonal curves. *Comput. Math. Math. Phys.*, 43(1):42–59, 2003.

[9] J. Pintér. *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht, 1996.

[10] Ya. D. Sergeyev. An information global optimization algorithm with local tuning. *SIAM J. Optim.*, 5(4):858–870, 1995.

[11] Ya. D. Sergeyev. An efficient strategy for adaptive partition of N-dimensional intervals in the framework of diagonal algorithms. *J. Optim. Theory Appl.*, 107(1):145–168, 2000.

[12] R. G. Strongin and Ya. D. Sergeyev. *Global optimization with non-convex constraints: Sequential and parallel algorithms*. Kluwer Academic Publishers, Dordrecht, 2000.

Survivable Network Design under Various Interdiction Scenarios

J. Cole Smith,[1] Fransisca Sudargho,[2] and Churlzu Lim[1]

[1]*Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL,* {cole,clim}@sie.arizona.edu

[2]*Department of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ,* siscasud@email.arizona.edu

Abstract We examine the problem of building or fortifying a network to defend against enemy attack scenarios. In particular, we examine the case in which an enemy can destroy all or part of the arcs that we construct on the network, subject to some interdiction budget. This problem takes the form of a three-level, two-player game, in which we act first to construct our network and simultaneously transmit an initial set of flows through the network. The enemy acts next to destroy a set of constructed arcs in our network, and we act last to transmit a final set of flows in the network. Most studies of this nature assume that the enemy will act optimally; however, in real-world scenarios one cannot assume rationality on the part of the enemy. Hence, we prescribe network design principals for three different profiles of enemy action: an enemy destroying arcs based on capacities, based on initial flows, or acting optimally to minimize our maximum revenues obtained from transmitting flows.

Keywords: Network design, integer programming, network interdiction, game theory.

1. Introduction

The continuous increase of competition among suppliers due to society's fast growing telecommunications and transportation needs has made the design of cost-efficient networks that meet requirements concerning flexibility and survivability a major challenge. Many traditional network design algorithms do not take into account the survivability aspect of a network, which can perhaps leave the network susceptible to failures of small subsets of its arcs. It is important to design networks that are robust with respect to accidental failures like transportation breakdowns, road closures, and telephone line breaks, or some failures made maliciously by enemy entities. Important applications that inspire this research include networks in voice and data communication, military services, and transportation.

In this paper, we discuss the design of capacitated networks built in anticipation of a competitor or an enemy who inflicts some destruction to the networks. Some prior research efforts consider the enemy's response to be some stochastic scenarios with estimated probabilities of happening. By contrast, as is often the case in real-world scenarios, we consider the case in which the enemy has its own objective function with a budget constraint on destroying the arcs we construct, and may or may not be smart enough to optimize it.

This problem can be modeled as a three-stage problem. In the first stage, we construct a network in which each arc has a fixed construction cost, a maximum capacity, and a per-unit flow cost. We also stipulate a budget on the arc construction costs. Before the enemy acts, we place an initial set of multicommodity flows on the network, which gives us some measure of initial profits (measured by the revenue generated by successfully shipping units of commodity from origins to destinations minus the flow costs). The purpose of these flows might not only be to generate initial revenues for us, but might also be used as decoys to trick the opponent into making a poor interdiction decision, if their algorithm relies on these initial flows.

In the second stage, the enemy acts to inflict damage to our network arcs by reducing the capacity of certain arcs (perhaps setting some of their capacities to zero). This action is often referred to as interdiction. The enemy's objective function can vary depending on its goal. In particular, we consider three different cases of the enemy's objective function. The first case handles the situation in which the enemy destroys the largest-capacity arcs in our network design. The second case deals with the problem in which the enemy destroys arcs having the largest initial commodity flows. In the third case, the enemy makes an effort to minimize the total profit obtained from transporting commodities to the destination nodes in the network. The first two cases might represent the case in which our enemy intends to maximally disrupt our network, but is acting according to a heuristic strategy due to real-time considerations or due to the complexity of the system. For all of these cases, it is important to note that the enemy can destroy portions of arcs by removing some of their capacity, rather than being restricted to integer interdiction actions.

In the third stage, after the enemy interdiction, some of the arcs in the network will be inaccessible and we will modify the flow in the surviving proportion of the arcs in order to meet the demand. If some arcs in the network that consist of some demand flows should fail and become completely or partially unavailable, another flow pattern must be built on the existing (previously constructed) arcs. There might not be any possible flows at all if there exists a small cut-set of network arcs that are inexpensive to destroy. Under this condition, redundant arcs and nodes in the network design are needed to ensure the survivability of the network, given the importance of meeting demands under inopportune situations.

The network interdiction problem has received much attention in the literature due to its applications in military and homeland security operations. Such papers form the basis for solving the last two echelons of the problem considered in this paper. Wollmer [6] proposes an algorithm that interdicts a prescribed number of arcs in a network in order to minimize the follower's maximal flow. Wood [7] provides an integer programming formulation for a discrete interdiction problem, and provided an extension of the model to allow for continuous interdiction, multiple sources and sinks, undirected networks, multiple interdiction resources, and multiple commodities. This work is continued by Cormican et al. [1], in which the leader minimizes the expected maximum flow, given uncertainties regarding the success of interdiction and arc capacities. A different interdiction problem is examined by Fulkerson and Harding [2], who examine the problem of maximizing the shortest source-sink path in the presence of arc-extension costs, which serve as interdiction costs. A recent study by Israeli and Wood [3] develops two decomposition algorithms using super-valid inequalities and set covering master problems. Finally, Lim and Smith [4] examine the multicommodity flow network interdiction problem under assumptions of discrete and continuous enemy action.

2. Models and Solution Methods

In this section, we describe models and solution methods under three different interdiction scenarios. To facilitate our discussion, consider a graph $G(N, A)$, with node set N and arc set A. Associated with each arc $i \in A$ are a nonnegative construction cost c_i, a nonnegative disruption cost b_i, and a mutual arc capacity q_i. Define $RS(j)$ and $FS(j)$, $\forall j \in N$, to be the set of arcs going to and going from node j, respectively. Furthermore, define K to be the set of commodities, and let d_k^j denote the demand of commodity $k \in K$ at node $j \in N$. If $d_k^j > 0$, then j is a supply node of commodity k, while $d_k^j < 0$ implies that j is a destination of commodity k. Without loss of generality, we assume that $\sum_{j \in N} d_k^j = 0$, $\forall k \in K$. We are given per-unit flow revenue f_k^i for transmitting a unit of commodity k over arc i for each $i \in A$ and $k \in K$. This flow revenue accommodates the flow cost as well as the reward as necessary. That is, f_k^i includes the reward for commodity k if it enters a destination node of k, and has this reward subtracted from the flow cost if it exits a destination node for k. If neither or both of these cases hold, f_k^i is simply the negative per-unit flow cost associated with arc i and commodity k. The enemy's interdiction is subject to a budget limitation of B, given the different disruption costs b_i for each arc $i \in A$. We will introduce some dummy arcs in the network to ensure the feasibility of our models; these dummy arcs have zero construction and flow costs, and large enough interdiction costs and capacities to ensure that they cannot be effectively destroyed by the enemy.

The first set of decision variables determines whether or not we construct an arc. Let x_i, $\forall i \in A$, be a binary decision variable that equals to 1 if arc i is constructed and 0 otherwise. For the flow decision variables, we let u_k^i and v_k^i, $\forall i \in A, \forall k \in K$, be the decision variables that represent actual flows of commodity k before and after the enemy's interdiction, respectively. Also, let $w_i \in [0, 1]$, $\forall i \in A$, represent the remaining percentage of arc i after the enemy destroys his/her preferred set of arcs. Although w_i is determined by the enemy, we view it as a decision variable induced by our choice of x-variables. Finally, we define $\tilde{u}$ and $\tilde{v}$ to be the weights of the initial flow profit and final flow profit, respectively.

2.1 Capacity-Based Greedy Interdiction Case

In this subsection, suppose that the enemy repeatedly destroys arcs with the largest capacity until the budget B is exhausted. Assume that the enemy breaks a tie with a certain preference-order known *a priori*. Order the arc indices $i = 1, \ldots, |A|$ so that the enemy prefers arc i to arc $i + 1$. A first attempt at formulating this problem is given as follows.

$$\begin{aligned}
\text{Maximize} \quad & \tilde{u} \sum_{k \in K} \sum_{i \in A} f_i^k u_i^k + \tilde{v} \sum_{k \in K} \sum_{i \in A} f_i^k v_i^k - \sum_{i \in A} c_i x_i && \text{(1a)} \\
\text{subject to} \quad & \sum_{i \in A} c_i x_i \leq C && \text{(1b)} \\
& \sum_{i \in FS(j)} u_i^k - \sum_{i \in RS(j)} u_i^k = d_j^k \quad \forall k \in K \quad \forall j \in N && \text{(1c)} \\
& \sum_{i \in FS(j)} v_i^k - \sum_{i \in RS(j)} v_i^k = d_j^k \quad \forall k \in K \quad \forall j \in N && \text{(1d)} \\
& \sum_{k \in K} u_i^k \leq q_i x_i \quad \forall i \in A && \text{(1e)} \\
& \sum_{k \in K} v_i^k \leq q_i w_i \quad \forall i \in A && \text{(1f)} \\
& w_i \leq x_i \quad \forall i \in A && \text{(1g)} \\
& w_i \geq 0 \quad \forall i \in A && \text{(1h)} \\
& u_i^k,\ v_i^k \geq 0 \quad \forall i \in A \quad \forall k \in K && \text{(1i)} \\
& x_i \in \{0, 1\} \quad \forall i \in A. && \text{(1j)}
\end{aligned}$$

Note that (1b) is a construction budget constraint, (1c)-(1d) are flow conservations before and after enemy's action, respectively, and (1e)-(1f) are arc capacities before and after enemy's action, respectively. Also, (1g) imply that the remaining portion of an arc can have a positive value only when its arc is constructed. However, these conditions are only necessary for the w-solution to this problem to reflect the true decision of the enemy, and are certainly not sufficient.

In order to force the w-variables to match the enemy's true solution, we can enact the following cutting plane method. Suppose that we solve (1) and obtain variables $\hat{x}$ and $\hat{w}$. Let $\overline{w}_i$ be the true action of the enemy for $i \in A$. For each $i \in A$ such that $\overline{w}_i < \hat{w}_i$, we add the following constraint:

$$w_i \leq \overline{w}_i x_i + \sum_{g=1}^{i-1} \left(\min\{1 - \overline{w}_i, b_g/b_i\}\right)(1 - \hat{x}_g)x_g. \tag{2}$$

Each such constraint is then passed back to (1), and the model is resolved until $\overline{w} = \hat{w}$. Essentially, (2) imposes an upper bound on w_i of $\overline{w}_i$, unless some other higher-preference arcs that are not currently being built will be built in the next iteration, thus forcing the enemy to expend resources on other arcs. There exist several other necessary conditions for w-feasibility that can be stated *a priori* in (1), but we omit them here for the sake of brevity.

A model that can be solved statically makes use of the fact that there will exist only one w-variable value that can be fractional, as in the case in any knapsack decision problem. Define binary decision variables z_i $\forall i \in A$ equal to one if and only if $x_i = 1$, all constructed arcs with an index smaller than i are completely destroyed, and all constructed arcs with an index greater than i are not affected by the enemy. Arc i itself may be interdicted in any manner. The total enemy interdiction must exhaust their entire budget. (This forces us to build at least enough capacity so that the enemy can destroy B units; we can remove this restrictive assumption by use of some simple modeling tricks.) These restrictions are enforced as follows:

$$\sum_{g \in A} z_i = 1 \tag{3a}$$

$$w_i \leq \sum_{g=1}^{i} z_g \quad \forall i \in A \tag{3b}$$

$$w_i \geq x_i - \sum_{g=i}^{|A|} z_g \quad \forall i \in A \tag{3c}$$

$$w_i \leq x_i \quad \forall i \in A \tag{3d}$$

$$\sum_{i \in A} b_i(x_i - w_i) = B. \tag{3e}$$

Our preliminary computational experience has shown that incorporating (3) into (1), and solving as a static integer program, is a much more effective method of solving this problem than by using the initial cutting-plane method developed above.

2.2 Flow-Based Greedy Interdiction Case

In this case, the enemy wishes to interdict the arcs having the most initial flows on them. Since the enemy decision does not depend on a simple set of binary decision variables, the foregoing algorithm must be modified for this case.

Once again, we determine an index i^* such that arc i^* may be partially destroyed, implying that every arc with a greater flow than i^* must be completely destroyed, while all arcs with a smaller flow than i^* cannot be interdicted. Unlike the previous case, we do not define a decision variable to determine the identity of i^*, but must solve one integer program for each possible value that i^* can take. If there exists an arc that has the same amount of flow as i^*, we will use the enemy's secondary priority to determine whether or not it is destroyed. Letting p_i $\forall i \in A$ be the enemy's secondary priority for arc i (a smaller number denotes a higher preference), we can add the following constraints to model (1):

$$\sum_{k\in K} u_i^k - \sum_{k\in K} u_{i^*}^k \leq M(x_i - w_i) + \varepsilon(p_i - p_{i^*}) \quad \forall i \neq i^*,\ i \in A \tag{4a}$$

$$\sum_{k\in K} u_{i^*}^k - \sum_{k\in K} u_i^k \leq M(1-(x_i - w_i)) + \varepsilon(p_{i^*} - p_i) \quad \forall i \neq i^*,\ i \in A \tag{4b}$$

$$\sum_{i\in A} b_i(x_i - w_i) = B \tag{4c}$$

$$0 \leq w_{i^*} \leq 1 \quad \forall i^* \in A \tag{4d}$$

$$w_i \in \{0,1\} \quad \forall i \in A - \{i^*\}, \tag{4e}$$

where ε is a very small constant. The addition of the constraints in (4) captures the flow-based greedy enemy interdiction model. If there exists more initial flow on arc i than on i^*, then since $x_i = 1$, we have by (4a) that arc i must be interdicted (by setting $w_i = 0$). If there is a tie, but i^* is preferred to i, then (4a) once again implies that $w_i = 0$. Similarly, (4b) forces $w_i = x_i$ if there is less flow on i than i^*, or if there is a tie when i has a lower priority than i^*.

2.3 Optimal Interdiction Case

Finally, we consider the third case in which the enemy optimally disrupts arcs so as to minimize our profit. We assume that the enemy has complete information of our network design including arc capacities, flow revenues, and demands. Given a network topology x, therefore, the enemy solves a continuous multicommodity flow network interdiction problem in a min-max structure. Taking the linear dual of the inner maximization problem, this problem can be reformulated as a disjointly constrained bilinear program (BLP), in which the enemy's decision variables and our dual variables associated with arc capacity constraints constitute bilinear terms in the objective function.

One important property of this BLP formulation is that a global optimum can be found among pairs of extreme points from respective feasible regions. In particular, the enemy has a single knapsack constraint besides bounds on variables, and hence, each extreme point has only one basic variable while other nonbasic variables are set at one of their bounds 0 or 1. Exploiting this fact, Lim and Smith [4] recently proposed an optimal algorithm that solves linearized mixed integer subproblems obtained by designating one variable as basic. Thus, an exact solution can be identified after solving $|A|$ subproblems. (See [4] for details.)

Note that there exists a finite number of pairs of extreme points for disjoint polyhedral sets of the bilinear programming problem. Let P denote the set of such pairs. Furthermore, let $\theta_x(p)$ denote the objective value of the bilinear program at $p \in P$ given x. Then, our network design problem can be formulated as follows.

$$\begin{aligned} \text{Maximize} \quad & \tilde{u}\sum_{k\in K}\sum_{i\in A} f_i^k u_i^k - \sum_{i\in A} c_i x_i + \tilde{v}z && \text{(5a)}\\ \text{subject to} \quad & \text{(1b)}, \text{(1e)}, \text{(1j)}, \text{ and } u_i^k \geq 0 \quad \forall i \in A \quad \forall k \in K && \text{(5b)}\\ & z \leq \theta_x(p) \quad \forall p \in P && \text{(5c)}\\ & z \quad \text{unrestricted.} && \text{(5d)} \end{aligned}$$

As a solution method, we prescribe the following cutting plane algorithm, **BCPA** that exploits Benders cuts in an iterative fashion.

Algorithm BCPA.

```
Step 0 Set $P = \emptyset$ and $k = 1$.
Step 1 Solve the problem (5) to obtain a solution $x^k$ and $z^k$.
Step 2 Given $x^k$, solve the bilinear programming problem using the algorithm of [4] to obtain a solution $p^k$ and its objective value $\theta_{x^k}(p^k)$.
Step 3 If $z^k \leq \theta_{x^k}(p^k)$, then $x^k$ is optimal and stop. Else, put $P = P \cup \{p^k\}$, increment $k \leftarrow k + 1$, and return to Step 1.
```

3. Summary

The three problems described above are all strongly NP-hard, but can be optimally solved with the using of integer programming and decomposition algorithms. Although the computational results and finer details of these algorithms have been suppressed here due to space limitations, the development of these models is only part of the challenge in solving our test problems. Additional research is being conducted on the effectiveness of initial cutting planes and model tightening procedures. For instance, one notable tightening strategy could use the Special Structures Reformulation Linearization Technique [5] to obtain the convex hull of solutions for which the z-variables are binary in the problem described by Section 2.1. We continue to explore these avenues for improving the solvability of the problems encountered in this research. Our outcomes have application in actual human subject testing, in which the subjects play the role of the enemy interdictor. These experiments will help us to examine whether or not humans rely on simple capacity-based or initial flow-based heuristics, and whether a practical network design against suboptimal humans should actually prepare for the worst-case scenario (as discussed in Section 2.3), or if it is more practical to prepare for suboptimal enemy behavior.

Acknowledgments

The authors gratefully acknowledge the support of the Office of Naval Research under Grant Number N00014-03-1-0510 and the Air Force Office of Scientific Research under Grant Number F49620-03-1-0377.

References

[1] K.J. Cormican, D.P. Morton, and R.K. Wood. Stochastic Network Interdiction. *Operations Research* **46**:184-196, 1998.

[2] D.R. Fulkerson and G.C. Harding. Maximizing Minimum Source-Sink Path Subject to a Budget Constraint. *Mathematical Programming* **13**:116-118, 1977.

[3] E. Israeli and R.K. Wood. Shortest-Path Network Interdiction. *Networks* **40**:97-111, 2002.

[4] C. Lim and J.C. Smith. Algorithms for Discrete and Continuous Multicommodity Flow Network Interdiction Problems, Working Paper, Department of Systems and Industrial Engineering, The University of Arizona, 2005.

[5] H.D. Sherali, W.P. Adams, P.J. Driscoll. Exploiting Special Structures in Constructing a Hierarchy of Relaxations for 0-1 Mixed Integer Problems. *Operations Research* **46**:396-405, 1998.

[6] R. Wollmer. Removing Arcs from a Network. *Operations Research* **12**:934-940, 1964.

[7] R.K. Wood. Deterministic Network Interdiction. *Mathematical and Computer Modelling* **17**:1-18, 1993.

Precision and Accuracy in Generating Globally Optimum Shapes

Fazil O. Sonmez

Department of Mechanical Engineering, Bogazici University, sonmezfa@boun.edu.tr

Abstract Two issues that come up during shape optimization are precision and accuracy. The optimization problem is to obtain globally optimal shapes for two-dimensional structures subject to in-plane loads. The objective function to be minimized is the volume (or weight) of the structure. Optimization is achieved by a stochastic search algorithm called Direct Simulated Annealing (DSA), which seeks the global minimum through randomly generated configurations. In order to obtain random configurations, a boundary variation technique is used. In this technique, a set of keypoints is chosen and connected by cubic splines to describe the boundary of the structure. Whenever the positions of the keypoints are changed in random directions, a new shape is obtained. Thus, coordinates of the keypoints serve as design variables. The precision of the global optimum configuration is an indication of how well the optimized system is defined by the design variables.As to the precision in our problem, the larger the number of keypoints, the more precisely the shape of a structure can be defined. The accuracy of the optimum configuration is an indication of how well it represents the global optimum configuration for the given design variables. In the following, the ways of increasing precision and accuracy in the search of global configuration are discussed.

Keywords: Shape optimization, Direct Search Simulated Annealing (DSA), boundary variation, FEM, $2D$-structures

1. Introduction

The general purpose of shape optimization is to find the best shape for a structure under various constraints imposed by the requirements of the design. The best shape provides either the most efficient and effective use of material or the best performance.

In typical structural optimization problems, there may be many locally minimum configurations. For that reason, a downhill proceeding algorithm, in which a monotonically decreasing value of objective function is iteratively created, may get stuck into a locally minimum point other than the globally minimum solution. Its success depends on the choice of initial design. In order to find the absolute minimum of an objective function without being sensitive to the starting position, a global optimization method has to be employed in structural optimization problems. The simulated annealing (SA) algorithm as one of the most popular stochastic optimization techniques is suitable in this respect. SA is relatively easier to implement and can easily be adapted to any application.

Kirkpatrick et al. [1] first proposed simulated annealing as a powerful stochastic search technique. The method attempts to model the behavior of atoms in forming arrangements in solid material during annealing. Although the atoms move randomly, as their natural behavior they tend to form lower-energy configurations. As the temperature is slowly decreased, the arrangement of the atoms gets closer and closer to the lower energy state.

There is an analogy between the physical annealing process and an optimization process. Different configurations of the problem correspond to different arrangements of the atoms. The cost of a configuration corresponds to the energy of the system. Optimal solution corresponds to the lowest energy state. Just in the same manner the atoms find their way to build a perfect crystal structure through random movements, the global optimum is reached through a search within randomly generated configurations.

The main advantage of SA is that it is generally more reliable in finding the global optimum even with large numbers of design variables. In this study, we adopted an improved version of SA called direct search simulated annealing (DSA) with slight modifications. DSA was developed by Ali, Törn and Viitanen [2], which differs from SA basically in two aspects. Firstly, DSA uses a set of current configurations rather than just one current configuration. Secondly, it always retains the best configuration. In a way, this property imparts a sort of memory to the optimization procedure.

2. Problem Statement

Consider a two dimensional structure as in Figure 1 which can be defined by its boundary and thickness. The structure is subject to in-plane static loads; it is also restrained from moving at some points of the boundary. Applied loads and restraints are considered as boundary conditions. The structure should be able to resist the loads without failure. This means stresses should not exceed the yield strength of the material. Besides, no part of the structure is to lose its connection to the restraints; i.e. the structure should remain in one piece. Therefore, the constraints imposed on the structure are the maximum allowable stress and the model connectivity. Our objective is to minimize the volume (or weight) of the structure without violating the constraints.

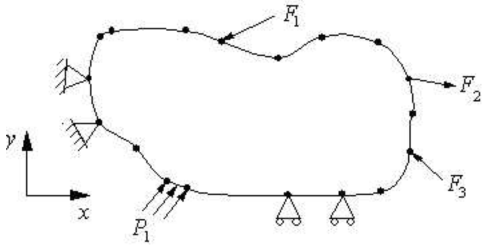


Figure 1. Representation of a $2D$ shape optimization problem

3. Problem Solution

As optimization algorithm we adopted DSA [2] with minor modifications, and applied it to the shape optimization problem as follows.

3.1 Cost Function

Because the thickness of the structure is fixed and only its lateral area is allowed to vary, the objective function to be minimized may be taken as its area instead of its volume. By integrating a penalty function for constraint violations into the cost function, the constrained optimization problem can be transformed into an unconstrained problem, for which the algorithm is suitable. A combined cost function may be constructed as

$$f = \frac{A}{A_{ini}} + c\langle\frac{\sigma_{max}}{\sigma_{allow}} - 1\rangle^2 \tag{1}$$

Here, the bracketed term is defined as

$$\langle\frac{\sigma_{max}}{\sigma_{allow}}-1\rangle=\begin{cases}0 & \text{for} \quad \sigma_{max}\leq\sigma_{allow}\\ (\sigma_{max}/\sigma_{allow})-1 & \text{for} \quad \sigma_{max}>\sigma_{allow}\end{cases} \tag{2}$$

where A_{ini} is the area of the initial configuration; c is a weighing coefficient, for which a value of 0.7 was found to be satisfactory. Any excursion into the infeasible region $(\sigma_{max} > \sigma_{allow})$ results in an increase in the cost function. On the other hand, if model connectivity is lost, the new configuration is never accepted, and therefore there is no need to calculate its cost function.

3.2 The Boundary Variation Technique

SA requires random generation of a new configuration (in our case a new shape) in each iteration. Definition of a configuration should be made based on optimization variables. Shape of a $2D$-structure can easily be described by spline curves passing through a number of keypoints. Some of these points may be fixed, while others are allowed to move during optimization. As illustrated in Figure 2, whenever the positions of the moving keypoints are changed in random directions through random distances, a new boundary, thus a new configuration is obtained. The x and y coordinates of the moving keypoints thus become optimization variables. The keypoints are allowed to move only within a region, S, defined by the designer.

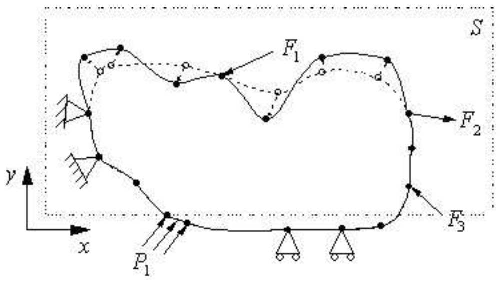


Figure 2. Generation of a random configuration through movement of keypoints

3.3 The Set of Current Configurations

DSA unlike ordinary SA starts with a set of N configurations, A, rather than starting with only one configuration. The initial set of N configurations is selected randomly within the search domain, S, and their cost function values are stored. The highest and lowest cost function values are denoted as f_h and f_l. The number of these configurations depends on the dimension of the problem.

$$N = 7(n+1) \tag{3}$$

where n is the dimension of the problem, i.e. the number of design variables. Since the x and y coordinates of the moving keypoints are the variables, in our case n is two times the number of moving keypoints.

3.4 Acceptability

Acceptability of a new configuration depends on the value of its cost function, f_t; its acceptability, A_t, is calculated by

$$A_t=\begin{cases}1 & \text{if} \quad f_t\leq f_h\\ \exp((f_h-f_t)/T_k) & \text{if} \quad f_t>f_h\end{cases} \tag{4}$$

here f_h is the highest cost function value in A. This means every new design having a cost lower than the cost of the worst design is accepted. But, if the cost is higher, the trial configuration may be accepted depending on the value of A_t. If it is greater than a randomly generated number, the trial configuration is accepted, otherwise rejected. If the trial function is accepted, it replaces the worst configuration.

3.5 Cooling Schedule

The simulated annealing process consists of first "melting" the system being optimized at a high "temperature", T, then lowering the temperature slowly until the system "freezes" and no further changes occur. Here, temperature, T, has no physical meaning; it is just a control parameter. Melting corresponds to the initial stage where configurations are generated within the solution domain, S, without much regard to the cost. At high temperatures, (high values of T) the probability of acceptance is high as Eq. (4) implies. Accordingly, configurations that have even very high cost values may be accepted, just as in the physical annealing process the atoms may form configurations having very high energy in the melting state. At low values of temperature parameter, acceptability becomes low and acceptance of worse configurations is then unlikely. The cooling schedule in SA controls the convergence of the algorithm to the global minimum just as the cooling scheme in the physical annealing process controls the final microstructure. Therefore, performance of SA depends on the cooling schedule. In a cooling schedule, first an initial value, T_o, is specified for the temperature parameter. A scheme is then required for reducing T and for deciding how many trials are to be attempted at each value of T. Lastly, the freezing (or final value of the) temperature parameter is specified.

4. Accuracy of the Optimum Design

The measure for the accuracy of the optimum configuration is the degree of how well it represents the global optimum configuration for the given design variables.

As one of the sources detracting from the accuracy, search algorithm may get stuck into one of the local optimums that fail to approximate the global optimum. One may not ensure that the resulting configuration is globally optimal, but may use relatively reliable search algorithms such as simulated annealing. With a good choice of optimization parameters, one may achieve reliability greater than 90% at the same time avoid excessively long computational times.

Another source of low accuracy is due to errors in calculating the cost of the configurations. In many structural optimization problems, maximum stress value is used either in calculating the cost function or in checking constraint violations. Designers usually carry out a finite element (FE) analysis to calculate the stress state in the structure, but they tend to choose a rough FE mesh to alleviate the computational burden. However, this may lead to erroneous values of stress, and the resulting design, as will be shown, will not be the optimum design.

5. Precision of the Optimum design

How well the optimized system is defined by the design variables is a measure for the precision of the global optimum configuration. Some of the parameters that define the system are allowed to be changed during an optimization process. The number of these parameters and the range of values they may take determine the degree at which the system can be tailored to the best performance. By increasing the number of the design variables and range of their values, one may obtain a better performance. In our shape optimization problem, by using a larger number of moving keypoints, one may better describe the boundary, i.e. shape, of the structure, and minimize its volume to a larger extent. However, the chances of locating the globally optimum design become smaller and smaller as the number of design variables increases, at the same time computational times get longer and longer. With a large number of optimization variables, locating the global minimum also becomes difficult. As the designer tries to obtain a more precise definition, he or she becomes less sure of the accuracy of the results. One may repeat the optimization process many times and designate the design having the lowest cost as the global optimum design. However, because of the long computational times this approach is not feasible. Another way to overcome this difficulty is to start optimization by choosing only a few design variables and finding the optimum design. In this case, the reliability of locating globally optimum design will be high, even though precision will be low. Then, the designer keeps increasing the number of variables and finding the optimum design. With the assumption that the difference between the optimum designs will not be large as the number of design variables is increased, one may observe convergence to the most precise global optimum design.

6. Results and Discussions

One of the problems that we considered in this study is the optimal design of an eccentrically loaded plate restrained at one end and loaded at the other (Figure 3). One segment of the border on the right is fixed in length and subject to $200MPa$ pressure. The left border is defined by a line between two keypoints; the lower one is fixed, while the upper one is free but restrained to move in the vertical direction. The problem is to find the optimum shape of this plate as stated above.

Firstly, five keypoints were used to describe the boundary and using their coordinates as design variables the best shape was found. Figure 4 shows the optimal shape with its finite element mesh. Because the number of design variables was low, the optimum design could be obtained with high reliability. Then, optimization process was repeated using seven and nine moving keypoints resulting in the optimum designs shown in figures 5 and 6. The lateral areas of the optimum shapes defined by five, seven, and nine keypoints turned out to be 37.100, 36.313, and 35.903 cm^2, respectively. Consequently, with a higher number of keypoints, we could obtain a better definition of shape and also find an optimum configuration with a lower cost. Although reliability of the solution is low when the number of design variables is large, because optimum designs defined by a high number of keypoints approximate the more reliable optimum designs defined by lower number of keypoints, the reliability is ensured.

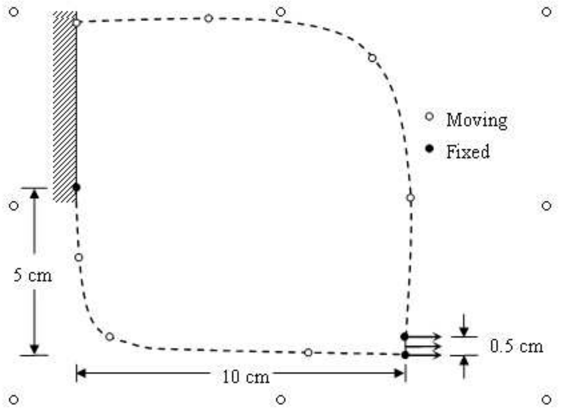


Figure 3. The design problem

In order to ensure the accuracy at which the cost was calculated, convergence of the FE solution was frequently checked during the optimization process. If the change in the magnitude of the maximum stress was greater than 1% when FE analysis was carried out using one

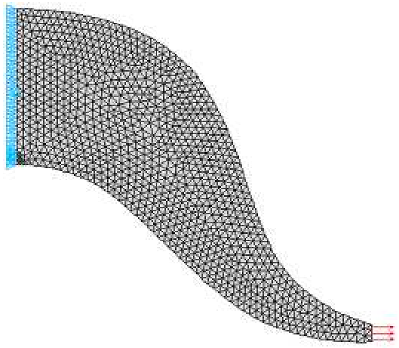


Figure 4. The optimal shape defined by five keypoints

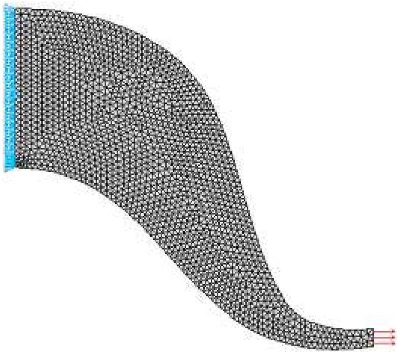


Figure 5. The optimal shape defined by seven keypoints

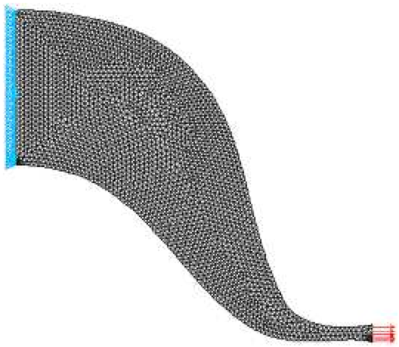


Figure 6. The optimal shape defined by nine keypoints

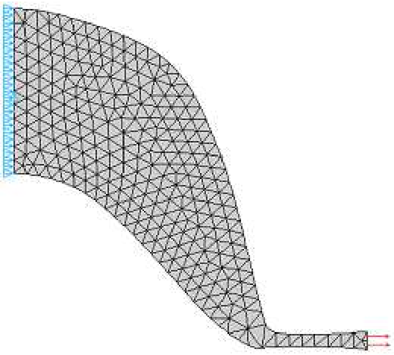


Figure 7. Inaccurate optimal shape obtained with a rough mesh

fourth of the current size of finite elements, then the E mesh was refined. Figure 7 shows the optimum shape obtained when a rough mesh was used. Although the error in stress level was only 10%, the discrepancy in shape was quite large.

7. Summary

In this extended abstract, a $2D$ shape optimization procedure based on the DSA method was presented. The importance of precise definition of the optimized system was indicated. With a more precise definition, one may find a better optimum design. Accurate calculation of the cost function was also found to be a crucial factor in global optimization. If the FE mesh is not sufficiently refined, consequently the cost function is not accurately calculated, the resulting shape could not considered to be optimum.

Acknowledgments

This paper is based on the work supported by the Scientific Research Projects of Bogazici University with the code number 01A601.

References

[1] Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. Optimization by simulated annealing. *Science*, 220:671-680, 1983.

[2] Ali, M.M., A. Törn and S. Viitanen, A direct search variant of the simulated annealing algorithm for optimization involving continuous variables. *Computers δ Operations Research*, 29:87-102, 2002.

New approach to nonconvex constrained optimization*

Alexander S. Strekalovsky

Institute of System Dynamics and Control Theory, Laboratory of Global Optimization Methods, Lermontov st. 134, Irkutsk, Russia, strekal@icc.ru

Abstract In this paper we propose new approach based on Global Optimality Conditions for solving continuous nonconvex optimization problems.

Keywords: nonconvex optimization, global optmzality conditions, local search, global search

1. Introduction

A huge of optimization problems arising from different application areas are really nonconvex problems [1]– [3]. The most of such problems deal with (d.c.) functions which can be represented as a difference of two convex functions.

The present situation in Continuous Nonconvex Optimization may be viewed as dominated by methods transferred from other sciences [1]– [3], as Discrete Optimization (Branch & Bound, cuts methods, outside and inside approximations, vertex enumeration and so on), Physics, Chemistry (simulated annealing methods), Biology (genetic and ant colony algorithms) etc.

On the other hand the classical method [7] of convex optimization have been thrown aside because of its inefficiency [1]– [6]. As well-known the conspicuous limitation of convex optimization methods applied to nonconvex problems is their ability of being trapped at a local extremum or even a critical point depending on a starting point [1]– [3]. So, the classical apparatus shows itself inoperative for new problems arising from practice.

In such a situation it seems very probable to create an approach for finding just a global solution to nonconvex problems on one side connected with Convex Optimization Theory and secondly using the methods of Convex Optimization.

Nevertheless we risked to propose such an approach [8] and even to advance the following principles of Nonconvex Optimization.

1. The linearization of the basic (generic) nonconvexity of a problem of interest and consequently a reducing of the problem to a family of (partially) linearized problems.

2. The application of convex optimization methods for solving the linearized problems and, as a consequence, within special local search methods.

3. Constructing of "good" (pertinent) approximations (resolving sets) of level surfaces and epigraph boundaries of convex functions.

Obviously, the first and the second are rather known. The deepness and effectiveness of the third may be observed in [8]– [19].

*This work was supported by RFBR Grant No. 05-01-00110

Developing the principles we get the solving methodology for nonconvex problems which can be represented as follows.

1. Exact classification of a problem under study.
2. Application of special local search methods.
3. Applying the special conceptual global search methods (strategies).
4. Using the experience of similar nonconvex problems solving to construct pertinent approximations of levels surfaces of corresponding convex functions.
5. Application of convex optimization methods for solving linearized problems and within special local search methods.

It is easy to note that this approach lifts Classical Convex Optimization up a new altitude, where the effectiveness and the fastness of the methods become of paramount importance not only for Convex Optimization, but for Nonconvex.

Our computational experience certifies that if you follow the methodology above you have more chance to reach a global solution of a nonconvex problem of a big size (≥ 1000) than applying the Branch&Bound or cuts methods.

2. Investigation

At present, together with our collaborators we are investigating teh following problems.

1. Solving the system of nonlinear equations

   $$F_i(x) = 0, \quad i = 1, ..., m$$

   by means of reducing on to the following optimization problem:

   $$F(x) = \sum_{i=1}^{m} |F_i(x)| \downarrow \min . \tag{1}$$

   Here every function F_i is supposed to be d.c. so that

   $$F_i = g_i(x) - h_i(x), i = 1, ..., m,$$

   where g_i, h_i are convex functions.

2. The polyhedral separation problem when you have to separate two finite sets of points in $I\!R^n$ by means of a minimal number of hyperplanes. This problem is equivalent to the minimization of a d.c. function which is not differentiable as it was in problem 1.
3. An optimal control problem with a terminal functional given by d.c. function.

For all problems we investigate the convergence of local and global search and carry out the computational experiments. The results of computational solving are very promising.

References

[1] Horst, R. and Tuy, H. (1993), *Global Optimization. Deterministic Approaches*, Springer-Verlag, Berlin.

[2] Horst, R., Pardalos, P.M. and Thoai, V. (1995), *Introduction to Global Optimization*, Kluwer, Dordrecht.

[3] Tuy, H. (1995), *D. C. Optimization: Theory, Methods and Algorithms* //Handbook of Global Optimization/Ed. by Horst. R., Pardalos P.Kluwer Academic Publishers, 149–216.

[4] Tuy, H. (1987), *Convex programs with an additional reverse convex constraint*, Journal of Optimization Theory and Applications, 52, 463–485.

[5] Tuy, H. (1992), *The Complementary Convex Structure in Global Optimization*, Journal of Global Optimization, 2, 21–40.

[6] Bansal, P.P. and Jacobsen, S.E. (1975), *An algorithm for optimizing network flow capacity under economy of Scale*, Journal of Optimization Theory and Applications, 15, 565–586.

[7] Vasil'ev, F.P. (1988), *Numerical methods of extremal problems solving*, Nauka, Moscow (in russian).

[8] Strekalovsky, A.S. (2003), *Elements of Nonconvex Optimization*, Nauka, Novosibirsk (in russian).

[9] Strekalovsky, A.S. (1994), *Extremal problems on complements of convex sets*, Cybernetics and System Analysis, 1, Plenum Publishing Corporation, 88-100.

[10] Strekalovsky, A.S. (1993), *The search for a global maximum of a convex functional on an admissible set*, Comput. Math. and Math. Physics, 33, 315–328, Pergamon Press.

[11] Strekalovsky, A.S. (1998), *Global optimality conditions for nonconvex optimization*, Journal of Global Optimization, 12, 415–434.

[12] Strekalovsky, A.S. (1997), *On Global Optimality Conditions for D. C. Programming Problems*, Irkutsk University Press, Irkutsk.

[13] Strekalovsky, A.S., (2000), *One way to Construct a Global Search Algorithm for d.c. Minimization Problems* In: Pillo, G. Di., Giannessi, F. (eds.), *Nonlinear Optimization and Related Topics*, 36, Kluwer, Dordrecht, 429–443.

[14] Strekalovsky, A.S. and Tsevendorj, I. (1998), *Testing the $\Re$-strategy for a Reverse Convex Problem*, Journal of Global Optimization, 13, 61–74.

[15] Strekalovsky, A.S. and Kuznetsova, A.A. (1999), *The convergence of global search algorithm in the problem of convex maximization on admissible set*, Russian Mathematics (IzVUZ), 43, 70–77.

[16] Strekalovsky, A.S. and Yakovleva, T.V. (2004), *On a Local and Global Search Involved in Nonconvex Optimization Problems*, Automation and remote control, 65, 375–387.

[17] Strekalovsky, A.S. (2001), *Extremal Problems with D.C.-Constraints* Comp. Math. And Math. Physics, 41, 1742–1751.

[18] Strekalovsky, A.S., Kuznetsova, A.A. (2001) *On solving the Maximum Clique Problem*, Journal of Global Optimization, 21, 265–288.

[19] Strekalovsky, A.S., Kuznetsova, A.A., Tsevendorj, I. (1999) *An approach to the solution of integer optimization problem* Comp. Math. And Math. Physics, 39, 9–16.

Pruning a box from Baumann point in an Interval Global Optimization Algorithm*

Boglárka Tóth[1,†] and L.G. Casado[2]

[1] *Dpt. Statistics and Operations Research, University of Murcia, Spain,* boglarka@um.es

[2] *Dpt. of Architecture and Electronics, University of Almeria, Spain,* leo@ace.ual.es

Abstract This work is in the context of reliable global optimization algorithms, which use branch and bound techniques and interval arithmetic to obtain inclusion functions. The search region have to have edges parallel to the axes (box) in order to allow the use of interval arithmetic to obtain bounds of the objective function in that region. We study a method to determine the largest region inside the current box where the global minimum cannot exist, based on the gradient information. If the current box cannot be rejected completely, the removed region have to satisfy that the generated subproblems (that can contain the global minimum) have to have a box shape, in order to apply the branch and bound algorithm, but we are also interested in generate the smaller number of them to reduce the overall computational cost. The method is presented for two and n dimensional problems and numerical results show that is worth to incorporate it to interval global optimization algorithms.

1. Introduction

The context of our study is unconstrained global optimization. Thus, the general problem can be defined as $\min_{x\in S} f(x)$, where f is a continuously differentiable function defined over the n-dimensional interval $S \subset \mathbb{R}^n$, where the minimum have to be found.

In our context, real numbers are denoted by $x, y, \ldots$ and compact intervals by $X = [x^L, x^U], Y = [y^L, y^U], \ldots$, where $x^L = \min\{x \in X\}$ and $x^U = \max\{x \in X\}$ are the lower and upper bounds of X, respectively. The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a, b \in \mathbb{R},\ a \leq b\}$. The notation $x = (x_1, \ldots, x_n)^T,\ x_i \in \mathbb{R}$ and $X = (X_1, \ldots, X_n)^T,\ X_i \in \mathbb{I}\ (i = 1, \ldots, n)$ is used for real and interval vectors, respectively. The set of n-dimensional interval vectors (also called boxes) is denoted by $\mathbb{I}^n$.

Let $f : Y \subseteq \mathbb{R}^n \to \mathbb{R}$ be a continuous function, and $\mathbb{I}(Y) = \{X \mid X \in \mathbb{I}, X \subseteq Y\}$. The function $F : \mathbb{I}(Y) \subseteq \mathbb{I}^n \to \mathbb{I}$ is an *inclusion function* of f, if for every $X \in \mathbb{I}(Y)$ and $x \in X$, $f(x) \in F(X)$, i.e. $f(X) = \{f(x) \mid x \in X\} \subseteq F(X)$. This inclusion can be obtained, for instance, by Interval Arithmetic [2].

Let X be such a box that $x, c \in X$. If $G(X)$, i.e. an inclusion function of the gradient vector $g(X)$ is known, then the centered form is defined as $F_c(X) = F(c) + G(X)(X - c)$. Many times c is the midpoint of X, but it can be anywhere in X.

*This work has been partially supported by the Ministry of Education and Science of Spain through grant CICYT-TIC2002-00228.

[†] On leave from the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary.

2. The idea (in 2 dimensions)

Let us examine the visualization of the centered form for a box $X = (X_1, X_2)$. We suppose that $0 \in G(X)$, because otherwise the function is monotone in the box and it could be discarded or reduced. A "tent" can be drown from the point $(c, F^L(c))$ using the bounds of the derivatives (see Figure 1). It is easy to see that the function have to be above the tent. If we have a good upper bound on the global minimum $(\tilde{f})$ that is smaller than $F^L(c)$ we know that the minimum cannot be in the region defined by the intersection of the tent and the plane defined by $\tilde{f}$. This region is drawn in dark(red) in Figure 2, and will be called as the Pruneable Region (PR) in the rest of the paper. The use of this Pruneable region was discarded in [3] because PR is not parallel to the coordinate axes and the division of X to reject some region will generate a lot of boxes. We will show here that PR is still of interest.

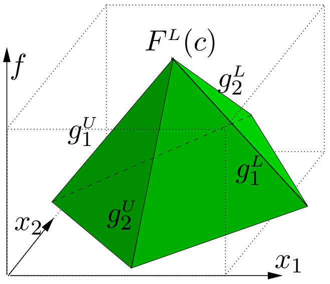


Figure 1. The tent.

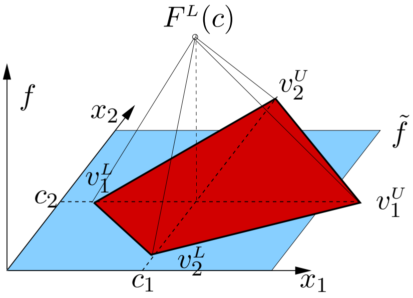


Figure 2. The section of the tent and the plane $\tilde{f}$.

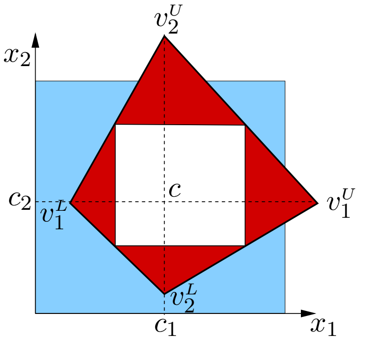


Figure 3. A possible pruning.

PR can be defined by its vertices:

$$v_1^U = \left\{(x_1, c_2) \mid \tilde{f} = F^L(c) + g_1^L(x_1 - c_1)\right\} = \left(c_1 + \left(\tilde{f} - F^L(c)\right)/g_1^L, c_2\right), \tag{1}$$

$$v_1^L = \left(c_1 + \left(\tilde{f} - F^L(c)\right) g_1^U, c_2\right), \quad v_2^I = \left(c_1, c_2 + \left(\tilde{f} - F^L(c)\right)/g_2^I\right), \quad I = L, U, \tag{2}$$

if $g_i^L \neq 0$ and $g_i^U \neq 0, i \in 1,2$, otherwise $v_i^L = -\infty$ or $v_i^U = \infty$ for the appropriate vertex as the limit of the fractions suggest.

Let us introduce the notation $pr_i^I = \left(\tilde{f} - F^L(c)\right)/g_i^I,\ I = L, U,\ i = 1,2$ to reduce the formulas. Then $v_1^L = (c_1 + pr_1^L, c_2)$, $v_1^U = (c_1 + pr_1^U, c_2)$ and $v_2^L = (c_1, c_2 + pr_2^L)$, $v_2^U = (c_1, c_2 + pr_2^U)$.

Because our algorithm works with boxes, we cannot use other shapes (different than boxes) to divide the non-rejected area. On the other hand, if we try to prune generating boxes, too many boxes can be generated, and/or only a small part of the pruneable region can be discarded. In general more than four generated subboxes is not desired.

One of the goals of the pruning can be to obtain the largest box to be removed from the original box. This can be done by computing the largest rectangle in the triangle defined by its vertices v_1^U, v_2^L, c. The area of the rectangle is $A = a \cdot b$, where $b = pr_2^U - pr_2^U/pr_1^U a$. Thus, it can be computed by maximizing A respect to a and b. The maximal area is $pr_1^U pr_2^U/4$ with $a = pr_1^U/2$ and $b = pr_2^U/2$. From the above result one can see, that the same can be obtained for all of the four triangles. Fortunately all the rectangles share one edge, thus the resulting box can be given by $c + ([pr_1^L/2, pr_1^U/2], [pr_2^L/2, pr_2^U/2])$.

To calculate the pruneable box, the only important thing is the intersection between PR and OB. Therefore, to obtain easier formulas let us change c to be the origin.

Thus, we introduce the following notation:

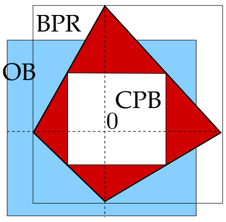


Figure 4. The new notation for the easier formulation.

$$\begin{aligned} &\text{OB (Original Box):} & OB &= X - c, \\ &\text{BPR (Box containing PR):} & BPR &= ([pr_1^L, pr_1^U], [pr_2^L, pr_2^U]), \\ &\text{PB (Pruneable Box):} & PB &= ([pb_1^L, pb_1^U], [pb_2^L, pb_2^U]), \end{aligned} \tag{3}$$

where BPR is the smallest box which contains the pruneable region (PR). In our new notation $CPB = ([pr_1^L/2, pr_1^U/2], [pr_2^L/2, pr_2^U/2])$, and the area of the CPB is

$$A(CPB) \;=\; \left(\frac{pr_1^U}{2} - \frac{pr_1^L}{2}\right)\left(\frac{pr_2^U}{2} - \frac{pr_2^L}{2}\right) = \frac{1}{4}(pr_1^U - pr_1^L)(pr_2^U - pr_2^L) = \frac{1}{4}A(BPR),$$

i.e. half of the area of PR, what is the half of the area of BPR.

2.1 Shifting CPB (Centered Pruneable Box)

As it can be seen in Figure 5, sometimes it is better to shift the CPB to the edge of OB. This can improve the method by reducing the number of the generated subboxes at the cost of the reduction of the rejected area compared to the area of CPB. To differentiate the centered pruneable box (CPB) from the shifted pruneable box, we notate the later as SPB. In the cases when $OB \cap CPB \neq CPB$, the area of SPB can be larger than the area of CPB (see the third case in Figure 5).

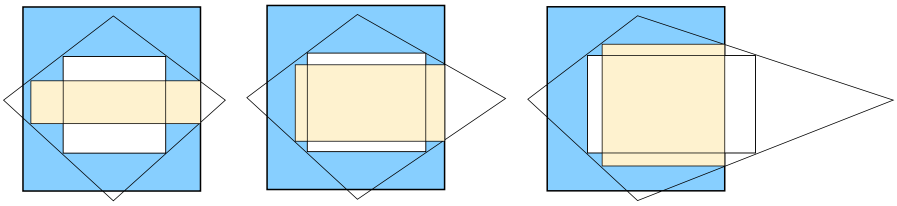


Figure 5. Shifting cases.

The areas of the boxes (supposing that the resulting SPB is inside of the original one (OB)) can be computed as:

$$\begin{aligned} A(SPB_{ob_2^U}) &= \left(pr_1^U\left(1 - \frac{ob_2^U}{pr_2^U}\right) - pr_1^L\left(1 - \frac{ob_2^U}{pr_2^U}\right)\right)\left(ob_2^U - pr_2^L\frac{ob_2^U}{pr_2^U}\right) \\ &= (pr_1^U - pr_1^L)\left(1 - \frac{ob_2^U}{pr_2^U}\right)\frac{ob_2^U}{pr_2^U}(pr_2^U - pr_2^L) = \left(1 - \frac{ob_2^U}{pr_2^U}\right)\frac{ob_2^U}{pr_2^U}A(BPR) \qquad (4) \\ A(SPB_{ob_i^I}) &= \left(1 - \frac{ob_i^I}{pr_i^I}\right)\frac{ob_i^I}{pr_i^I}A(BPR), \qquad i = 1, 2, I = L, U \qquad (5) \end{aligned}$$

If an SPB is inside the original box then the shifted area only depends on the value $\frac{ob_i^I}{pr_i^I}, i = 1, 2, I = L, U$, obtaining larger area if it is nearer to $1/2$. If it equals to $1/2$ we obtain the $A(CPB)$, what also means that $SPB_{ob_i^I} = CPB$.

These equations show, that knowing the vector $\left(\frac{ob_1^U}{pr_1^U}, \frac{ob_1^L}{pr_1^L}, \frac{ob_2^U}{pr_2^U}, \frac{ob_2^L}{pr_2^L}\right)$ one can choose the best Pruneable Box (PB), if these are inside the original box. The other cases do not differ too much, but those have to be treated differently. This would lead to a case analysis, that cannot be extended for multidimensional case. In case we use the Baumann point, it is guaranteed that the CPB is inside OB, therefore the case analysis can be avoided.

2.2 Simplification using the Baumann point

In this section we will see that the use of the Baumann point [1], in place of the center c, simplify our computations. For a given box $X = (X_1, X_2)$, if the enclosure of the derivative of f, i.e. $G(X) = ([g_1^L, g_1^U], [g_2^L, g_2^U])$ is given, the Baumann point b is defined as:

$$b_i = \begin{cases} \dfrac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L}, & \text{if } g_i^L < 0 < g_i^U \\ x_i^U & \text{if } g_i^U \leq 0 \\ x_i^L & \text{if } g_i^L \geq 0 \end{cases} \qquad i = 1, 2 \tag{6}$$

In the following theorem it is shown that using the Baumann point the different intersection cases between OB and PR can be avoided from the case analysis.

Theorem 1. *Suppose a given differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$, its inclusion function F and the inclusion function of its derivative $G(X) = ([g_1^L, g_1^U], [g_2^L, g_2^U])$ over a given box X. The pruneable region defined by its vertices as in (1-2) centered in the Baumann point (i.e. $c = b$) include all the corners of X, or none of them.*

Corollary 2. *From the fact, that $\frac{x_i^U - b_i}{pr_i^U} = \frac{x_i^L - b_i}{pr_i^L}$, $\forall i$, it can be deduced easily that the shifted pruneable box $SPB_{ob_i^U}$ is exactly the same as $SPB_{ob_i^L}$, thus it can be denoted as SPB_i. Therefore, in the shifted cases there are only two new generated boxes, and only the direction to shift have to be determined.*

Remark 3. *The above results suggest to denote for $i = 1, 2$ the $\frac{x_i^U - b_i}{pr_i^U} \left(= \frac{x_i^L - b_i}{pr_i^L}\right)$, i.e. $\frac{ob_i^U}{pr_i^U} \left(= \frac{ob_i^L}{pr_i^L}\right)$ values and the $1 - \frac{x_i^U - b_i}{pr_i^U}$, i.e. $1 - \frac{ob_i^U}{pr_i^U}$ values by sf_i (Shifting Factor) and osf_i (Opposite Shifting Factor), respectively. Therefore, $A(SPB_i) = sf_i osf_i A(BPR)$.*

Theorem 4. *If $CPB \not\subset OB$, then exists SPB such that $A(SPB) > A(CPB \cap OB)$.*

The advantages of the usage of the Baumann point are that it makes our computation easier, and it equilibrates the pruneable region above the box.

3. The n-dimensional case

Now we generalize the above results for multi-dimensional case. As in the two-dimensional case let us center the problem at the point c. Thus, we will use the same notation for the Original Box $OB = X - c$.

It is easy to see that, similarly to the two-dimensional case, but already re-centered in c, the vertices are:

$$v_i^L = (0, \dots, 0, pr_i^L, 0, \dots, 0), \qquad v_i^U = (0, \dots, 0, pr_i^U, 0, \dots, 0), \qquad i = 1, \dots, n, \tag{7}$$

where $pr_i^U = \frac{\tilde{f} - F^L(c)}{g_i^L}, pr_i^L = \frac{\tilde{f} - F^L(c)}{g_i^U}, i = 1, \dots, n.$

From (7) we know that in a 3-dimensional case the shape of PR is as in Figure 6. To see the properties of this body, we do a small geometrical evasive.

Definition 5. *An n-dimensional polytope is called* orthopeder, *if the diagonals intersect in one point and are orthogonal to each other.*

Proposition 6. *The PR defined by its vertices (7) is an orthopeder.*

Definition 7. *The* cross polytope *is the regular polytope in n dimensions corresponding to the convex hull of the points formed by permuting the coordinates $(\pm 1, 0, 0, \dots, 0)$.*

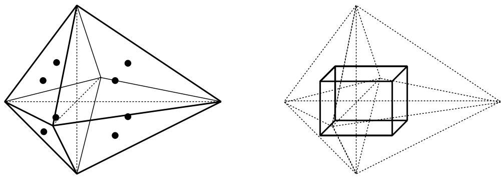


Figure 6. The shape of PR in 3-dimensional case.

Every orthopeder is a special scaled cross polytope, with different scaling on all direction. It can be seen, that the corresponding scaling vectors are PR^L and PR^U. The cross polytope is a dual of an n-dimensional hypercube, i.e. its vertices are the centers of the faces of the hypercube. As it is known in geometry, in the cross polytope the largest hyper-rectangle is its dual, i.e., the hypercube what has its vertices on the center of the faces of the cross polytope.

Proposition 8. *It is trivial that, in a scaled cross polytope, the largest hyper-rectangle is the rescaled hypercube with the same scalevector.*

Corollary 9. *The largest box in the orthopeder PR defined by its vertices (7) is $CPB = (CPB_1, \dots, CPB_n)$, where $CPB_i = [pr_i^L/n, pr_i^U/n]$.*

Corollary 10. *The volume of CPB is $V(CPB) = \prod_{i=1,\dots,n} \frac{pr_i^U - pr_i^L}{n} = \frac{1}{n^n} V(BPR)$.*

Theorem 11. *Suppose a given differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, its inclusion function F and the inclusion function of its derivative $G(X) = ([g_1^L, g_1^U], \dots, [g_n^L, g_n^U])$ over a given box X. The pruneable region defined by its vertices as in (7), centered in the Baumann point (i.e. $c = b$), include all the corners of X, or none of them.*

4. Shifting the n-dimensional CPB

When CPB is inside of OB, the number of generated boxes pruning CPB is $2n$. If $n > 3$ this is much more than desired. Shifting in k directions reduces the number of new boxes to $2(n-k)$, therefore it is important to find the best directions to shift.

To choose the best PB we are going to construct a simple and powerful method that determine the SPB with the *largest volume ratio/less new boxes* index. We already know that using the Baumann point the equations $\frac{ob_i^U}{pr_i^U} = \frac{ob_i^L}{pr_i^L}$ hold for $i = 1, \dots, n$. In the following formulas these values would appear many times. Thus let us introduce the notation as in the two-dimensional case for the Shifting Factor, i.e. $sf_i = \frac{ob_i^U}{pr_i^U} = \frac{ob_i^L}{pr_i^L}, \quad i = 1, \dots, n$.

Let us formulate only the general shifting form. For instance, let shift in the $j_1, \dots, j_k \in \{1, \dots, n\}$ dimensions. First, let us generalize the notation of the Opposite Shifting Factor for this case as $osf_{j_1,\dots,j_k} = 1 - \sum_{i=j_1,\dots j_k} sf_i$. The shifting is possible only if $osf_{j_1,\dots,j_k} > 0$. After some computation we can obtain the coordinates of the $SPB_{j_1,\dots,j_k}$ as:

$$spb_i^I = \frac{pr_i^I}{n-k}\, osf_{j_1,\dots,j_k}, \qquad i = 1, \dots, n \quad I = L, U.$$

Now we can obtain the volume, when $SPB_{j_1,\dots,j_k}$ is inside of OB:

$$\begin{aligned} V(SPB_{j_1,\dots,j_k}) &= \prod_{i=j_1,\dots j_k} (ob_i^U - ob_i^L) \prod_{\substack{i=1,\dots,n \\ i \neq j_1,\dots,j_k}} \left(\frac{pr_i^U}{n-k}\, osf_{j_1,\dots,j_k} - \frac{pr_i^L}{n-k}\, osf_{j_1,\dots,j_k} \right) \\ &= \frac{n^n}{(n-k)^{n-k}} \left(\prod_{i=j_1,\dots,j_k} sf_i \right) osf_{j_1,\dots,j_k}^{n-k}\, V(CPB) \end{aligned}$$

One can see, that the nearer a sf_j to $1/n$, the grater the volume is. It also implies that knowing the vector $(sf_1, \ldots, sf_n)$ we can choose the best directions to shift.

Let us note, that if we have the volume for the SPB shifted in the dimensions $j_1, \ldots, j_{k-1} \in \{1, \ldots, n\}$, and we want to shift it in one more dimension, $j_k \in \{1, \ldots, n\}$ too, the new volume can be computed from the known $V(SPB_{j_1,\ldots,j_{k-1}})$ in the following way:

$$V(SPB_{j_1,\ldots,j_k}) \quad = \quad sf_{j_k} \left(\frac{n-k+1}{n-k}\right)^{n-k} \left(\frac{osf_{j_1,\ldots,j_{k-1}} - sf_{j_k}}{osf_{j_1,\ldots,j_{k-1}}}\right)^{n-k} \frac{n-k+1}{osf_{j_1,\ldots,j_{k-1}}} V(SPB_{j_1,\ldots,j_{k-1}})$$

Similarly to Theorem 4 it can be shown that if CPB is not inside OB we can always show an SPB with larger volume inside OB. Although in the next theorem we state a bit more.

Theorem 12. *Suppose $CPB \not\subset OB$ in the dimensions $j_1, \ldots, j_k \in \{1, \ldots, n\}, (k < n)$, i.e. $ob_i^U < cpb_i^U$ and $ob_i^L > cpb_i^L$ for all $i \in \{j_1, \ldots, j_k\}$, but not for $i \in \{1, \ldots, n\} \setminus \{j_1, \ldots, j_k\}$. Then $SPB_{j_1,\ldots,j_k}$ is such that $V(SPB_{j_1,\ldots,j_k}) > V(CPB \cap OB)$, and $V(SPB_{j_1,\ldots,j_k}) > V(SPB_{j_1,\ldots,j_l} \cap OB), l < k$.*

5. Results and conclusions

The presented pruning method was included in a general Interval Global Optimization algorithm which use centered form and naiv interval arithmetic as inclusion functions and monotonicity test, for some hard test problems. Table 1 shows the effort of the algorithm without and with the new pruning method which let to cut the pruneable box with the best *pruning rate = volume ratio/new boxes* rate. The smallest allowed pruning rate was 0.035. One can see, that the pruning method works for multi-dimensional cases, and it is worth to incorporate it to Interval Global Optimization Algorithms.

Table 1. Numerical results for the use of the new pruning method compared to the general B&B algorithm.

Function name	n	ε	Effort	Effort With Pruning	SpeedUp
Ratz	3	10^{-3}	633508	535196	1.18
Kowalik	4	10^{-3}	4883230	2125330	2.30
EX2	5	10^{-3}	3448563	1970996	1.75
Ratz	5	10^{-3}	1201476	1035922	1.16
Ratz	7	10^{-3}	1900512	1667050	1.14
Ratz	9	10^{-3}	3183190	2435704	1.31
Griewank	10	10^{-3}	1551366	1551366	1.00
Rastrigin	10	10^{-3}	1551366	1593770	0.97
Rosenbrock	10	10^{-3}	510041	335433	1.52

$\#F$: number of function evaluation, $\#G$: number of gradient evaluation, Effort= $\#F + n\#G$

References

[1] E. Baumann. Optimal centered forms. *BIT*, 28(1):80–87, 1988.

[2] R.E. Moore. *Interval analysis.* Prentice-Hall, New Jersey, (USA), 1966.

[3] D. Ratz. *Automatic Slope Computation and its Application in Nonsmooth Global Optimization.* Shaker Verlag, Aachen, Germany, 1998.

Proceedings of GO 2005, pp. 247 – 252.

A Hybrid Multi-Agent Collaborative Search Applied to the Solution of Space Mission Design Problems

Massimiliano Vasile

Department of Aerospace Engineering, Politecnico di Milano, via La Masa 34, 20156, Milan, Italy, vasile@aero.polimi.it

Abstract In this paper a novel algorithm is proposed for the solution of single and multiple objective problems in space mission analysis and design. The underlying idea is to look for the set collecting all the solutions that satisfy a given optimality criterion or property. A population of agents is associated to a set of general solutions and behavioural rules are devised for each agent and for the entire population in order to drive the agents toward the optimal set. This mechanism is hybridised with a particular adaptive domain decomposition technique, which is used either to extend the search or to favour convergence. Some example cases will show the effectiveness of the proposed approach.

Keywords: global optimisation, multiobjective optimisation, hybrid methods, space mission design.

1. Introduction

In many space mission design problems there is the need to extensively search for optimal solutions in highly complex solution domains. Typical problems of trajectory design, optimal control, multidisciplinary design and concurrent engineering can be translated into multiobjective optimisation problems characterised by multimodal functions and heterogeneous parameters (mixed integer-real parameters) [1,2]. Moreover it is often required to procure reliable sets of solutions, where reliability is measured in terms of repeatability and predictability of the result. This would suggest a purely deterministic approach to many problems occurring in mission analysis. In turn the request for exhaustiveness of the search leads often to long computational time (few days up to a week). Although this is generally acceptable compared to standard development time of a space mission, it might become not acceptable in the early or preliminary phase of the design or in case of extremely costly evaluations (as in multidisciplinary design). An alternative to deterministic search could be to use stochastic based approaches, especially when the solution space is deceiving. In this case predictability of the algorithm has to be dropped. In particular in most stochastic algorithms the width of the intervals for each parameter is not always easy to define although it can change the probability to find the global optimum. Moreover the need for sets of optimal solutions, rather than a single global optimum, requires a reformulation of single objective problems in order to allow the convergence to a set as for multiobjective problems. In this paper a unifying formulation of both multiobjective and single objective problems is proposed and an algorithm that combines a deterministic domain decomposition technique and a stochastic-based multiagent search is presented. This algorithm is intended to improve performances and reliability of optimisation in some space mission design problems by allowing a partial or complete reconstruction of the set of optimal solutions. The domain decomposition procedure generates branches along which the multiagent collaborative search is performed. Non promising branches are then pruned while promising ones are further explored until termination. The branching scheme is initially inferred and then updated on the basis of the outcome of the multiagent exploration. The proposed approach is here tested at first on a well known suite of multiobjective optimisation problems and compared to known algorithms and then on one standard mission analysis problems and compared to some state of the art global optimisation tools either based on deterministic or stochastic methods

2. Problem Formulation

In many practical cases both single and multiobjective problems require the identification of multiple optimal solutions and therefore to reconstruct a set of values and not just a single one. Therefore the general problem, no matter if single or multiple objective, could be to find a set X of feasible solutions $\mathbf{x}$ such that a property $P(\mathbf{x})$ is true for all $\mathbf{x} \in X \subseteq D$:

$$X = \{\mathbf{x} \in D \mid P(\mathbf{x})\} \tag{1}$$

where the domain D is a hyperectangle defined by the upper and lower bounds on the components of the vector $\mathbf{x}$:

$$D = \{x_i \mid x_i \in [b_i^l, b_i^u] \subseteq \Re, i = 1, \ldots, n\} \tag{2}$$

All the solutions satisfying property P are here defined to be optimal with respect to P or P-optimal and X can be said to be a P-optimal set. Now property P might not identify a unique set therefore a global optimal set X_o can be defined such that all the elements of X_o dominate the elements of any other X:

$$X_o = \{\mathbf{x}^* \in D \mid P(\mathbf{x}^*) \wedge \forall \mathbf{x} \in X \Rightarrow \mathbf{x}^* \prec \mathbf{x}\} \tag{3}$$

where $\prec$ represents the dominance of the $\mathbf{x}^*$ solution over the $\mathbf{x}$ solution. In the case of single objective function, the set X may contain all solutions that are local minimisers or are below a given value. In this case if more than one solution exists within the required domain D the interest could be more to find a number of solutions forming the set X, rather than finding the global optimum with a high level of accuracy. In the case of multiobjective optimization, if P is a dominance condition or Pareto optimality condition for the solution $\mathbf{x}$ then the solution is Pareto-optimal if $P(\mathbf{x})$ is true.

3. Multiagent Collaborative Search

The proposed multiagent collaborative search is based on the following principle: each one of a set of agents explores locally the solution space within a hypercube (local environment perception), at the end of each exploration session agents showing improvements communicate (collaborate) with the others, their findings. A pool of embryonic agents is maintained in order to randomly generate new agents. A filter is used to select exploring and embryonic agents: if an agent falls out of the filter, is inserted in the pool.

Each solution $\mathbf{x}$ is associated to an agent j and is represented by a string, of length n, containing in the first m components integer values and in the remaining $n - m$ components real values. This particular encoding allows the treatment of problems with a mixed integer-real data structure. A hypercube S is associated to each agent $\mathbf{x}_j$, the hypercube, enclosing a region of the solution space surrounding the individual, is defined by a set of intervals $S = S_1 x S_2 x S_n \subseteq D_l$, where $x_i \in S_i$. The solution space is then explored locally by acquiring information about the landscape within each region S and globally using a population of agents. A sequence of actions is then performed by each agent $\mathbf{x}_j$ according to a behavioural scheme β_j, in order to acquire a minimum set of samples sufficient to decide in which direction to take the next move. For an agent $\mathbf{x}_j$, a behavioural scheme is a collection of displacement vectors $\Delta\xi_s$ generated by a function f_β :

$$\beta_j = \{\Delta\xi_s \mid \forall s \in N \Rightarrow \mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \Delta\xi_s \in S \subseteq D \wedge \Delta\xi_s = f(\mathbf{x}_i^k, \mathbf{x}_i^{k-1}, \mathbf{w}, r, \mathbf{x}, s) \wedge s \leq s_{max}\} \quad (4)$$

where f_β is a function of the current and past agent state $\mathbf{x}_j^k$ and $\mathbf{x}_j^{k-1}$, of a set of weights $\mathbf{w}$, a set of random numbers $\mathbf{r}$, the current state of the other agents in the population $\mathbf{x}$ and the index s. The index s is increased until a better solution is found and s is not larger than a given s_{max}. The behaviour scheme is conceptually equivalent to the pattern of common pattern search direct methods. The size of the hyperrectangle S and the maximum number of available samples (level of resources) are adapted according to the number of discoveries of each agent.

3.1 Knowledge Sharing and the Global Archive

At every stage of the optimisation process, a number of solutions (i.e. of agents) belongs to an optimal set X. This set is saved at every stage into a global archive which represents also the repository of the best achieved and perceived locations of each agent. After the behavioural scheme has been applied, all the agents presenting an improvement are inserted in a communication list and communication is performed between each element of the list and an equal number of agents randomly selected form the population. Moreover the elements of X are ranked according to their crowding factor and used to define the behavioural scheme of dominated agents and of agents presenting no improvement. The crowding factor is the number of agents in the hypercube S divided by the number of agents in the population.

4. Domain Decomposition

Before each run of MACS the solution space is decomposed according to a predictive scheme. A branching model is initially provided by the user and later updated using the MACS output. During MACS the subdomains generated by the branching scheme are used to evolve niches in particular highly crowded agents are regenerated into subdomains with a low density.

The initial domain D is progressively decomposed into smaller domains $D_l \subseteq D$ according to the branching scheme. The branching scheme is represented by a set I_s containing the indices of the coordinates that have to be split and a set C_B containing the cutting points for each coordinate. The initial set I_s is defined by the user while the set C_B contains the middle point of the interval defining each coordinate. After each run of MACS the branching scheme is adapted depending on the outcome of the MACS. The subdomains D_l with the minimum number of collected samples among the ones containing elements of X is selected for further decomposition provided that the number of times n_b its parent subdomains have been branched without improvement is below a given threshold. This strategy however excludes from further exploration all the subdomains containing no elements of X. Therefore when an exhaustive search is required, the following merit function is used:

$$\psi_{D_l} = (1 - \upsilon)\varpi_{D_l} + \upsilon\varphi_{D_l} \quad (5)$$

where ϖ_{D_l} is the density of function evaluations in D_l,φ_{D_l} is best fitness in D_l and υ is a weighting factor used to favor either convergence or exploration. For multiobjective problems, in the following, we used the former strategy.

5. Sample Multiobjective Problems

The proposed hybrid approach (EPIC) was tested on several different test cases: a standard set of multiobjective problems and some space related problems. Multiobjective optimisation problems were taken from [4] and [3] and listed in Tab. 1. EPIC was compared to MPSO, NSGA-II and PAES in terms of number of function evaluations and of average distance from the optimal Pareto set (metric M_1 in [3]).Test function 1 is one-dimensional and has a disconnected Pareto set made of 2 subsets. EPIC was run with 5 exploring agents and a total population of 10 agents. The number of function evaluations was fixed to 600. Test function 2 has a disconnected Pareto set made of 4 subsets. In this example, 5 exploring agents were used over a total number of 10 agents in the population. The total number of function evaluations was fixed to 3000. Test function 3 has 60 local Pareto fronts for $q = 1$ and $n = 2$ while has 21^9 local Pareto fronts with $n = 10$ and $q = 2$. The maximum number of function evaluations was fixed to 3000 for $n = 2$ and to 20000 for $n = 10$ and the number of exploring agents was 3 over a population of 5.

Table 1. Multiobjective test functions

Scha	$f_2 = (x-5)^2\ ; f_1 = \begin{cases} -x & if \quad x \leq 1 \\ -2+x & if \quad 1 < x < 3 \\ 4-x & if \quad 3 < x \leq 4 \\ -4+x & if \quad x > 4 \end{cases}$	$x \in [-5, 10]$
Deb	$f_1 = x_1$ $f_2 = (1+10x_2)\left[1 - \left(\frac{x_1}{1+10x_2}\right)^{\alpha} - \frac{x_1}{1+10x_2} sin(2\pi q x_1)\right]$	$x_1, x_2 \in [0, 1]$ $\alpha = 2;. q = 4$
T4	$g = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10cos(2\pi q x_i)]; h = 1 - \sqrt{\frac{f_1}{g}}$ $f_1 = x_1; f_2 = gh$	$x_1 \in [0, 1]; x_i \in [-a, a]$ $i = 2, \ldots, n$

The objective space for a typical run of EPIC is represented in Fig. 1 for all the three test functions. The results for the performance index M_1 averaged over 30 runs are summarised in Tab. 2 and compared to MPSO, PAES and NSGA-II (the standard deviation is reported in brackets). In [4] the three algorithms were run for a maximum of 4000, 1200 and 3200 function evaluations respectively on case *Deb*, *Scha* and *T4* with $q = 1$,$n = 2$ and $a = 30$. Then the average value μ_{M_1}=1.542e-3 and the standard deviation σ_{M_1}=5.19e-4 of M_1 over 20 runs of EPIC on problem *T4* with $q = 2$,$n = 10$ and $a = 5$ was compared to the results in [5] for NSGA-II(μ_{M_1}=0.513053 σ_{M_1} =0.118460),SPEA (μ_{M_1}=7.340299 σ_{M_1} =6.572516)and PAES (μ_{M_1}=0.854816 σ_{M_1} =0.527238). In should be underlined that, for this test function, in all the 20 runs, EPIC converged to the global Pareto front.

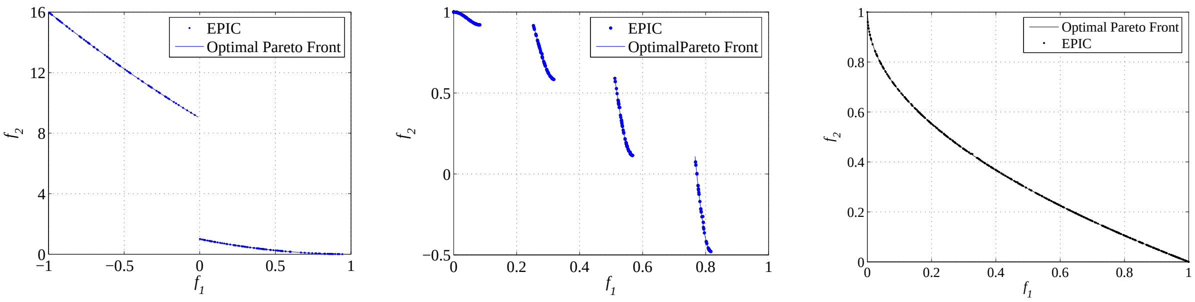


Figure 1. Pareto fronts for the three test functions, *Scha* on the left, *Deb* in the middle, *T4* on the right.

5.1 Multi Gravity Assist Trajectories

A common problem in space mission analysis is the optimal design of transfer trajectories exploiting one or more gravity assist manoeuvres (MGA) to change the orbital parameters of a spacecraft. Each gravity manoeuvre occurs at a planet and exploits the gravity action of the

Table 2. Summarising results for multiobjective test cases

	MPSO	*PAES*	*NSGA-II*	*EPIC*
Deb	0.002057 (0.000286)	0.002881 (0.00213)	0.002536 (0.000138)	5.4179e-4 (1.7e-4)
Scha	0.00147396 (0.00020178)	0.070003 (0.158081)	0.001594 (0.000122)	1.4567e-4 (3.61e-4)
T4 (n=2,q=1,a=30)	0.0011611 (0.0007205)	0.259664 (0.573286)	0.094644 (0.117608)	2.8527e-4 (8.38e-4)

planet to produce a change in the velocity vector of the spacecraft, a so called Δv. Since Δv changes are normally produced by propelled manoeuvres, optimal sequences of gravity assist manoeuvres minimise the total propelled Δv required to reach a given target. A general gravity assist manoeuvre in the solar system can be modelled assuming the planet a point mass and the manoeuvre to be instantaneous. In this simplified model the gravity action produces a simple rotation of the incoming velocity vector, relative to the planet, in the plane identified by the incoming and the outgoing velocity vectors. This rotation is normally a function of the modulus of the incoming velocity and of the minimum distance of the spacecraft relative to the planet [6]. Due to this functional dependency, caused by physical reasons, not all the desired rotations can be achieved. Therefore an additional propelled manoeuvre is generally necessary to correct the outgoing velocity vector. The total cost of a MGA transfer can be defined as the sum of all the corrective v_i^c manoeuvres for all the N planetary encounters plus the initial v_0 at launch from Earth.

$$f(x) = \Delta v_0 + \sum_{i=1}^{N} \Delta v_i^c \tag{6}$$

The conic arc connecting two subsequent planets i and $i+1$ at position r_i and r_{i+1} is computed as a Lambert's [7] solution. The two position vectors are functions of the encounter dates t_i and t_{i+1}. Therefore the solution vector $x = [t_0, T_1, ..., T_i, ..., T_N]$ is defined by the launch date t_0 and by all the times of flight (TOF) from one planet to the subsequent one $T_i = t_{i+1} - t_i$.

The results for a transfer to Saturn through a multiple gravity assist of Venus and the Earth is reported in Tab.4. The sequence of planets is fixed and has been taken equal to that of the Cassini mission to Saturn, i.e. Earth-Venus-Venus-Earth-Jupiter-Saturn, the boundaries of the domain D for this problem are reported in Tab. 3. The Δv column in Tab. 4 shows the average value of the solutions found over 10 runs of the algorithms listed in the first column, along with the standard deviation σ. The last column reports the mean value of the total number of function evaluations for each algorithm with the associated standard deviation. EPIC has been compared to 10 algorithms, 7 based on stochastic processes: GAOT and GAOT-s (respectively real coded Genetic Algorithms and Genetic Algorithms with sharing), GATBX and GATBX-mig (different forms of Genetic Algorithms), FEP (fast evolutionary programming), DVEC (differential evolution), ASA (fast simulated annealing). The last three are deterministic algorithms: GlbSolve (implementation of the branch and prune strategy based on DIRECT), MCS (multilevel coordinate search developed by Neumaier et al.), and RbfSolve (response surface based algorithm), read [2] for additional details on the algorithms. For the stochastic algorithms the stopping criterion is the number of function evaluations after which no improvement of the objective is registered. For EPIC the stopping criterion is approximately the number of function evaluations of the best performing algorithm. A total of 20 exploring agents, over a population of 30, have been used and the maximum branching level n_b has been fixed to 1.

Table 3. Solution domain D for the Earth-Saturn MGA transfer

	t_0(MJD2000)	T_1 (day)	T_2 (day)	T_3 (day)	T_4 (day)	T_5 (day)
b_l	-1277	14	11	14	99	365
b_u	548	292	449	146	1000	3650

Table 4. Summarising results for the MGA test case

Software Tool	Δv(m/s)	*Function Evaluations*
GAOT	8256.416 (σ =1555.107)	8543.4 (σ =4075.382)
GAOT-s	21874.731 (σ =5741.406)	1350.4 (σ =559.057)
GATBX	8317.45 (σ =2339.832)	39468 (σ =29981.599)
GATBX-m	8237.81 (σ =972.517)	59220 (σ =27105.666)
FEP	9287.112 (σ =2860.194)	22238.3 (σ =16233.713)
DVEC	10145.388 (σ =3494.605)	10250 (σ =4696.157)
ASA	12712.987 (σ =6646.187)	96255.8 (σ =3281.118)
MCS	13782.954	46601
GlbSolve	15347.899	4345
RbSolve	16970.001	1000
EPIC	7133.900 (σ = 431.79)	10127 (σ =115.9)

6. Summary

In this paper a novel algorithm for single and multiple objective optimisation has been presented. The algorithm blends together a deterministic branching technique with a stochastic based multiagent collaborative search. This particular hybridisation is used to partially reconstruct a set of solutions, which are optimal with respect to a given criterion. This formulation of the problem allows a unifying prospective on both single and multiobjective problems. The effectiveness of the proposed approach has been demonstrated through a suite of standard well known mutliobjective optimisation problems and on common space mission analysis problems. The hybridisation of domain decomposition and MACS has performed quite well on all the test cases showing a significant gain in accuracy along with a reduction of function evaluations in the solution of both multiobjective and single objective problems.

References

[1] Vasile M., Summerer, L., De Pascale, P., "Design of Earth-Mars Transfer Trajectories using evolutionary branching technique", Acta Astronautica Volume 56, Issue 8 , April 2005, Pages 705-720

[2] Di Lizia P., Radice G., "Advanced Global Optimization Tools for Mission Analysis and Design" Final Report of ESA Ariadna ITT AO4532,Call 03/4101, 2004

[3] Zitzler E., Deb K., , Thiele L., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", Evolutionary Computation,8(2):173-195, Summer 2000

[4] Coello Coello C. A., Lechuga M. S., "MOPSO: a proposal for multiple objective particle swarm optimization", Mexico , 2001.

[5] Deb K., Pratap A., Agarwal S. and Meyarivan T., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II", KanGAL Report No. 200001.

[6] Labunsky A.V., Papkov O.V., Sukhanov K.G., Multiple Gravity Assist Interplanetary Trajectories, ESI Book Series,1998.

[7] Battin R., "An Introduction to the Mathematics and Methods of Astrodynamics", 1st ed., AIAA, New York, 1987.

Proceedings of GO 2005, pp. 253 – 255.

Improved lower bounds for optimization problems related to atom clusters*

Tamás Vinkó[1] and Arnold Neumaier[2]

[1]*Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary* tvinko@inf.u-szeged.hu

[2]*Institut für Mathematik, Universität Wien, Nordbergstrasse 15, A-1090 Wien, Austria* Arnold.Neumaier@univie.ac.at

Abstract A general method is introduced to obtain size independent lower bounds for the minimal interparticle distances in optimal atom cluster problems with pair potential functions. For the considered pair potential function only reasonable properties are supposed. Derivation of explicit linear lower bounds on the optimal energy values are also given. As demonstration new lower bounds for the Lennard-Jones and Morse clusters are reported.

Keywords: Lower bounds, atom cluster problems, Lennard-Jones cluster, Morse cluster.

1. Introduction

Given a cluster of n atoms, define $x_i \in \mathbb{R}^d$ $(i = 1, \dots, n$, and $d > 1)$ as the center of the ith atom. The potential energy of the cluster $x = (x_1, \dots, x_n) \in \mathbb{R}^{dn}$ is defined as the sum of the two-body inter-particle pair potentials over all of the pairs, i.e.,

$$E(x) = \sum_{i<j} v(r_{ij}), \tag{1}$$

where $r_{ij} = \|x_i - x_j\|$ and $v(r)$ is the value of a pair potential of distance r.

The aim of this work is to obtain lower bounds for the minimal interatomic distance in the optimal structure of (1) under weak, natural conditions on the pair potential $v(r)$.

A global minimizer of the function E is any configuration $x^* \in \mathbb{R}^{dn}$ with

$$E^* := E(x^*) = \min_{x \in \mathbb{R}^{dn}} E(x). \tag{2}$$

Let r_{ij} be the Euclidean distance of the points x_i^* and x_j^* $(i, j = 1, \dots, n)$. The minimal interparticle distance in the optimal structure is $r^* = \min_{i,j} r_{ij} \quad (i, j = 1, \dots, n)$. Without loss of generality let us suppose that $x_1 = 0$ and $0 = r_1 < r_2 \leq \ldots \leq r_n$, where

$$r_j = \|x_j - x_1\| = \|x_j\| \quad (j = 1, \dots, n).$$

We consider only the cases $n > 2$.

2. Improved lower bounds

We consider the particles in the cluster as balls and based on geometrial properties a lower bound on the largest distance (measured between the origin and the balls) is given.

*This work has been supported by the grants OTKA T 034350 and AÖU 56oü11.

Lemma 1. *Suppose that we have k small d-dimensional balls with radius $r/2$. Let r_i be the distance of the ball i from the origin and suppose that $r_1 \leq \ldots \leq r_k$. Then a lower for the maximal distance r_k of a small ball from the origin is $\frac{\sqrt[d]{k}-1}{2} r \leq r_k$.*

Theorem 2. *Let $E_i(x^*) := \sum_{i \neq j} v(\|x_i^* - x_j^*\|) \quad (i = 1, \ldots, n)$ and assume that in the configuration taken into account the minimal distance between the particles equals to q.*
(i) $E_1 < -|v(s)|$ holds.
(ii) $E_1^* = \sum_{r_j \leq s} v(r_j) + \sum_{r_j > s} v(r_j) \geq F(q) + S(q)$.
(iii) If the inequality $F(y) + S(y) > -|v(s)|$ holds for all $y \in Y$ then $r^* \notin Y$, where

$$F(y) := v(y) + v(s) \left(\left(\frac{2s+y}{y} \right)^d - 1 \right) \quad \textit{and} \quad S(y) := \sum_{j = \lceil (2s/y+1)^d \rceil}^{\infty} v\left(\frac{\sqrt[d]{j}-1}{2} y \right).$$

If we have a size independent lower bound on the minimal inter-particle distance then lower bound on the values E_i^* for all $i = 1, \ldots, n$ can be established. Thus linear (in the number of atoms) lower bound on the optimal values can be given as well [6].

3. Lennard-Jones clusters

In general form the Lennard-Jones pair potential function is $v_{\sigma,\epsilon}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$, where ϵ is the pair well depth and $2^{1/6}\sigma$ is the pair separation at equilibrium. In the global optimization literature the function $v_{\sigma,\epsilon}$ with reduced units, i.e. $\epsilon = \sigma = 1$ and $s = 2^{1/6}$, or the so-called scaled Lennard-Jones pair potential ($\epsilon = 1$, $\sigma = 2^{-1/6}$, $s = 1$) is investigated.

In the literature one can find previous results about the minimal distance in optimal scaled Lennard-Jones clusters. These results are compared in the following table including that one obtained in the present work. Note that all these results are independent of the number of particles in the configuration.

dimension	*Xue [6]*	*Blanc [1]*	*Vinkó [5]*	*present work*
2	–	0.7286	(0.7284)	0.7533
3	0.5	0.6108	0.6187	0.6536

Based on these results linear lower bounds on the optimal values can be calculated. They are $-67.88673405n \cdot \epsilon$ in dimension three and $-9.565562565n \cdot \epsilon$ in dimension two (n=2, 3, ...).

Note that for the scaled Lennard-Jones cluster even better lower bound ($q \geq 0.72997$) is reported by Huang. This result is not included in the table above since one can prove that his argument leading to such a good result is incorrect.

4. Morse clusters

The pair potential function in Morse cluster is $v_\rho(r) = e^{\rho(1-r)} \left(e^{\rho(1-r)} - 2 \right)$, where $\rho > 0$ is a parameter. In the context of global optimization, the cases $\rho > 6$ are interesting, since these are more difficult problems than finding the optimal Lennard-Jones structures [2].

We must emphasize that Theorem 2 gives an exclusion interval for r^*. For the Morse clusters (because it is defined even in the case $r = 0$) the function $F(y) + S(y) + |v_\rho(s)|$ in Theorem 2 becomes negative for small y values. The very tiny y values are handled using [4].

In Table 1 the results from [4] and form [5] collected and compared with the results can be achieved with the usage of the general technique introduced in this paper. Note that the new

Table 1. Lower bounds for the minimal distances in optimal three dimensional Morse clusters for different ρ. The last column contains linear lower bounds for the optimal values.

ρ	*Locatelli-Schoen [4]*	*Vinkó [5]*	*present work*	lower bound on the optimal value
6	0.114	0.499	0.559	$-81.23699155n$
7	0.376	0.611	0.652	$-51.83256650n$
8	0.468	0.680	0.710	$-40.34618808n$
9	0.528	0.727	0.752	$-33.74384634n$
10	0.574	0.762	0.782	$-29.76671162n$
11	0.613	0.789	0.807	$-26.58857202n$
12	0.644	0.810	0.826	$-24.62136302n$
13	0.672	0.828	0.841	$-23.40692214n$
14	0.695	0.842	0.854	$-22.11827825n$
15	0.715	0.855	0.865	$-21.19287270n$

results are achieved using the fact that by [4] q must be greater than the second column in Table 1.

5. Usefulness of the results

The information on the minimal interatomic distance can be applied

- in Branch-and-Bound methods as an accelerating tool;
- in a starting point generator used in incomplete or asymptotically complete global solvers. For instance, in [3] this kind of information is used to improve the performace of the proposed solving technique;
- and –as it is proved in [7]– one can construct efficient data structure to accelerate the computation of the potential function. Surprisingly, the value of the potential function (1) can be computed in $\mathcal{O}(n)$ time (with a naive method we have $\mathcal{O}(n^2)$ time complexity).

The lower bound for the global minimum can be used in Branch-and-Bound methods as a cut-off test.

References

[1] X. Blanc. Lower bounds for the interatomic distance in Lennard-Jones clusters. *Computational Optimization and Applications* 29:5-12,2004.

[2] J.P.K. Doye, R.H. Leary, M. Locatelli and F. Schoen. The global optimization of Morse clusters by potential energy transformations. *INFORMS Journal On Computing*, 16:371-379, 2004.

[3] M. Locatelli and F. Schoen. Fast global optimization of difficult Lennard-Jones clusters *Computational Optimization and Applications*, 21:55-70, 2002.

[4] M. Locatelli and F. Schoen. Minimal interatomic distance in Morse-clusters. *Journal of Global Optimization* 22:175-190, 2002.

[5] T. Vinkó. Minimal inter-particle distance in atom clusters. *To appear in Acta Cybernetica*. 2005.

[6] G.L. Xue. Minimum inter-particle distance at global minimizers of Lennard-Jones clusters. *Journal of Global Optimization*, 11:83-90, 1997.

[7] G.L. Xue. An $O(n)$ time hierarchical tree algorithm for computing force field in n-body simulations. *Theoretical Computer Science*, 197:157-169, 1998.

Global optimisation applied to pig nutrition

Graham R. Wood[1], Duangdaw Sirisatien[1], and Patrick C.H. Morel[2]

[1] *Department of Statistics, Macquarie University, NSW 2109, Australia,* gwood@efs.mq.edu.au, dsirisat@efs.mq.edu.au

[2] *Institute of Food, Nutrition and Human Health, Massey University, Palmerston North, New Zealand,* P.C.Morel@massey.ac.nz

Abstract Global optimisation techniques can be used to determine how to feed animals in order to maximise profitability, so extending the way in which linear programming has been used to find minimum cost diets. Keys to the extension are recently developed animal growth models and algorithms for nonlinear optimisation. The problem is described, a solution is presented and the nature of the objective function is explored.

Keywords: Feed cost, genetic algorithm, gross margin, pig genotype, weaner cost

1. Introduction

Efficient animal production is of critical importance on our increasingly finite planet. For many decades, linear programming has been used to determine minimum cost animal diets, based on a range of feedstuffs, their cost, their composition and dietary constraints. With the advent of animal growth models (see, for example [2]) and efficient nonlinear optimisation algorithms, it is now possible to extend this traditional use of optimisation to determine a feeding schedule which maximises profitablity [1]. The purpose of this paper is to describe, from a global optimisation perspective, how this is carried out.

The main findings to date of the research are i) that efficient optimisation methods are needed to find practical and useful feeding schedules and ii) that optimal feeding schedules can differ from the “feed-to-lean" schedules [4] generally used in the industry. The approach is limited by our ability to accurately measure needed on-farm growth model parameters (such as maximum protein deposition levels) and the accuracy of the growth models themselves. Nevertheless, trials have been conducted in New Zealand, based on the outcomes of the optimisation, in an attempt to improve production efficiency.

The paper is organised as follows. In the next section we describe the objective function to be maximised and its domain. In Section 3 we discuss algorithms used to find the objective function optimum. The nature of the objective function is explored in Section 4 and an idea for an improved algorithm proposed. The paper concludes with a summary.

2. The domain and the objective function

In this section the domain of the objective function and the objective function itself are described.

2.1 Domain

An animal "diet", specifying nutrition for a certain growth period, and required as input to a simple growth model, can be described using only three parameters, p, r and d, defined as follows:

$$\begin{aligned} p &= \text{proportion of the } \textit{ad libitum} \text{ digestible energy intake} \\ r &= \text{minimum lysine to digestible energy ratio, in grams per megajoule} \\ d &= \text{digestible energy density, in megajoules per kilogram} \end{aligned}$$

The *ad libitum* digestible energy intake is determined by a standard National Research Council (NRC) [3] curve, relating digestible energy to live weight of the animal. Parameter p determines the proportion (typically around 0.8) of that amount to be fed. Lysine is an amino acid, required for growth, and generally the amino acid found to be limiting in a diet. For that reason we specify the level of lysine required using parameter r and specify the other amino acids needed for growth (in a particular ratio to lysine - providing so-called "balanced" protein) using the lysine level. Finally, the energy density d of the diet must be within limits in order to ensure palatability.

We term a "feeding schedule" to be a finite sequence of diets, indexed by i, with the ith diet fed for a predetermined time of T_i days (fixed at the outset). Thus we write a feeding schedule as

$$F = (p_1, r_1, d_1; p_2, r_2, d_2; \ldots; p_n, r_n, d_n)$$

Typically $T_i = 7$ for each i, so diets are fed for a week, and $n = 15$ diets are offered. This total period of 105 days, for New Zealand conditions, amply covers the usual time from weaner arrival (it is assumed that the producer buys in weaners) to slaughter date. Typical parameter ranges used are $[0.8, 1]$ for p, $[0.2, 1.2]$ for r and $[12, 17]$ for d. Our aim will be to find the optimal feeding schedule and slaughter date, so the domain of the problem is

$$P_1 \times R_1 \times D_1 \times P_2 \times R_2 \times D_2 \times \ldots \times P_n \times R_n \times D_n \times \{1, 2, \ldots, T\}$$

where $P_i = [0.8, 1]$, $R_i = [0.2, 1.2]$ and $D_i = [12, 17]$ for each $i = 1, \ldots, n$ and $T = \sum_i T_i$.

2.2 Objective function

The objective function to be maximised is profit, or gross margin at market, given by

$$g(F, x) = \text{Gross Return}(F, x) - \text{Feed cost}(F, x) - \text{Weaner Cost}$$

where F is a feeding schedule and x the slaughter date. Throughout, we measure this as "profit per pig" (an alternative is to measure "profit per pig place per year"). The weaner cost is fixed, typically in New Zealand at NZ$70. Feed cost *FC* is a function of F and x, as is the gross return *GR*. In Table 1 we give a typical linear programme using New Zealand data, showing ingredients, their cost, their composition and the associated constraints, when using a diet with r and d. In Table 2 we show a typical New Zealand price schedule; the return on a pig depends on backfat thickness and carcass weight.

Objective function: Single pig The flow chart in Figure 1 details how, given a feeding schedule F and slaughter date x, together with the p_i and genotype parameters of

$$\begin{aligned} minLP &= \text{minimum allowable lipid to protein ratio} \\ Pdmax &= \text{maximum daily protein deposition} \end{aligned}$$

The gross return $GR(F, x)$ and total feed cost $FC(F, x)$ can be computed. In turn, the objective function $g(F, x)$ is then calculated. An iteration of the routine uses r_i and d_i to complete the

Table 1. A linear programme providing the least cost diet.

	Barley		Soybean		. . .		Meat and bone		
Minimise:	0.25B	+	0.715S	+	· · ·	+	0.5M		
Digestible energy	13.2B	+	15.86S	+	· · ·	+	12M	=	d
Lysine lower bound	3.19B	+	27.33S	+	· · ·	+	16.19M	≥	rd
Balanced	2.85B	+	10.74S	+	· · ·	+	20.3M	≥	0.62rd
amino	1.2B	+	4.68S	+	· · ·	+	1.09M	≥	0.19rd
acid	2.42B	+	11.48S	+	· · ·	+	9.95M	≥	0.32rd
lower	8.34B	+	36.75S	+	· · ·	+	18.91M	≥	0.95rd
bounds	3B	+	16.22S	+	· · ·	+	9.54M	≥	0.67rd
	6.85B	+	33.25S	+	· · ·	+	21.5M	≥	rd
					⋮				
Mineral	0.5B	+	3S	+	· · ·	+	105M	≥	8
bounds	0.5B	+	3S	+	· · ·	+	105M	≤	13
	3.1B	+	2.6S	+	· · ·	+	52M	≥	7
	3.1B	+	2.6S	+	· · ·	+	52M	≤	11
	0.2B	+	0.1S	+	· · ·	+	5.5M	≥	10
	0.2B	+	0.1S	+	· · ·	+	5.5M	≤	20
					⋮				
Ingredient	$-0.4B$	+	0.6S	−	· · ·	−	0.4M	≤	0
upper	$-0.1B$	−	0.1S	−	· · ·	−	0.1M	≤	0
bounds	$-0.3B$	−	0.3S	−	· · ·	−	0.3M	≤	0
	$-0.4B$	−	0.4S	−	· · ·	−	0.4M	≤	0
					⋮				
Diet mass	B	+	S	+	· · ·	+	M	=	1

Table 2. A New Zealand price schedule giving prices in cents per kg for pigs at slaughter in July 2001.

	Carcass weight (kg)										
Backfat (mm)	35.0 and under	35.1 to 40.0	40.1 to 45.0	45.1 to 50.0	50.1 to 55.0	55.1 to 60.0	60.1 to 65.0	65.1 to 70.0	70.1 to 75.0	75.1 to 80.0	80.1 and over
<6	300	300	300	300	300	300	300	300	300	300	300
6-9	360	385	395	395	385	370	370	370	370	365	335
10-12	360	385	385	390	375	370	370	370	370	365	335
13-15	330	330	330	330	330	335	335	335	335	330	305
16-18	260	260	260	260	260	270	270	270	270	270	270
>18	230	230	230	230	230	240	240	240	240	240	240

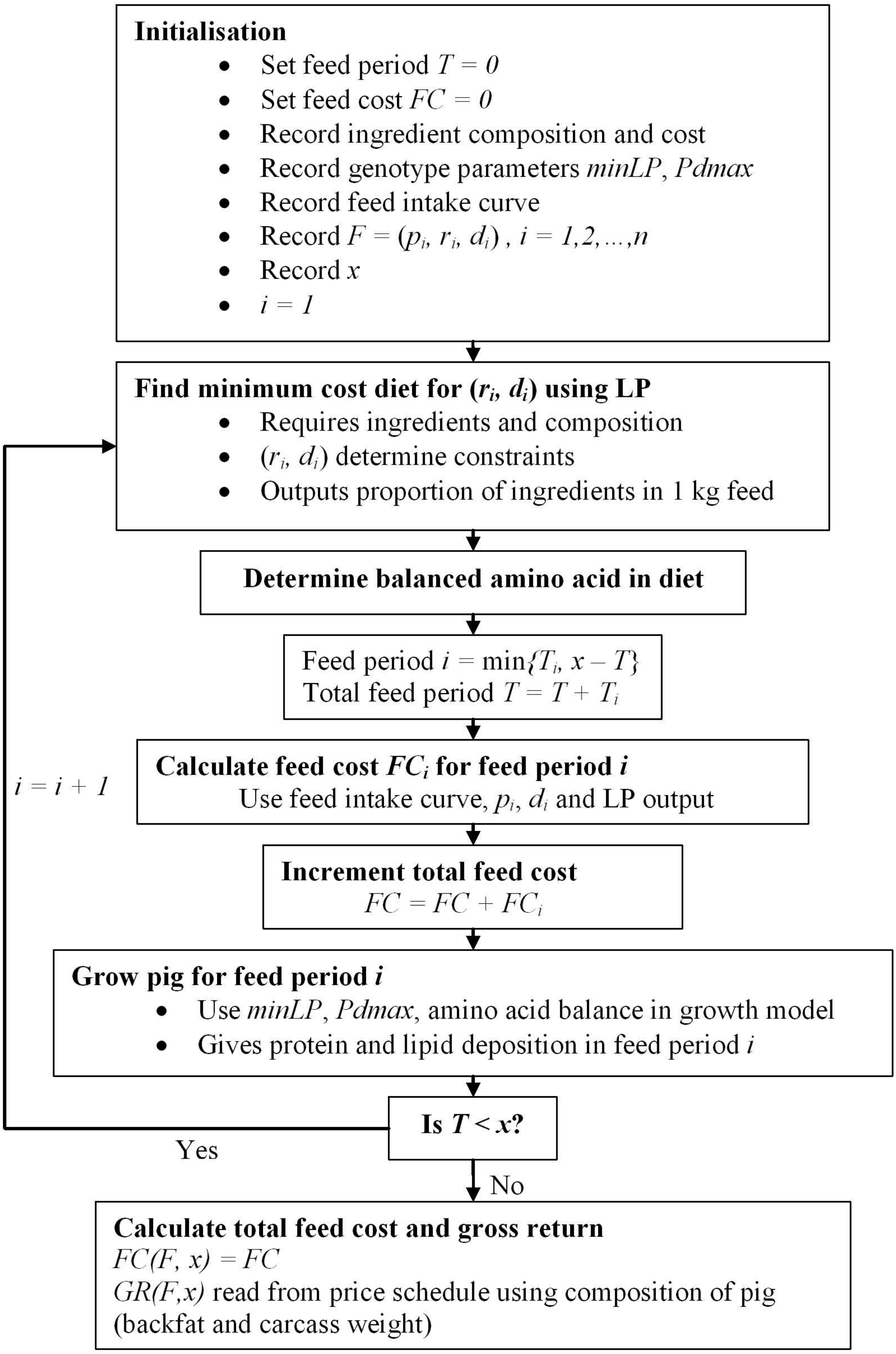

Initialisation
Set feed period $T = 0$
Set feed cost $FC = 0$
Record ingredient composition and cost
Record genotype parameters *minLP*, *Pdmax*
Record feed intake curve
Record $F = (p_i, r_i, d_i)$, $i = 1,2,\ldots,n$
Record x
$i = 1$
Find minimum cost diet for (r_i, d_i) using LP
Requires ingredients and composition
(r_i, d_i) determine constraints
Outputs proportion of ingredients in 1 kg feed
Determine balanced amino acid in diet
Feed period $i = \min\{T_i, x - T\}$
Total feed period $T = T + T_i$
$i = i + 1$
Calculate feed cost FC_i for feed period i
Use feed intake curve, p_i, d_i and LP output
Increment total feed cost
$FC = FC + FC_i$
Grow pig for feed period i
Use *minLP*, *Pdmax*, amino acid balance in growth model
Gives protein and lipid deposition in feed period i
Is $T < x$?
Yes
No
Calculate total feed cost and gross return
$FC(F, x) = FC$
$GR(F,x)$ read from price schedule using composition of pig
(backfat and carcass weight)

Figure 1. Calculation of $g(F, x)$

right-hand-side constraints in the linear programme (see Table 1); the least cost makeup of 1kg of feed for this period is the output. Together with p_i and the standard NRC feed intake curve this allows the feed cost for this ith period to be computed. The amount of balanced amino acid can also be calculated. This, together with the genotype parameters (and at the start the initial mass P_0 of protein in the pig) and the growth model, allows us to grow the pig for the ith period; protein and lipid deposition are recorded. Continuing until the slaughter date in this way allows us to compute $FC(F, x)$ (by summing the individual period feed costs) and $GR(F, x)$ (by referring the configuration of the pig at slaughter date to the price schedule of Table 2).

Finally, we determine the maximum gross margin per pig, using feeding schedule F, as $g(F) = \max_x g(F, x)$.

Objective function: Multiple pigs The optimal feeding schedule for a single pig is an unrealistic schedule, since in practice many pigs, exhibiting minor variations in genotype and feed intake, are grown on a single feeding schedule. The optimum schedule in such a situation is different from that found for a single pig. For this reason we have introduced variation to $minLP$, $Pdmax$ and the feed intake curve values across the pigs in a batch (using a coefficient of variation for these quantities of generally 5%, and 200 pigs in a batch). In this case, $g(F, x) = \sum_{i=1}^{N} g_i(F, x)$ where g_i is the gross margin for the ith pig, calculated as in the previous subsection. As before, $g(F) = \max_x g(F, x)$

3. Optimisation

For moderately complex problems, pure random search is not able to find the optimum. To date, a number of standard optimisation techniques have been applied to this problem - tabu search, ascent, simulated annealing, Nelder-Mead and a genetic algorithm. The last mentioned has been found to be most successful to date. The “genome" is the feeding schedule; crossover and mutation act in a natural way. Associated software, called “Bacon Max" has been developed in Visual C++. Extensive testing to tune the genetic algorithm has been conducted and generally a population size of 20 feeding schedules has been found to be satisfactory. Twenty iterations with no change in the value of the objective function has provided a suitable stopping rule. The genetic algorithm is slow, however, taking up to an hour to satisfactorily solve a typical problem.

4. The nature of the objective function

Cross-sections through the objective function have been drawn, revealing the structure of a craggy volcano. A representative cross-section is shown in Figure 2.

We are currently exploring applying the idea behind the “conjugate gradients" method to this problem - successively performing one-dimensional optimisations, using orthogonal directions, to reduce the time required to find the optimum.

5. Summary

A combination of nonlinear optimisation tools, an animal growth model and linear programming can be used to find feeding schedules which maximise pig producer profitability. Pure random search fails to find the optimum; a genetic algorithm has been found so far to be the most successful approach. Knowledge of the objective function shape is expected, in the near future, to lead to an improved optimisation routine.

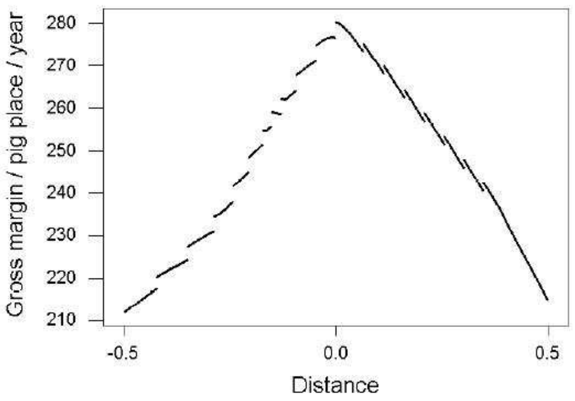


Figure 2. The craggy volcano shape of a section through the objective function at the optimal solution. (Dr. D.L.J. Alexander is thanked for this graphic.)

References

[1] Alexander, D.L.J., Morel, P.C.H. , and Wood, G.R. (2005). *Feeding strategies for maximising gross margin in pig production*, Optimization in Action, Kluwer (to appear).

[2] de Lange, C.F.M. (1995). *Framework for a simplified model to demonstrate principles of nutrient partitioning for growth in the pig, Modelling Growth in the Pig*, EAAP Publication 78 (Moughan, P.J., Verstegen, M.W.A. and Visser-Reyneveld, M.I. (eds)) 71-85.

[3] National Research Council (1998). *Nutrient requirements of swine*. 10th edition. Reeves R.C., and E.J. Rowe. 2002. Genetic algorithms-principles and perspectives, A Guide to GA theory. Kluwer Academic Publishers.

[4] Varley, M. (2001). *The genetics of lean tissue growth in the pig*, SCA Nutrition Ltd, North Yorkshire, UK. 4th Animal Feed Manufacturers Association Congress

Proceedings of GO 2005, pp. 263 – 267.

Optimal Triangulation: Old and New Problems*

Yinfeng Xu[1] and Wenqiang Dai[2]

[1] *School of Management, Xi'an Jiaotong University, Xi'an, 710049, P. R. China,*
The State Key Lab for Manufacturing Systems Engineering, P. R. China, yfxu@mail.xjtu.edu.cn

[2] *School of Management, Xi'an Jiaotong University, Xi'an, 710049, P. R. China,* wqdai@mail.xjtu.edu.cn

Abstract The optimal triangulation is an important topic in computational geometry and arises in many different fields. This paper surveys several unsolved problems in optimal triangulations and especially focus on some new problems which need further research.

Keywords: minimum weight triangulation, Delaunay triangulation, $k-$optimality, $\beta-$skeleton, computational complexity

1. Introduction

A *triangulation* of a given points set S in the plane is a maximal set of non-crossing line segments (called edges) which have both endpoints in S. In two dimensions, a triangulation partitions the interior of the convex hull of the given point set into triangles and these triangles interest only at shared edges and vertices. There are many areas of engineering and scientific applications for triangulation, such as finite element, numerical computation, computer aided design(CAD), computational geometry, etc [4,7].

From the view of application, it is important to confine geometric constraints on the shape of triangle in obtained triangulation. Several measures of triangle quality have been proposed, which are based on edges length, angles, areas and other elements of the individual triangles in a triangulation. An optimal triangulation is the one that is best according to some criterion of these measures.

More recently, there have already some surveys on the optimal triangulation, such as [5,5, 8,20,22]. This paper also deals with this topic, but our focus is the set of planar points with the Euclidean metric in E^2, which is the most important case for applications. The emphasis of our work is on problems of optimal triangulation. The goal is to survey some unsolved problems and propose some new problems which need further research.

2. Optimal Triangulation

2.1 Delaunay Triangulation

The Delaunay triangulation of a point set S, $DT(S)$, is the planar dual of the Voronoi diagram, $V(S)$. The Voronoi diagram is a partition of the plane into polygonal cells, one for each input point, so that the cell for input point s consists of the region of the plane closer to s than to any other input point. See [4,13,18] for extensive discussion and surveys. Various global

*This research is supported by NSF of China under Grants No. 10371094 and 70471035.

optimality properties of $DT(S)$ can be proved by observing that every good flip gives a *local improvement* of the respective optimality measure [5,5]. On the contrary, $DT(S)$ fails to fulfill some optimization criteria, such as minimizing the longest edge or minimizing the maximum angle. There are still no any results on some optimal criteria such as following.

Old Problem 1: *How to find a minmax area triangulation or a minmax perimeter traingulation?*

2.2 Minimum Weight Triangulation(MWT)

The most longstanding problem among all the optimal criteria is the *minimum weight triangulation* (MWT), in which we assign a weight for the edge as its Euclidean length, define the weight of a triangulation as the sum of the edge weight in the triangulation and our goal is to find or generate a triangulation with the minimum weight.

The complexity of computing the MWT for arbitrary planar point set is still open: this problem is not known to be NP-hard, nor is it known to be solvable in polynomial time.

Old Problem 2: *Whether MWT is a $NP - hard$ problem?*

Dynamic Programming is a powerful tool to deal with some discrete optimization problems. Gilbert [14] and Klincsek [15] gave a dynamic programming algorithm for computing a minimum weight triangulation of a convex polygon, and their algorithm do also well for simple polygon. The time complexity and the space complexity are $O(n^3)$ and $O(n^2)$ respectively. If we may find a fast algorithm to compute a MWT of a convex polygon, there will be a light to improve some heuristics for computing the MWT.

Old Problem 3: *Whether the MWT of a convex polygon can be found in $o(n^3)$ time?*

However, for some special points arrangement, such as [3,10,16,20], the minimum weight triangulation of the point set can be computed in polynomial time with Gilbert and Klincsek's algorithm. The detailed discussion is omitted in this extend abstract.

Some studies, see [20,24], have pointed out that to find more edges in a MWT can improve the performance for some heuristics. Thus to find some new subgraphs in a MWT is very helpful. Some problems about the edges which always belong to MWT and their properties are discussed in the full version of this paper.

For the *minimum weight Steiner triangulation (MWST)* problem, our goal is to find or generate a triangulation of a superset of S with minimum total Euclidean length. There have large number of result in this field, such as [12]. Among these studies, following problem is an important and unsolved one.

Old Problem 4: *Whether the Steiner points in the minimum weight Steiner triangulation of a convex polygon all lie on the boundary?*

2.3 Local Optimal Triangulation

Call a triangulation local optimal triangulation if no any diagonal edge of any convex quadrilateral can be flipped to reduce its weight. A triangulation $T(S)$ is called a *k-optimal triangulation* for $4 \le k < n$, denoted by $LOT_k(S)$, if every k-sided simply polygon drawn from $T(S)$ is optimally triangulated by some edges of $T(S)$ with minimal weight. It is easily to observed that $GT(S)$ and the triangulation obtained by longer diagonal edge's flipped to shorter diagonal edge in any convex quadrilateral are all local optimal, thus are all 4-optimal triangulations. All of these triangulations can be computed in polynomial time, but up to now there is no any result on how to get a 5-optimal triangulation in polynomial time.

New Problem 1: *Whether there is a polynomial time algorithm to find a 5-optimal triangulation?*

We are able to give an algorithm, which is motivated by the *Edge Insertion algorithm* [6], to compute a 5-optimal triangulation for a point set S:

Given a triangulation $\mathcal{A}$ of a point set S, the function EDGE-lNSERTION of ab, $a, b \in S$ is

```
Function EDGE-lNSERTION($\mathcal{A}$, $ab$): triangulation.
  1. $\mathcal{B} := \mathcal{A}$;
  2. Remove from $\mathcal{B}$ all edges that intersect $ab$ and produce a polygon $P_{ab}$;
  3. Use the Dynamic Programming to retriangulate the polygonal regions $P_{ab}$ in step 2;
  4. return $\mathcal{B}$.
```

Thus, our algorithm can now be formulated as follows.

```
Input. A set $S$ of $n$ points in $\Re^2$.
Output. A triangulation $\mathcal{T}$ of $S$.
Algorithm. Construct an arbitrary triangulation $\mathcal{A}$ of $S$.
          repeat $\mathcal{T} := \mathcal{A}$;
            for all pairs $a, b \in S$ do
                $\mathcal{B}$: =EDGE-INSERTION($\mathcal{A}$, $ab$);
                if the weight of $\mathcal{B}$ is less than the weight of $\mathcal{A}$ then $\mathcal{A} := \mathcal{B}$;
                exit the for-loop
                endif
            endfor
          until for all pairs $a, b \in S$ and $\mathcal{B}$=EDGE-INSERTION($\mathcal{A}$, $ab$), the weight of $\mathcal{B}$ is no
                less than the weight of $\mathcal{A}$.
```

We can obtain the following theorem.

Theorem 1. *The above algorithm gives a 5-optimal triangulation for given point set S.*

However, we do not know whether the above algorithm terminates in polynomial time.

New Problem 2: *Whether the above algorithm is a polynomial time algorithm to find a 5-optimal triangulation?*

Let $E(S)$ denote the set of all the segments with endpoints in S. A line segment pq with $p, q \in S$ is called a *stable line* segment of all triangulations of S, if no line segment in $E(S)$ properly intersects pq. The stable line segments thus have to appear in any triangulation of S and are in $MWT(S)$. Denote $E_3(S)$ is the set of all stable line segments in S. Xu [21] has discussed some properties of this segment set and shown that the number of $E_3(S)$ does not exceed $2n - 2$. An important property is the relationship between $E_3(S)$ and the k-optimal triangulations. Let $E_k(S)$ denote the intersection of all possible $LOT_k(S)$, then we have

$$E_3(S) \subseteq E_4(S) \subseteq \cdots \subseteq E_k(S) \subseteq \cdots \subseteq E_{n-1}(S) \subseteq MWT(S)$$

For some special cases of point set S, $E_3(S)$ forms a connected graph, thus an $MWT(S)$ can be constructed in polynomial time using the dynamic programming algorithm. Mirzaian et al. [17] have shown that $E_3(S)$ can be found in $O(n^2 \log n)$ time and $O(n)$ space.

It is pointed out that for uniformly distributed points, the expected number of connected components is Θ(n), see Bose et al. [9]. A surprisingly larger subgraph of $E_4(S)$, the so-called $LMT-skeleton$ can be constructed in $O(n^3 \log n)$ time and $O(n^2)$ space [11]. But even $LMT-skeleton$ has many edges and it is still a proper subset of $E_4(S)$ for some point set [1]. From the practical point of view, the $LMT - skeleton$ is always nearly a triangulations of S [5].

The fact that $E_4(S)$ is the subgraph of $MWT(S)$, but no algorithm is found to compute $E_4(S)$ in polynomial time, leads to the following problem.

Old Problem 5: *Is it possible to find the intersection of all local optimal triangulations in a polynomial time?*

For a given edge e, we may ask a question as whether there is a local optimal triangulation $T_L(S)$ such that e is an edge of $T_L(S)$. To observe an edge e in $E_4(S)$, we may know that e is in any local optimal triangulation of S and for any edge e' with endpoints in S interests e, e' can not be in any local optimal triangulation. Let $E_4^{'}(S)$ denote the set of edges do not in any 4-optimal triangulation for point set S, and $E_4^{*}(S) = \{e \mid$ there is a local optimal triangulation $T(S)$ such that $e \in T(S)\}$. From the definition of $E_4(S)$, $E_4^{'}(S)$ and $E_4^{*}(S)$, we have

$$E(S) = E_4^{'}(S) \cup E_4^{*}(S), E_4(S) \subset E_4^{'}(S), E_4(S) \cap E_4^{'}(S) = \phi, E_4^{'}(S) \cap E_4^{*}(S) = \phi$$

If any one of $E_4(S)$, $E_4^{'}(S)$, $E_4^{*}(S)$ can be computed in polynomial time then we may compute the another two in polynomial time easily. So the following problem is interesting.

New Problem 3: *How to test whether there is a local optimal triangulation contains a given edge e?*

2.4 Pseudo-triangulation

A *pseudo-triangle* is a simple polygon with exactly three vertices where the inner angel is less than π. A *pseudo-triangulation* of a point set S is a partition of the interior of the convex hull of S into a set of pseudo-triangles.

In many ways, pseudo-triangulations have nicer properties than classical triangulations of a point set [2,19]. Among them, a *minimum pseudo-triangulation* of a points set is one with the smallest possible number of edges and a *minimum weight pseudo-triangulation ($MWPT$)* is a pseudo-triangulation which minimizes the sum of the edge lengths. No result is known about the complexity of computing a $MWPT$ of a given point set S, and a similar problem arises for finding a pseudo-triangulation within a given triangulation.

New Problem 4: *Whether to find a $MWPT$ is NPC?*

New Problem 5: *How to find a $MWPT$ as a subgraph of a given triangulation?*

2.5 Mesh Generation

Generating triangular meshes is one of the fundamental problems in computational geometry, and has been extensively studied; see e.g. the survey article by Bern Eppstein [5]. In view of the field of application, it is quite natural to consider mesh generation problem under some optimal criteria. Recently, we consider the problem of a mesh generation with some edge length constraints. Among the most fascinating and challenging, we mention the following.

New Problem 6: *For given real numbers $\alpha \leq \beta \leq \gamma$, and a convex polygon P, how to find a triangulation, $T(P)$, of P such that the inner edge length in $T(P)$ is in the interval $[\alpha, \gamma]$ and the number of edges with edge length different from β is minimum?*

In [23], we have presented a heuristics to generate a triangular mesh for a special case of the above problem, but for a general case, this problem is still open.

3. Conclusion

We give a list of unsolved problems related with optimal triangulations both from the theoretical and the application aspects, and show some partial results on the problems in the paper. To keep our attention on the practical aspects of computing for computational geometry, some new challenge problems on optimal triangulations need to be solved.

References

[1] O. Aichholzer, F. Aurenhammer and R. Hainz, New results on MWT subgraphs, *Information Processing Letters*, 69, 215-219,1999.

[2] O. Aichholzer, D. Orden, F. Santos and B. Speckmannt, On the number of pseudo-triangulations of certain point sets, In *Proc. CCCG'03*, 141-144, Halifax, Nova Scotia, Canada, 2003.

[3] E. Anagnostou and D. Corneil, Polynomial-time instances of the minimum weight triangulation problem, *Computational Geometry: Theory and Application*, 3,247-259, 1993.

[4] F. Aurenhammer, Voronoi diagrams-a survey of a fundamental geometric data structure, *ACM Computing Surveys*, **23**, 345-405, 1991.

[5] F. Aurenhammer and Y. F. Xu, Optimal triangulations, In PM. Pardalos and CA. Floudas, editor, *Encyclopedia of Optimization*, volume **4**, 160-166, Kluwer Academic Publishing, 2000.

[6] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell and T. S. Tan, Edge insertion for optimal tiangulations, *Dist. Comput. Geom*, **10**, 47-65, 1993.

[7] M. Bern and D. Eppstein. Mesh generation and optimal triangulation, In : D. -Z. Du and F Hwang, editors, *Computing in Euclidean Geometry, Lecture Notes Series in Computing 4*, World Scientific, Singapore, 47-123, 1995.

[8] M. Bern and D. Eppstein, Approximation algorithms for geometric problems, In D. S. Hochbaum, editors, *Approximation Algorithms for NP-hard Problems*, 290-345, PWS , 1995.

[9] P. Bose, L. Devroye and W. Evans, Diamonds are not a minimum weight triangulation's best friend, In *Proc. CCCG'96*, 68-73,1996.

[10] S. W. Cheng, M. J. Golin and J. C. F. Tsang, Expected-case analysis of β-skeletons with applications to the construction of minimum-weight triangulations, In *Proc. CCCG'95*, 279-283, 1995.

[11] S. W. Cheng, N. Katoh and M. Sugai, A study of the LMT-skeleton, in *Proc. ISAAC'96*, LNCS 1178, Springer-Verlag, 256-265, 1996.

[12] D. Eppstein. Approximating the minimum weight triangulation, *Disc. and Comp. Geometry*, **11**, 163-191, 1994.

[13] S. Fortune, Voronoi Diagrams and Delaunay triangulations, in: D. -Z. Du and F. Hwang, ed., *Computing in Euclidean Geometry, Lecture Notes Series in Computing 4*, World Scientific, Singapore, 225-265, 1995.

[14] P. D. Gilbert, *New results in planar triangulation*, report R-850, Coordinated Science Laboratory, University of Illinois, 1979.

[15] G. T. Klincsek, Minimal triangulations of polygonal domains, *Ann. Discrete Math*, 9, 127-128, 1980.

[16] H. Meijer and D. Rappaport, Computing the minimum weight triangulation for a set of linearly ordered points, *Information Processing Letters*, 42, 35-38,1992.

[17] A. Mirzain, C. A. Wang and Y. F. Xu, On stable line segmets in triangulations, in: *Proc. CCCG'96*, 68-73, 1996.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introuduction*, Springer-Verlag, 1985.

[19] G. Rote, Pseudotriangulations, polytopes, and how to expand linkages, In *Proc of the 18th annual symposium on Computational Geometry*, 133-134, 2002.

[20] Y. F. Xu, *Minimum weight triangulation problem of a planar point set*, Ph.D. Thesis. Institute of Applied Mathematics, Academia Sinica, Beijing, 1992.

[21] Y. F. Xu, On stable line segments in all triangulations, *Appl. Math. -JCU*, 11B,2, 235-238, 1996.

[22] Y. F. Xu, Minimum weight triangulations, In D. -Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimaization (Vol. 2)*, Kluwer Academic Publishers, 617-634, 1998.

[23] Y.F.Xu, W.Q.Dai, N. Katoh and M. Ohsaki, Triangulating a convex polygon with small number of non-standard bars, to appear in *Proc. COCOON05*, 2005.

[24] Y. F. Xu and D. Zhou, Improved heuristics for the minimum weight triangulation, *Acta Mathematicae Applicatae Sinica* 4(11), 359-368, 1995.

Author Index