TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Juliane Müller

Surrogate Model Algorithms for Computationally Expensive Black-Box Global Optimization Problems

Julkaisu 1092 • Publication 1092

Tampere 2012

Tampereen teknillinen yliopisto. Julkaisu 1092
Tampere University of Technology. Publication 1092

Juliane Müller

Surrogate Model Algorithms for Computationally Expensive Black-Box Global Optimization Problems

Thesis for the degree of Doctor of Philosophy to be presented with due permission for public examination and criticism in Sähkötalo Building, Auditorium S4, at Tampere University of Technology, on the 28th of November 2012, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2012

ISBN 978-952-15-2959-7 (printed)
ISBN 978-952-15-2991-7 (PDF)
ISSN 1459-2045

Abstract

Surrogate models (also called response surface models or metamodels) have been widely used in the literature to solve continuous black-box global optimization problems that have computationally expensive objective functions. Different surrogate models such as radial basis functions, kriging, multivariate adaptive regression splines, and polynomial regression models have been used in various applications. It is in general however unknown which model will perform best for a given application, and computation time restrictions do not allow trying different models. Thus, in the first part of this thesis, a family of algorithms (SO-M, SO-M-c, SO-M-s) based on using a mixture of surrogate models is developed. The second part of the thesis extends the research in using surrogate models for mixed-integer (algorithm SO-MI) and purely integer (algorithms SO-I) optimization problems. Finally, a real world application problem arising in the agricultural land use management of a watershed is examined (algorithms SO-Ic).

The algorithm SO-M uses Dempster-Shafer theory to combine information derived from various model characteristics in order to determine the influence of individual models in the mixture. Extensions of SO-M with respect to the sampling strategy (algorithms SO-M-c and SO-M-s) have been compared in numerical experiments, and it was found that whenever it is a priori unknown which surrogate model should be used, it is advisable to use a mixture model in order to prevent accidentally selecting the worst model. It could be shown that mixture models containing radial basis function interpolants generally work very well, whereas using only polynomial regression models should be avoided. Moreover, algorithms using mixture models often outperform the algorithms that use only the single models that are contributing to the mixture.

Although there are many computationally expensive black-box optimization applications that have besides continuous also integer variables, or that have only integer variables, algorithms for solving these types of problems

are scarce. In the second part of this thesis two algorithms, namely SO-MI for mixed-integer problems, and SO-I for purely integer problems have been developed and were shown to find accurate solutions for computationally expensive problems with black-box objective functions and possibly black-box constraints. The constraints were treated with a penalty approach and numerical experiments showed that the surrogate model based algorithms outperformed commonly used algorithms for (mixed-) integer problems such as branch and bound, and genetic algorithms. Also NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) has been included in the comparison. NOMAD is suitable for integer and mixed-integer black-box problems, but its performance for these problem types has not been studied in the literature. In the numerical experiments, NOMAD also proved superior as compared to branch and bound and the genetic algorithm, but it performed worse than SO-I and SO-MI for most test problems.

Lastly, the algorithm SO-I has been further extended to directly handling constraints with a response surface. The algorithm, SO-Ic, has been developed specifically for a watershed management problem that has only one constraint, but SO-Ic is easily generalizable for problems with more constraints. In the considered application problem parts of the agricultural land in the Cannonsville reservoir watershed in upstate New York have to be retired in order to decrease the total phosphorus runoff to a given limit at minimal cost. A computationally expensive simulation model has to be used to compute the costs and phosphorus runoff. The performance of SO-Ic has been compared to a genetic algorithm, NOMAD, and the discrete dynamically dimensioned search algorithm on three problem instances with different sizes of the feasible region. The surrogate model based algorithm SO-Ic performed also for these problems significantly better than all other algorithms and could be shown to be the most robust.

Acknowledgments

At this point I want to thank my dear friend Jarno "The King of Sweden" Marttila for the quality company at the mathematics department coffee table, for making me laugh, and for distracting me from work whenever there was a deadline lurking. I also want to thank Nina Flink for her friendship and for looking after Professor Gauss whenever I was traveling to conferences.

I would like to show my gratitude to my Tekiila-friends Jaakko "Padawaani" Tyynismaa, Joonas Multanen, Juha Manninen, Tero Marttila, and Kalle Laakso for the first class climbing time we spent together at the Finnish crags and who made me forget about work during weekends. I am thankful for having found so many friends in Ithaca who made my research visit at Cornell University unforgettable: Sheila Saia, Ben Currens, Andy Casler, Scott Cambo, Chris Chilas, Brian Harrington, Stephen Demjanenko, and Sebastian Bauer. I want to thank Carlos Nieto for his constant encouragement. Thanks to Flor and Mark Cianchetti for their friendship and moral support at Cornell University. Thanks also to Markus "The Peasant" Rajala and Jussi Kangas who always made the days look a bit brighter.

I want to express my gratitude to Prof. Shoemaker and Prof. Piché for supervising my thesis and giving me the chance to visit Cornell University. I would like to thank my thesis examiners, Rommel Regis and Stefan Wild, for their helpful comments and suggestions, and Jorge Moré for being my opponent.

Although she probably does not know how much of a help she has been, I would like to thank Professor Gauss for the companionship.

Lastly, I would like to thank the Tampere Graduate School in Information Science and Engineering for the financial support, and the Finnish Academy of Science and Letters, Vilho, Yrjö and Kalle Väisälä Foundation for the scholarship for my stay at Cornell University.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Abbreviations and Nomenclature

Discrete-DDS	Discrete dynamically dimensioned search
MARS	Multivariate adaptive regression splines
NOMAD	Nonsmooth optimization by mesh adaptive direct search
RBF	Radial basis function
SO-I	Surrogate Optimization - Integer
SO-Ic	Surrogate Optimization - Integer constraints
SO-M	Surrogate Optimization - Mixture
SO-M-c	Surrogate Optimization - Mixture - candidate sampling
SO-M-s	Surrogate Optimization - Mixture - surface minimum
SO-MI	Surrogate Optimization - Mixed Integer

$f(\cdot)$	Objective function, see equation (1.1a)
$\mathbf{x}$	Decision variable vector, see equation (1.1d)
$\mathbf{x}^T$	Transpose of $\mathbf{x}$
x_i	ith decision variable, $i = 1, \ldots, k$, see equation (1.1c)
x_i^l, x_i^u	Lower and upper bound for ith decision variable, see equation (1.1c)
k	Problem dimension, see equation (1.1c)
Ω	Variable domain, see equation (1.1d)
$c_j(\cdot)$	jth constraint function, $j = 1, \ldots, m$, see equation (1.1b)
m	Number of constraints
$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integer numbers
$s(\cdot)$	Surrogate model
n	Number of already evaluated points
$\mathbf{x}_\iota$	ιth sample point, $\iota = 1, \ldots, n$
$\mathbf{y}$	Vector of objective function values $(y_1, \ldots, y_n)^T$
$s_p(\cdot)$	Polynomial regression model, see equations (1.3) and (1.6)
$s_m(\cdot)$	MARS model, see equation (1.7)
$s_b(\cdot)$	RBF model, see equation (1.9)
$s_k(\cdot)$	Kriging response surface model, see equation (1.12)
$s_{\text{mix}}(\cdot)$	Mixture surrogate model, see equation (1.16)
$\mathcal{M}$	Set of all models contributing to the mixture, see equation (1.16)
$s_r(\cdot)$	rth surrogate model in the mixture, see equation (1.16)
w_r	Weight of the rth model in the mixture, see equation (1.16)

1.1 Motivation and Problem Statement

Application problems in engineering design, management, and finance, for example, often require computationally expensive computer analysis codes to capture the physical behavior of the systems under consideration. Continuously increasing computational power enables the development of simulation models that become more and more complex. With increasing model complexity, the computation time of these simulations increases significantly. When addressing optimization problems where the objective function evaluation is based on such computationally expensive simulations, the obvious goal is to find a good approximation of the global optimum within as few function evaluations as possible in order to obtain a solution within reasonable time.

In this thesis the optimization problems are formulated as minimization problems (a maximization problem can be reformulated by minimizing the negative of the objective function). The global optimization problems considered here are in general of the following form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) && \text{(1.1a)} \\ \text{subject to} \quad & c_j(\mathbf{x}) \leq 0,\ j = 1, 2, \ldots, m && \text{(1.1b)} \\ & -\infty < x_i^l \leq x_i \leq x_i^u < \infty,\ i = 1, 2, \ldots, k && \text{(1.1c)} \\ & \mathbf{x} \in \Omega, && \text{(1.1d)} \end{aligned}$$

where $f(\mathbf{x}) : \mathbb{R}^k \to \mathbb{R}$ is the deterministic objective function whose value is computed by a computationally expensive simulation model, $c_j(\mathbf{x})$ are deterministic constraint functions, and x_i^l and x_i^u are the lower and upper bounds of the ith variable. If there are no constraints $c_j(\mathbf{x})$, then the problem is referred to as a box-constrained (also unconstrained) global optimization problem. If the constraints $c_j(\mathbf{x}), j = 1, \ldots, m$, are present, it is assumed that their function values are computed in the same computationally expensive simulation as the objective function, i.e. the simulation model returns the objective *and* constraint function values. Moreover, it is assumed that the feasible set is nonempty.

The number range that can be assumed by the variables $x_i, i = 1, \ldots, k$, is either $\mathbb{R}$ (purely continuous problems, addressed in Chapters 2 and 3), $\mathbb{Z}$

(purely integer problems, addressed in Chapters 5 and 6), or $\mathbb{R}$ for some variables and $\mathbb{Z}$ for other variables (mixed-integer problems, addressed in Chapter 4).

The objective function has several characteristics that make it difficult to find its global optimum. As stated before, evaluating the objective function requires running the computationally expensive simulation model, which may take from several minutes to several hours or even days. Furthermore, the simulation model is in general a black-box, and all that is known is the deterministic output for a given input variable vector. The algebraic form of the objective function $f(\mathbf{x})$ is unknown, and therefore derivative information is not available. Finite differencing or automatic differentiation are in general not an option. Finite differencing may require too many function evaluations, which in turn increases the computation time. For a k-dimensional problem, computing the gradient and the Hessian of $f(\mathbf{x})$ is, respectively, k-times and k^2-times as expensive as computing $f(\mathbf{x})$ [56]. Furthermore, finite differencing gives only derivative estimates and no exact values. Automatic differentiation is often not applicable because for many problems the codes of the simulation models are not fully available due to confidentiality restrictions. Also, if the evaluation of the objective function involves a large number of operations, storage requirements may become large when using the reverse mode of automatic differentiation. Moreover, in many applications the objective function $f(\mathbf{x})$ is often badly behaved and has several local and global minima[1]. Hence, methods that are able to search locally as well as globally have to be developed for finding accurate solutions of the problem defined in (1.1a)-(1.1d) within as few function evaluations as possible, and derivative-free optimization methods have proved successful also in this respect [30].

1.2 Literature Review of Surrogate Model Algorithms

In order to solve problems of the type described in Section 1.1, surrogate models (also called response surface models, or metamodels) have been developed in the literature. Generally speaking, a surrogate model is a model of

[1]A global minimum is defined as a feasible point $\mathbf{x}^*$ for which $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all feasible $\mathbf{x}$. A local minimum is defined as a feasible point $\mathbf{x}^*$ for which $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all feasible points $\mathbf{x}$ for which $\|\mathbf{x} - \mathbf{x}^*\|_2 < \epsilon$, where $\epsilon > 0$ and $\|\cdot\|_2$ denotes the Euclidean norm. A global minimum is therefore always also a local minimum.

a model. While the computationally expensive simulation models are used to describe complex physical phenomena, surrogate models are used to replace the computationally expensive simulation models [21]:

$$f(\mathbf{x}) = s(\mathbf{x}) + \epsilon(\mathbf{x}), \tag{1.2}$$

where $f(\mathbf{x})$ denotes the computationally expensive simulation model output, $s(\mathbf{x})$ denotes the prediction of the surrogate model at point $\mathbf{x}$, and $\epsilon(\mathbf{x})$ is the difference between the two.

The idea is to use the surrogate model $s(\mathbf{x})$ during the iterative optimization procedure instead of the true objective function $f(\mathbf{x})$ as much as possible because $s(\mathbf{x})$ is computationally cheap to evaluate, and thus the computation times can be considerably reduced. The surrogate model $s(\mathbf{x})$ is used to predict the objective function values of points in the variable domain, and this information is then exploited for determining promising points for doing the expensive function evaluations.

1.2.1 Global Surrogate Model Algorithms

The surrogate model algorithms widely used in the literature are so-called two-stage approaches [73], and consist in general of the following steps:

Algorithm 1 ***General global surrogate model algorithm***

1. *Create an initial experimental design and evaluate the computationally expensive objective function at the selected points.*
2. *Use the data from 1. to compute the parameters of the response surface.*
3. *Use the information from the response surface to determine the point for doing the next computationally expensive function evaluation.*
4. *Given the new data, update the response surface parameters.*
5. *Iterate through 3. and 4. until a given stopping criterion has been met.*

The initial experimental design in the first step can be created, for example, by Latin hypercube sampling, orthogonal arrays, factorial designs, or some other sampling strategy that seems useful for the given application [101]. In the second step, a surrogate model must be chosen. There are different kinds of surrogate models. Radial basis functions and kriging, for example, are interpolating models, whereas polynomial regression models and multivariate adaptive regression splines are non-interpolating. Each of these

models will be described in more detail in the following sections. In this thesis global surrogate models, i.e. surrogate models that are fit over the whole variable domain, are employed.

When determining the point for doing the next expensive function evaluation in the third step, it is important to use a strategy that is able to search locally as well as globally. During the local search, the neighborhood of promising points should be examined more thoroughly (exploitation) in order to find further objective function value improvements. On the other hand, the global search is necessary to escape from possible local optima, and to find new promising regions of the variable domain where the global optimum may be located (exploration). It is not sufficient to use an algorithm that only searches globally and fails to search locally. Once a promising point has been encountered, its close vicinity must be further examined in order to find possible improvements. This is especially of importance if the objective function has very steep minima which may easily be missed if the search is only global.

Several strategies have been developed to achieve a balance between local and global search. Gutmann [57], for example, minimizes a so-called "bumpiness" measure. A target value f^* for the objective function is defined, and the point of the variable domain that minimizes the bumpiness measure given the assumed value f^* becomes the next sample point.

Jones *et al.* [74] use a kriging surrogate model to approximate the objective function and exploit in the sampling strategy the estimate of the predictor error computed by the kriging model. Their criterion for selecting the next sample site is to maximize the expected improvement.

Regis and Shoemaker [124] developed a global search method, AQUARS, that uses a measure on how thoroughly the local minima of the response surface have already been explored. The next sample point is chosen either in the vicinity of the best partially explored or the least explored local minimum point of the response surface.

All of the aforementioned approaches select a sample point by optimizing an auxiliary function (minimize the bumpiness measure, maximize the expected improvement, minimize the response surface), which is in general itself a global optimization problem because it cannot be guaranteed that these auxiliary optimization problems are unimodal. However, because the optimization is done on the cheap-to-evaluate response surface, this step is

computationally inexpensive.

Regis and Shoemaker [119] also developed a stochastic sampling strategy. In this approach a large number of candidate points for the next expensive objective function evaluation is generated, and based on a weighted score derived from the objective function value prediction of the response surface and the distance of the candidates to the set of already sampled points, the candidate point with the best score is selected for doing the next expensive function evaluation. The advantage of this approach is that no auxiliary optimization problem has to be solved and savings in computation times can be achieved especially when high-dimensional problems are considered. Moreover, the candidate point approach has been shown to often perform superior as compared to the approaches where an auxiliary optimization problem has to be solved [119].

The stopping criteria in Step 5 of Algorithm 1 can be, for example, a given maximum number of allowed function evaluations, a maximum CPU time, a maximum number of unsuccessful consecutive improvement trials, or some other algorithm-specific criterion. The algorithm by Jones *et al.* [74], for example, was stopped when the maximum expected improvement was less than 1%.

The surrogate models used in this thesis are described in more detail in the following sections. For a summary of global surrogate model based algorithms, see, for example, Jones [73], and Shan and Wang [130]. In the following, the n distinct points where the objective function $f(\mathbf{x})$ has already been evaluated are denoted by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, and the corresponding objective function values by $y_1, y_2, \ldots, y_n$.

1.2.2 Polynomial Regression Models

Polynomial regression models are non-interpolating and widely used in engineering disciplines [101]. A linear regression model is in general defined as

$$s_p(\mathbf{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i \tag{1.3}$$

where x_i is the ith component of $\mathbf{x}$. The least squares estimator $\hat{\boldsymbol{\beta}}$ for the parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)^T$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.4}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ is the vector of objective function values, and

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_1^T \\ 1 & \mathbf{x}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^T \end{bmatrix} \tag{1.5}$$

is the matrix of sample sites augmented with a column vector of ones for the intercept β_0.

The regression model can as well be extended to higher orders. A quadratic regression model, for example, is defined as

$$s_p(\mathbf{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \beta_{ij} x_i x_j. \tag{1.6}$$

By their definition, quadratic models can capture the curvature of the true model and interactions between variables better than linear models.

Polynomial regression models have been used in the literature, for example, by Hosder *et al.* [66] for solving a multidisciplinary design optimization problem in high speed civil transport. Liao *et al.* [86] used a quadratic polynomial as response surface for solving a multiobjective optimization problem about crash safety design of vehicles. Quadratic polynomials have also been used by Madsen *et al.* [90] to optimize the design of incompressible diffusers.

1.2.3 Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) [49] are non-interpolating and non-parametric models that are able to automatically model nonlinearities and variable interactions. The general MARS model is defined as

$$s_m(\mathbf{x}) = \sum_{l=1}^{L} a_l B_l(\mathbf{x}), \tag{1.7}$$

where a_l are constant coefficients and B_l are basis functions that can either be a constant, a hinge function, or the product of two or more hinge functions. Pairs of hinge functions have the form

$$\max\{0, \mathbf{c} - \mathbf{x}\} \quad \text{and} \quad \max\{0, \mathbf{x} - \mathbf{c}\}, \tag{1.8}$$

where $\mathbf{c}$ denotes the so-called knot.

The MARS model is built in two stages, namely a forward and a backward pass. In the forward pass pairs of basis functions are repeatedly added to the model such that the sum of squared residual errors is reduced as much as possible. During the backward pass the model is pruned, and at each step the least effective term is deleted from the model according to the generalized cross-validation criterion.

1.2.4 Radial Basis Functions

The radial basis function (RBF) models [40, 110] used in this thesis are interpolating models. The advantage of RBFs is their ability to model curvature well. The RBF surrogate model is defined by

$$s_b(\mathbf{x}) = \sum_{\iota=1}^{n} \lambda_\iota \phi\left(\|\mathbf{x} - \mathbf{x}_\iota\|_2\right) + p(\mathbf{x}), \tag{1.9}$$

where $\|\cdot\|_2$ denotes the Euclidean norm, $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}$ are parameters to be determined, $\phi : \mathbb{R}_+ \mapsto \mathbb{R}$ is a radial basis function. Commonly used RBFs are summarized in Table 1.1.

Table 1.1: Commonly used RBF models, $\rho > 0$.

Name	$\phi(d)$
Cubic	d^3
Thin plate spline	$d^2 \log d$
Gaussian	$\exp\left(-\frac{d^2}{\rho^2}\right)$
Multiquadric	$\sqrt{d^2 + \rho^2}$
Inverse-multiquadric	$(d^2 + \rho^2)^{-1/2}$

$p \in \mathcal{P}^n_{d-1}$, where $\mathcal{P}^n_{d-1}$ is the space of polynomials in k variables and of total degree at most $d-1$. The polynomial tail ensures uniqueness of the model, and that the model belongs to a linear space that also contains the polynomial space $\mathcal{P}^n_{d-1}$ (trivial if $d = 0$) [149]. In the case of the cubic radial basis function, for example, $p(\mathbf{x}) = \mathbf{b}^T\mathbf{x} + a$ with $\mathbf{b} \in \mathbb{R}^k$, and $a \in \mathbb{R}$. The unknown parameters $\lambda_1, \ldots, \lambda_n$, $\mathbf{b}$ and a are obtained by solving the system

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}, \tag{1.10}$$

where

$$\mathbf{P} = \begin{bmatrix} \mathbf{x_1}^T & 1 \\ \mathbf{x_2}^T & 1 \\ \vdots & \vdots \\ \mathbf{x_n}^T & 1 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix}, \quad \boldsymbol{\gamma} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \\ a \end{bmatrix}, \tag{1.11}$$

and $\Phi_{\iota\nu} = \phi(\|\mathbf{x}_\iota - \mathbf{x}_\nu\|_2)$, $\iota, \nu = 1, \ldots, n$, and $\mathbf{0}$ is a matrix with all entries 0 of appropriate dimension. If $\text{rank}(\mathbf{P}) = k + 1$, then the matrix in equation (1.10) is nonsingular, and the system (1.10) has a unique solution [110]. Therefore, it is possible to obtain a unique radial basis function interpolant for the true function f.

Several algorithms using radial basis function models have been developed [18, 19, 57, 64, 118, 119, 120, 121, 122, 124, 149]. Li *et al.* [84] used radial basis function surrogate models for optimizing their injection molding process, and an improved expected improvement criterion has been used for guiding the search for promising points in the variable domain. RBFs have also been used by Regis and Shoemaker [119] for solving a groundwater bioremediation application problem, and by Regis [118] to solve a large dimensional problem from the automotive industry.

1.2.5 Kriging

Kriging is an interpolating surrogate model and incorporates a stochastic component that may be exploited by global optimization algorithms [74]. Kriging was introduced by Matheron [92], and has been used in numerical experiments for example by Currin *et al.* [32] and Jones *et al.* [74].

Kriging models consist of two components. The first component is some simple model that captures the trend in the data, and the second component measures the deviation between the simple model and the true function. Define the sample site matrix $\mathbf{S} = (\mathbf{x_1}, \ldots, \mathbf{x_n})^T$. The data is shifted and scaled such that the mean over every column in $\mathbf{S}$ and the mean of the objective function values in $\mathbf{y}$ equal zero, and that the variances equal 1 [88].

A model s_k is built as a realization of a regression model $\mathcal{F}$ and a random function z in order to express the deterministic response y for the input vector $\tilde{\mathbf{x}} \in \Omega$, where Ω denotes the variable domain. The model can be written as

$$s_k(\tilde{\mathbf{x}}) = \mathcal{F}(\boldsymbol{\psi}, \tilde{\mathbf{x}}) + z(\tilde{\mathbf{x}}), \tag{1.12}$$

where the regression model $\mathcal{F}$ is a linear combination of q functions $\varphi_\jmath : \mathbb{R}^k \mapsto \mathbb{R}$, $\jmath = 1, \ldots, q$,

$$\mathcal{F}(\boldsymbol{\psi}, \tilde{\mathbf{x}}) = \psi_1\varphi_1(\tilde{\mathbf{x}}) + \psi_2\varphi_2(\tilde{\mathbf{x}}) + \ldots + \psi_q\varphi_q(\tilde{\mathbf{x}}), \tag{1.13}$$

and can be written in matrix notation as

$$\mathcal{F}(\boldsymbol{\psi}, \tilde{\mathbf{x}}) = \boldsymbol{\varphi}(\tilde{\mathbf{x}})^T\boldsymbol{\psi}, \tag{1.14}$$

where $\boldsymbol{\varphi}(\tilde{\mathbf{x}}) = (\varphi_1(\tilde{\mathbf{x}}), \ldots, \varphi_q(\tilde{\mathbf{x}}))^T$ is the vector of functions, and $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_q)^T$ is the vector of regression parameters. The random process z is assumed to have zero mean, and covariance

$$\mathbb{V}\left[z(\mathbf{w}), z(\tilde{\mathbf{x}})\right] = \sigma^2\mathcal{R}(\boldsymbol{\theta}, \mathbf{w}, \tilde{\mathbf{x}}) \tag{1.15}$$

between $z(\mathbf{w})$ and $z(\tilde{\mathbf{x}})$. Here σ^2 is the process variance and $\mathcal{R}(\boldsymbol{\theta}, \mathbf{w}, \tilde{\mathbf{x}})$ is the correlation model depending on the parameters $\boldsymbol{\theta}$ that have to be optimized.

The Matlab toolbox DACE [88] allows the choice of different correlation functions (Table 1.2) and regression polynomials (Table 1.3). The parameters $\boldsymbol{\theta}$ are optimized with a modified version of the Hooke-Jeeves pattern search method. The considered correlations are products of stationary, one-dimensional correlations:

$$\mathcal{R}(\boldsymbol{\theta}, \mathbf{w}, \tilde{\mathbf{x}}) = \prod_{i=1}^{k} \mathcal{R}_i(\theta_i, w_i - \tilde{x}_i).$$

Table 1.2: Correlation functions of the Matlab toolbox DACE.

Name	$\mathcal{R}_i(\theta_i, d_i)$, $d_i = w_i - \tilde{x}_i$
Exponential	$\exp(-\theta_i\lvert d_i\rvert)$
General exponential	$\exp(-\theta_i\lvert d_i\rvert^{\theta_{n+1}})$, $0 < \theta_{n+1} \leq 2$
Gaussian	$\exp(-\theta_i d_i^2)$
Linear	$\max\{0, 1 - \theta_i\lvert d_i\rvert\}$
Spherical	$1 - 1.5\xi_i + 0.5\xi_i^3$, $\xi_i = \min\{1, \theta_i\lvert d_i\rvert\}$
Cubic	$1 - 3\xi_i^2 + 2\xi_i^3$, $\xi_i = \min\{1, \theta_i\lvert d_i\rvert\}$
Spline ($\xi_i = \theta_i\lvert d_i\rvert$)	$1 - 15\xi_i^2 + 30\xi_i^3$ for $0 \leq \xi_i \leq 0.2$ $1.25(1 - \xi_i)^3$ for $0.2 < \xi_i < 1$ 0 for $\xi_i \geq 1$

Table 1.3: Regression models of the Matlab toolbox DACE.

Type	Functions
Constant, $q = 1$	$\varphi_1(\tilde{\mathbf{x}}) = 1$
Linear, $q = k + 1$	$\varphi_1(\tilde{\mathbf{x}}) = 1, \varphi_2(\tilde{\mathbf{x}}) = \tilde{x}_1, \ldots, \varphi_{k+1}(\tilde{\mathbf{x}}) = \tilde{x}_k$
Quadratic, $q = \frac{1}{2}(k+1)(k+2)$	$\varphi_1(\tilde{\mathbf{x}}) = 1,$ $\varphi_2(\tilde{\mathbf{x}}) = \tilde{x}_1, \ldots, \varphi_{k+1}(\tilde{\mathbf{x}}) = \tilde{x}_k,$ $\varphi_{k+2}(\tilde{\mathbf{x}}) = \tilde{x}_1^2, \ldots, \varphi_{2k+1}(\tilde{\mathbf{x}}) = \tilde{x}_1\tilde{x}_k,$ $\varphi_{2k+2}(\tilde{\mathbf{x}}) = \tilde{x}_2^2, \ldots, \varphi_{3k} = \tilde{x}_2\tilde{x}_k,$ $\ldots \quad \varphi_q(\tilde{\mathbf{x}}) = \tilde{x}_k^2$

Denote by $\mathbf{F} = (\boldsymbol{\varphi}(\mathbf{x}_1), \ldots, \boldsymbol{\varphi}(\mathbf{x}_n))^T$ the $n \times q$ design matrix with entries $F_{\iota\jmath} = \varphi_\jmath(\mathbf{x}_\iota)$, $\iota = 1, \ldots, n$, $\jmath = 1, \ldots, q$. Let $\mathbf{R}$ be the matrix of stochastic process correlations between the z's at the design sites,

$$R_{\iota\nu} = \mathcal{R}(\boldsymbol{\theta}, \mathbf{x}_\iota, \mathbf{x}_\nu), \ \iota, \nu = 1, \ldots, n.$$

At the unsampled point $\tilde{\mathbf{x}}$ the vector of correlations between the z's at the design sites $\mathbf{x}_\iota$, $\iota = 1, \ldots, n$, and $\tilde{\mathbf{x}}$ is given by

$$\mathbf{r}(\tilde{\mathbf{x}}) = (\mathcal{R}(\boldsymbol{\theta}, \mathbf{x}_1, \tilde{\mathbf{x}}), \ldots, \mathcal{R}(\boldsymbol{\theta}, \mathbf{x}_n, \tilde{\mathbf{x}}))^T.$$

It can be shown that for the prediction $s_k(\tilde{\mathbf{x}})$ it holds that

$$s_k(\tilde{\mathbf{x}}) = \boldsymbol{\varphi}(\tilde{\mathbf{x}})^T \boldsymbol{\psi}^* + \mathbf{r}(\tilde{\mathbf{x}})^T \boldsymbol{\gamma}^*,$$

where $\boldsymbol{\psi}^*$ is the generalized least squares solution to

$$\mathbf{F}\boldsymbol{\psi} = \mathbf{y},$$

and thus

$$\begin{aligned} \boldsymbol{\psi}^* &= (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y} \\ \boldsymbol{\gamma}^* &= \mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\boldsymbol{\psi}^*), \end{aligned}$$

where $\mathbf{R}$ and $\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}$ are nonsingular. Thus, $\boldsymbol{\varphi}(\tilde{\mathbf{x}})^T\boldsymbol{\psi}^*$ can be interpreted as an approximation of $\mathcal{F}(\boldsymbol{\psi}, \tilde{\mathbf{x}})$, and $\mathbf{r}(\tilde{\mathbf{x}})^T\boldsymbol{\gamma}^*$ as some average approximation of $z(\tilde{\mathbf{x}})$.

Kriging models have been used for solving global optimization problems by Horowitz *et al.* [65], for example, who developed a parallel version of Jones' efficient global optimization (EGO) algorithm [74], and used it to optimize

a polymer injection strategy. The book by Forrester *et al.* [48] is a very thorough introduction to engineering design using surrogate models and features several application examples from engineering disciplines. Moreover, the accompanying Matlab codes are freely available online. Glaz *et al.* [52] studied the influence of different surrogate models on the solution to an optimization problem about the design of helicopter rotor blades with the goal of vibration reduction. Polynomial regression surrogates, RBFs, and kriging have been compared, and kriging was found to be the best model for this type of problem. Kriging has also been used by Jouhaud *et al.* [75] for solving a multidisciplinary shape optimization problem of a subsonic airfoil. Lam *et al.* [81] used the kriging model to solve an aerostructural design optimization problem. Morgans *et al.* [96] developed a kriging based algorithm for optimizing the shape of their horn-loaded loudspeakers. Yang *et al.* [155] used kriging for solving a frontal impact design optimization problem.

1.3 Contribution and Organization

In this section the content of the individual chapters is briefly summarized and the main contributions outlined. The goal in Chapters 2 and 3 is to develop mixture surrogate model algorithms in order to answer the question

Which surrogate model should be used for a given problem?

The developed algorithms SO-M, SO-M-s, and SO-M-c use Dempster-Shafer theory to determine the influence each individual model should have in the mixture. In Chapter 3 various sampling strategies and mixture models are compared in a numerical study to the efficient global optimization algorithm (EGO) by Jones *et al.* [74] and to Gutmann's RBF method [57].

In Chapters 4 and 5 the question

Can surrogate models be used to efficiently find good approximations of global optima of problems where integrality constraints are imposed on some or all variables?

is addressed. This is an important topic since many application problems have both continuous and discrete variables, but this has barely been studied in the literature. The algorithms SO-MI for mixed-integer problems and SO-I for purely integer problems have been developed in order to answer the question. Both algorithms deal with computationally expensive constraints by using a penalty approach.

In Chapter 6 the question

How can the agricultural land use of an upstate New York watershed be managed to reduce the phosphorus runoff at minimal cost?

has been studied. In this real world application problem all variables are discrete, and the problem has one constraint whose value is obtained from the same computationally expensive simulation as the objective function value. The algorithm SO-I has been extended to the algorithm SO-Ic that treats the constraint directly by using a response surface rather than with a penalty approach.

Finally, Chapter 7 concludes the thesis, and gives possible future research directions. The Appendices A-D contain information about the test problems used in the numerical experiments.

1.3.1 Summary of Chapter 2: Algorithm SO-M

When optimizing black-box functions, it is in general a priori unknown which surrogate model will be the most successful for finding an accurate solution for a given application problem. While, for example, a kriging model may perform well for some problem, it might perform poorly on others. In Chapter 2 an algorithm, SO-M, is developed that tries to answer the question of which surrogate model should be applied for a given problem by using mixture surrogate models.

SO-M fits several surrogate models to the sample data. Model characteristics such as correlation coefficients and various error measures are computed using leave-one-out cross-validation to determine which models are "good" (high correlation coefficients, low errors) and which models are "bad" (low correlation coefficients, high errors). A good model should have high influence on the prediction of a mixture model, whereas models considered bad should have no or only very little influence. The prediction of a mixture surrogate model at point $\mathbf{x}$ is a weighted sum of the predictions of individual surrogate models and it is defined by

$$s_{\text{mix}}(\mathbf{x}) = \sum_{r \in \mathcal{M}} w_r s_r(\mathbf{x}), \quad \text{where} \quad \sum_{r \in \mathcal{M}} w_r = 1, \tag{1.16}$$

and where $w_r \geq 0$ is the weight of the rth model in the mixture, $\mathcal{M}$ denotes the set of models in the mixture, and $s_r(\mathbf{x})$ is the prediction of the rth surrogate model at point $\mathbf{x}$. By this definition it is favorable to give large

weights, and therefore a high influence, to "good" surrogate models, whereas "bad" models should have low weights and thus only very low or no influence on the prediction of the mixture model.

Difficulties arise, however, if a model has, for example, high correlation coefficients but also large errors as compared to the other models. In this case it is difficult to decide whether the considered model should get a large weight because it has high correlation coefficients or a low weight because it has large errors. This *conflicting* model information has been treated by using Dempster-Shafer theory [37, 129], which is a mathematical theory of evidence that allows the combination of conflicting information from different sources.

The mixture surrogate model algorithm, SO-M (**S**urrogate **O**ptimization - **M**ixture), has been implemented with various rules for combining the model information and redistributing possible conflicts (Dempster's rule [37], Yager's rule [154], Inagaki's rule [70], proportional conflict redistribution [132]). The different algorithm versions have been compared in numerical experiments on six of the Dixon and Szegö [39] test problems (two to six dimensions). The results of these numerical experiments show that the approach of using mixture models is in general very promising, especially as the problem dimension increases. The mixture model algorithm is able to explore the variable domain, and for all but one test problem accurate approximations of the global optima have been found. The material of Chapter 2 has been published in the *Journal of Global Optimization*, Vol. 51, pages 79-104 [98].

1.3.2 Summary of Chapter 3: Algorithms SO-M-s and SO-M-c

Another major component of surrogate model algorithms is the strategy for selecting promising points in the variable domain for doing the next expensive function evaluation. The goal is to sample in the vicinity of points that have been found to have low objective function values in the hope to further improve the solution (local search, exploitation). However, since many problems are multimodal, there is a potential risk of getting stuck in a local optimum. Thus, a good sampling strategy should be able to escape from local optima and globally explore the whole variable domain in or-

der to find other promising regions where the global optimum may be located.

As briefly outlined in Section 1.2.1, there are various methods that can be used as sampling strategy. An auxiliary function, such as the bumpiness measure [57], the expected improvement [74], or the response surface [124], could be optimized. On the other hand, a random sampling approach [119] that does not require solving a global optimization subproblem could be applied. This topic is addressed in Chapter 3. SO-M has been improved with respect to its efficiency of computing the model characteristics ($\tilde{k}$-fold cross-validation instead of leave-one-out cross-validation), and two sampling strategies have been examined, namely a stochastic approach similar to the one by Regis and Shoemaker [119] (algorithm SO-M-c (**S**urrogate **O**ptimization - **M**ixture - **c**andidate sampling)), and an approach that uses the minimum of the response surface (algorithm SO-M-s (**S**urrogate **O**ptimization - **M**ixture - **s**urface minimum)).

SO-M-c and SO-M-s are used with various mixture surrogate models in order to examine the influence of specific models on the mixture. The algorithms are compared to EGO [74] and Gutmann's RBF method [57] on a wide range of literature test problems with up to 30 dimensions and two application problems. One application deals with groundwater bioremediation [158] where the goal is to determine a pumping strategy for cleaning up contaminated groundwater at minimal costs. The other application problem deals with energy generation using tethered kites [8, 22, 45, 67] where the design variables represent parameters for controlling the kite such that the net power produced is maximized. Both application problems involve differential equations.

The results show that there is no algorithm that performs best for all test problems (No Free Lunch theorem [151]), but comparing the performance of the algorithms after an equal number of function evaluations showed that SO-M-c performs on average better for problems of dimension 12 and larger, whereas SO-M-s performs better for low-dimensional problems. Gutmann's method performs in general worst, and EGO's performance varied between problem classes. The study showed, however, also that independent of the sampling strategy it is advisable to use mixture surrogate models rather than individual models. If it is not known beforehand which model will perform best, mixture models help to prevent selecting the worst model. Moreover, mixtures containing RBF models were found to perform in general well, whereas using only a polynomial model should be avoided.

Within the scope of this study a Matlab toolbox that allows the user to choose between different initial experimental design strategies, (mixture) surrogate models, and sampling strategies has been developed, and is freely available upon request from the author or on Matlab File Exchange ("Surrogate Model Optimization Toolbox", File ID #38530).

The material of Chapter 3 has been presented at the Global Optimization Workshop 2012 in Natal, Brazil [100].

1.3.3 Summary of Chapter 4: Algorithm SO-MI

The algorithms in Chapters 2 and 3 have been developed for black-box optimization problems with box-constraints and continuous variables only. In many application problems, such as engineering design optimization or reliability engineering, also integer variables are encountered [27, 36, 72, 80, 115, 136]. Moreover, application problems often have black-box constraints whose function values are computed within the same simulation model as the objective function. For example, when designing a truss structure the goal may be to determine the length (continuous variables) and the cross sectional area (discrete variables) of each truss member such that the total structural weight is minimized and constraints on nodal displacements and stresses are satisfied. Computing displacements and stresses requires a finite element analysis, which becomes computationally expensive as the number of truss members increases [36, 72, 115, 126, 136].

Although surrogate model based algorithms have been used extensively to solve continuous optimization problems, only very few papers deal with mixed-integer problems [14, 33, 60, 64, 87, 116]. A new algorithm, SO-MI (**S**urrogate **O**ptimization - **M**ixed **I**nteger), is introduced in Chapter 4 and is meant to extend the research in the area of mixed-integer global optimization where the objective function is computationally expensive to evaluate, and where black-box constraints may be present.

SO-MI uses a cubic radial basis function interpolant as surrogate model and a stochastic sampling strategy. Algorithms developed so far for mixed-integer global optimization problems with computationally expensive objective functions solve optimization subproblems on the computationally cheap response surface. The required MINLP solvers used in these subroutines may however become a computational burden as the number of variables increases. Thus, an alteration of the candidate point approach for continuous

variables has been used and avoids solving an optimization subproblem. In every iteration four groups of candidates are generated, and from each group the best point is selected for doing the expensive objective (and constraint) function evaluation in parallel. Note that although some variables have to be integers, all variables are assumed to be continuous in order to obtain a smooth response surface. The candidate points, and therefore also the sample points, are, however, generated such that the integrality constraints are satisfied. Computationally expensive constraints are treated with a penalty approach where the penalty factor is adjusted dynamically during the optimization process. It can be shown that SO-MI converges to the global optimum almost surely.

The performance of SO-MI has been compared in numerical experiments to NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) [2, 3, 4] and to two commonly used algorithms for solving mixed-integer problems, namely a genetic algorithm and a branch and bound method. An extensive study comparing NOMAD for mixed-integer problems with other derivative-free algorithms has not been done yet in the literature. Sixteen literature test problems and five instances of two types of application problems have been used in the numerical experiments. One type of application problem arises in structural optimization (two truss structures have been optimized), and the other application area is reliability design, where three reliability-redundancy allocation problems have been examined. The results show that SO-MI outperforms the other algorithms for most literature test problems and all application problems. Also NOMAD performs very well compared to the genetic algorithm and branch and bound, but is often outperformed by SO-MI.

The content of Chapter 4 has with minor changes been accepted for publication in the journal *Computers & Operations Research* [99].

1.3.4 Summary of Chapter 5: Algorithm SO-I

Computationally expensive optimization problems with only integer variables are considered in Chapter 5. This type of problems is encountered in many application areas. For example, in structural optimization there are problems where the locations of the truss nodes are fixed, and only the cross sectional areas or wall thicknesses of the members have to be determined. Similarly, in reliability engineering, purely discrete optimization problems arise when the system reliability is increased only by adding redundancy

rather than increasing also the reliability of individual components (redundancy allocation problems) [80, Chapters 3-5]. Furthermore, throughput maximization problems are often integer problems [108]. These applications are encountered, for example, in production planning and facility layout [51, 134], but also in the design of networks-on-chip routers [69]. Another purely integer optimization problem arises in hydropower energy generation [24, 85] where several turbines of different efficiencies are installed in a hydropower plant, and the question is how many turbines of the same kind should be in use to maximize the total generated power given a certain amount of water that can be released from the reservoir.

The SO-MI algorithm introduced in Chapter 4 has been further developed to solve purely integer problems. The algorithm SO-MI has been changed in two major aspects. A first optimization phase has been introduced in which the new algorithm, SO-I (**S**urrogate **O**ptimization - **I**nteger), minimizes a constraint violation function in order to find a first feasible point. Thus, in contrast to SO-MI, SO-I does not require an initially given feasible point. Secondly, the generation of candidate points has been adapted for purely integer problems. Candidate points are generated by perturbing the best point found so far with integer increments, and by uniformly selecting integer points from the whole box-constrained variable domain. Scoring criteria are used to select the one best candidate point in every iteration. Black-box constraints, if present, are treated with a penalty approach that works similarly to a barrier method.

The efficiency of SO-I has been compared in numerical experiments to a genetic algorithm, a branch and bound algorithm for nonlinear problems, and NOMAD on 17 generic test problems with a wide variety of characteristics, two throughput maximization problem instances, and six hydropower generation application problems. SO-I performed significantly better than all other algorithms for almost all test problems. Only NOMAD was able to find better solutions for four test problems, and branch and bound performed better for two problems with linear objective functions where optimizing the relaxed subproblem in the tree nodes directly led to an integer feasible solution which was also the globally optimal solution. The convergence of SO-I follows from a simple counting argument since the cardinality of the box-constrained variable domain is finite and no point is sampled more than once.

In the numerical experiments a major drawback of using branch and bound for black-box problems has been encountered when trying to solve the application problems. The throughput maximization simulation code failed

when variables assumed continuous values as is the case when optimizing the relaxed subproblems in the tree nodes of branch and bound in order to compute lower bounds for the objective function value. Also the hydropower generation simulation failed to compute an objective function value when some integer variable setting did not allow a feasible solution with respect to the total allowed water outflow from the reservoir. In contrast to the other algorithms, branch and bound cannot deal with such "missing" objective function values, and hence fails finding a solution. Moreover, if the relaxed subproblems are multimodal, there is no guarantee that the lower bounds computed by branch and bound (if possible) are valid. Therefore, pruning decisions may be wrong if the lower bounds are not derived from the global minimum of the relaxed subproblem.

1.3.5 Summary of Chapter 6: Algorithm SO-Ic

An application problem arising in the agricultural land use management of the Cannonsville reservoir watershed in upstate New York is considered in Chapter 6. The problem is to determine the optimal locations of land conversion (or land retirement) in order to reduce the amount of total phosphorus runoff in the watershed. Land conversion is a best management practice (BMP) employed in watersheds that drain to bodies of water that are particularly sensitive to agricultural runoff. This particular BMP involves contracts between government agencies and local land owners to remove certain land from crop production. In exchange for retiring the land from crop production, the government agencies agree to provide some percentage of setup costs, and yearly rental and maintenance payments. Since the cost of these projects can be quite high, the goal is to minimize the total conversion costs while keeping the total phosphorus runoff below a given threshold. The total conversion costs and phosphorus runoff must be computed with a highly nonlinear and computationally expensive simulation model that has been supplied by Joshua Woodbury from Cornell University.

The algorithm SO-I described in Chapter 5 has been further developed with respect to the constraint handling to solve this particular application problem. The algorithm, SO-Ic (**S**urrogate **O**ptimization - **I**nteger **c**onstraints), uses a cubic radial basis function interpolant as surrogate model for approximating the objective *and* the constraint function. In the first optimization phase the response surface of the constraint is used for minimizing the computationally expensive phosphorus constraint in order to find a first feasible solution. In the second optimization phase the response

surface for the constraint is used to discard infeasible-predicted candidate points, and thus it is more likely that points chosen for the computationally expensive simulations are feasible.

The performance of SO-Ic on this application problem has been compared to NOMAD, a genetic algorithm, and the discrete dynamically dimensioned search (discrete-DDS) algorithm [140]. Branch and bound could not be used for this application problem because the code for evaluating the objective and constraint function fails if the input variable vector contains continuous variables, which is the case whenever branch and bound tries to optimize a relaxed subproblem to compute lower bounds for the objective function value. The numerical experiments on three problem instances with different upper bounds for the allowable phosphorus content in the water showed that SO-Ic achieves significantly better results than the other algorithms. SO-Ic also has the lowest mean standard errors, which indicates that SO-Ic is a rather robust algorithm. NOMAD performed comparatively poorly on all problem instances, and was not able to find a feasible solution for several trials. The numerical experiments also showed that the cost increase does not behave linearly with the phosphorus reduction goals. Increasing the phosphorus reduction goal from 20% to 40% of the base case invokes significantly fewer additional costs than increasing the reduction goal from 40% to 60%.

The material of Chapter 6 is joint work with Joshua Woodbury from Cornell University who kindly provided the description of the problem and the cost benefit analysis.

Chapter 2

SO-M: A Mixture Surrogate Model Algorithm for Global Optimization Problems Using Dempster-Shafer Theory

Abstract

Research in algorithms for solving computationally expensive global optimization problems using surrogate models has shown that it is in general not possible to use the same type of surrogate model for solving different kinds of problems. While a radial basis function model may be the most suitable for some problems, for others the best results may be obtained with a polynomial regression model. In this chapter the approach of applying Dempster-Shafer theory to surrogate model selection and their combination is introduced. Cross-validation is used to compute various model characteristics, and, for dealing with conflicting characteristics, different conflict redistribution rules have been examined with respect to their influence on the results in numerical experiments. Furthermore, the effect of the surrogate model type, i.e. using mixture models, single models or a hybrid of both, has been studied. The various versions of the algorithm, SO-M, are compared on six well-known global optimization test problems from the Dixon and Szegö [39] test bench. The results indicate that SO-M is able to thoroughly explore the variable domain for all problems, and the vicinities of global optima could be detected. The global minima could except for one test problem be approximated with high accuracy within 150 or fewer function evaluations.

Abbreviations and Nomenclature

ARS	Accelerated random search
BPA	Basic Probability Assignment
BetP	Pignistic probability
CC	Correlation coefficient
D	Dempster's rule
DST	Dempster-Shafer Theory
I	Inagaki's rule
K	Kriging model
M, MARS	Multivariate adaptive regression spline
MAD	Median absolute deviation
MAE	Maximum absolute error
P	Polynomial regression model
PCR	Proportional conflict redistribution rule
R, RBF	Radial basis function surrogate model
RMSE	Root mean squared error
SO-M	Surrogate Optimization - Mixture
Y	Yager's rule
$f(\cdot)$	Objective function, see equation (2.1a)
$\mathbf{x}$	Continuous variable vector
$\mathbf{x}^T$	Transpose of $\mathbf{x}$
$\mathbb{R}$	Real numbers
x_i	ith continuous variable, $i = 1, \ldots, k$, see equation (2.1c)
x_i^l, x_i^u	Lower and upper bound for ith continuous variable, see equation (2.1b)
k	Problem dimension, see equation (2.1c)
Ω	Variable domain
s_{mix}	Mixture surrogate model, see equation (2.2)
w_r	Weight for the rth surrogate model in the mixture, $r \in \mathcal{M}$, see equation (2.2)
$\mathcal{M}$	Set of models contributing to the mixture, see equation (2.2)
s_r	rth surrogate model in the mixture, $r \in \mathcal{M}$, see equation (2.2)
s_{min}	Minimum of the response surface
f_{max}, f_{min}	Maximal and minimal objective function values found so far

2.1 Introduction

In this chapter box-constrained global optimization problems of the following form are considered:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) && \text{(2.1a)} \\ \text{s.t.} \quad & -\infty < x_i^l \leq x_i \leq x_i^u < \infty, && \text{(2.1b)} \\ & x_i \in \mathbb{R}, i = 1, 2, \ldots, k. && \text{(2.1c)} \end{aligned}$$

It is assumed that evaluating the objective function $f(\mathbf{x})$ requires a time consuming simulation model, and thus a good approximation of the global minimum should be found within as few function evaluations as possible. Moreover, the algebraic form of $f(\mathbf{x})$ is unknown (black-box). In many applications $f(\mathbf{x})$ is multimodal, and therefore an algorithm that is able to search locally as well as globally is needed. The box-constrained variable domain defined in equation (2.1b) will in the following be denoted by $\Omega \subset \mathbb{R}^k$.

As stated in Chapter 1, surrogate models have been developed in order to reduce the necessary number of costly simulations while searching for the global minimum [52, 75, 84, 86, 114, 155]. Surrogate models have for example been used by Glaz *et al.* [52] during the optimization of helicopter rotor blades, and also by Queipo *et al.* [114] who optimized a liquid-rocket injector with respect to multiple objectives. Efficient global optimization (EGO) has been applied by Morgans *et al.* [96] in order to optimize the shape of horn-loaded loudspeakers. Surrogate models have also been used in automotive design (see, for example, [86, 155, 163]).

However, as shown for example by Goel *et al.* [53] and Viana and Haftka [144], one surrogate model does not suit all kinds of problems, i.e. a certain surrogate model might perform very well for some problems, but poorly for others. If it is not known beforehand which surrogate model is the most suitable for the problem at hand, different models would have to be tried in order to find the most effective one. This approach is, however, not feasible due to restrictions on the computation time. Thus, the challenge is to either somehow determine the best surrogate model, or to adjust the influence of single surrogate models in mixtures such that good models have higher influence than bad models in order to obtain the best results. The prediction $s_{\text{mix}}(\mathbf{x})$ of such a mixture surrogate model can in general be represented as

$$s_{\text{mix}}(\mathbf{x}) = \sum_{r \in \mathcal{M}} w_r s_r(\mathbf{x}), \quad \text{where} \sum_{r \in \mathcal{M}} w_r = 1, \tag{2.2}$$

and where $s_r(\mathbf{x})$ denotes the prediction of the rth contributing model, $w_r \geq 0$ is the corresponding weight, and $\mathcal{M}$ is the set of surrogate models in the mixture.

Goel *et al.* [53] suggested different approaches for determining the weights of models in combinations. However, only one of the approaches allows emphasizing and restricting the influence of good and bad model characteristics, respectively. Moreover, in this approach parameters must be adjusted, which is in general a difficult task. Viana and Haftka [144] also considered mixture surrogate models and suggested the optimization of an auxiliary function in order to obtain an approximation matrix. This matrix is in turn used for determining the model weights, but it can lead to negative weights and weights larger than one, and thus results may become inaccurate.

In order to overcome the mentioned problems the choice and combination of surrogate models using Dempster-Shafer theory (DST) [37, 129] is introduced in this chapter. DST is a mathematical theory of evidence that provides means of combining information from different sources in order to construct a degree of belief. The theory allows the combination of imprecise and uncertain pieces of information that may even be conflicting. So-called basic probability assignments (BPA) contain information about certain hypotheses (focal elements[1]), and are combined to calculate the credibility of a given hypothesis. Three functions are usually associated with BPAs, namely the belief, plausibility, and pignistic probability (BetP) function.

In terms of surrogate models the BPAs can be derived, for example, from model characteristics such as correlation coefficients and various error measures. It is possible that one surrogate model has conflicting characteristics, i.e. good (e.g., high correlation coefficients, or low errors) and bad (e.g., high errors, or low correlation coefficients) characteristics simultaneously. This conflict must be taken into account when calculating the belief value for a given model. Several rules have been developed in the literature for dealing with such conflicting information. Dempster's rule of combination redistributes the conflict among all focal elements, regardless of which elements caused the conflict. However, as shown by Zadeh [161], the results of this approach may be counter-intuitive. Fairer conflict redistribution rules have been developed. The conflict can be assigned to the set reflecting complete ignorance [154] (Yager's rule), or one may apply the proportional

[1]Here a hypothesis would be, for example, "A mixture of RBF and MARS should be chosen".

conflict redistribution (PCR) rules [132] to redistribute the conflict among those focal elements that actually cause the conflict. The disadvantage of the latter approach is, however, the computational complexity that increases with the number of information sources. Inagaki [70] proposed a general parametric formulation for the redistribution of the conflict.

The goal of this chapter is to develop a surrogate model algorithm that uses Dempster-Shafer theory for choosing among different surrogate models the most suitable one for a given optimization problem and for finding the weights of single models contributing to mixture models as defined in equation (2.2). The considered surrogate models (polynomial regression models, MARS, RBF, kriging) have already been described in Section 1.2. Since it is in general unknown whether mixture models will be more successful than single models, the following alternative strategies are examined:

1. using the best mixture model for the first optimization steps, then switching to using always the best single surrogate model,
2. using only the best mixture model,
3. using only the best single surrogate model.

The remainder of this chapter is structured as follows. The algorithm SO-M (**S**urrogate **O**ptimization - **M**ixture) using DST is described in Section 2.2. The results of the numerical experiments on a subset of problems from the Dixon and Szegö [39] test bench are discussed in Section 2.3. The algebraic description of the examined test problems is given in Appendix A. Conclusions are drawn in Section 2.4.

2.2 SO-M: Mixture Surrogate Model Algorithm

SO-M starts by generating an experimental design using Latin hypercube sampling that maximizes the minimum distance between the design points. The minimum number of initial sample sites is problem dependent. In general, at least $k + 1$, where k is the problem dimension, initial design sites are required for building the surrogate models. However, since in a later stage leave-one-out cross-validation is used to assess the accuracy of the models, at least one additional initial sample site is required. Thus, if, for example, a two dimensional problem is considered,

at least $k+2 = 4$ sample points are needed in the initial experimental design.

Given the $k + 2$ sample sites, the expensive function values have to be computed. Next, model characteristics are computed by leave-one-out cross-validation. One sample site is left out from the experimental design, and the remaining sample sites and their corresponding function values are used to build the surrogate model, which is then used to re-predict the function value at the sample site that was left out. This is done for every surrogate model and every sample site, and thus for every sample site a prediction from every model is obtained. Since it is assumed that the objective function is black-box, it is unknown in which region of the variable domain the global optimum will be found. Hence, the goal is to determine which surrogate model fits all the data obtained so far best, and therefore not to influence the global search.

Correlation coefficients (CC), root mean squared errors (RMSE), maximal absolute errors (MAE), and median absolute deviation (MAD) have been chosen as model characteristics in SO-M. The number and type of model characteristics can of course be tailored to specific application problems. Correlation coefficients reflect how well the surrogate models capture the behavior of the true objective function. High correlation coefficients (values close to 1) indicate that the surrogate model is able to predict high values where the true objective function attains high values, and that the surrogate model predicts low values where the objective function attains low values. Error measures should in general be low and indicate how accurate the predictions of the surrogate models are. The model characteristics are calculated for every surrogate model based on the true and the re-predicted function values. Characteristics of a good model are high positive CC, and low RMSE, MAE, and MAD.

After the model characteristics have been calculated BPAs are computed for every model. For this purpose the model characteristics are scaled so that the sum over all models for each BPA (CC, RMSE, MAE, MAD) equals one and the non-negativity conditions of BPAs are satisfied. DST is applied to determine the pignistic probabilities for each model. Based on these values it can be decided which one of all considered models is the best, or in case mixture models are considered, which weight should be assigned to each contributing model.

CC, RMSE, MAE, and MAD must also be calculated for each mixture in order to determine the best mixture model. The goodness values are normalized to obtain the BPAs for each mixture:

$$m_r^{\mathrm{CC}} = \frac{\mathrm{CC}_r}{\sum_{j\in\mathcal{M}} \mathrm{CC}_j}, \qquad m_r^{\mathrm{RMSE}} = \frac{\frac{1}{\mathrm{RMSE}_r}}{\sum_{j\in\mathcal{M}} \frac{1}{\mathrm{RMSE}_j}},$$

$$m_r^{\mathrm{MAE}} = \frac{\frac{1}{\mathrm{MAE}_r}}{\sum_{j\in\mathcal{M}} \frac{1}{\mathrm{MAE}_j}}, \qquad m_r^{\mathrm{MAD}} = \frac{\frac{1}{\mathrm{MAD}_r}}{\sum_{j\in\mathcal{M}} \frac{1}{\mathrm{MAD}_j}},$$

where $\mathcal{M}$ is the set of models in the combination, and r is the index of the rth surrogate model in the combination. Four evidence sets are obtained, i.e. the models contained in each considered mixture build the focal elements, and the evidence sets are the BPAs for each model that will in turn be used in the decision theory.

In order to illustrate the above procedure consider the following example. Assume the four models described in Section 1.2 are available and denote the polynomial model by P, the RBF model by R, the kriging model by K, and the MARS model by M. After cross-validation the scaled characteristics of the single models are as follows:

$$\begin{array}{llll} m_{\mathrm{P}}^{CC} = 0.29, & m_{\mathrm{R}}^{CC} = 0.29, & m_{\mathrm{K}}^{CC} = 0.42, & m_{\mathrm{M}}^{CC} = 0 \\ m_{\mathrm{P}}^{RMSE} = 0.11, & m_{\mathrm{R}}^{RMSE} = 0.24, & m_{\mathrm{K}}^{RMSE} = 0.25, & m_{\mathrm{M}}^{RMSE} = 0.40, \\ m_{\mathrm{P}}^{MAE} = 0.10, & m_{\mathrm{R}}^{MAE} = 0.24, & m_{\mathrm{K}}^{MAE} = 0.25, & m_{\mathrm{M}}^{MAE} = 0.41, \\ m_{\mathrm{P}}^{MAD} = 0.17, & m_{\mathrm{R}}^{MAD} = 0.27, & m_{\mathrm{K}}^{MAD} = 0.26, & m_{\mathrm{M}}^{MAD} = 0.30. \end{array}$$

After applying DST the pignistic probabilities of the models are

$$\mathrm{BetP(P)} = 0.05,\ \mathrm{BetP(R)} = 0.40,\ \mathrm{BetP(K)} = 0.55,\ \mathrm{BetP(M)} = 0,$$

and thus, if a single model was to be used in the next step, the kriging model would be chosen because it has the highest pignistic probability. If, however, a mixture model is required, the given pignistic probabilities are used for calculating the weights of the contributing models. Assume that models P and R are supposed to be combined. Then the weights for both models are

$$w_p = \frac{\mathrm{BetP(P)}}{\mathrm{BetP(P)} + \mathrm{BetP(R)}} = \frac{1}{9},\ w_b = \frac{\mathrm{BetP(R)}}{\mathrm{BetP(P)} + \mathrm{BetP(R)}} = \frac{8}{9}, \qquad (2.3)$$

where w_p is the weight for the polynomial model, and w_b is the weight for the RBF model, respectively. Thus $w_p + w_b = 1$, and these weights are used in the computation of the mixture model prediction according to equation (2.2):

$$s_{\mathrm{mix}}(\mathbf{x}) = w_p s_p(\mathbf{x}) + w_b s_b(\mathbf{x}) = \frac{1}{9} s_p(\mathbf{x}) + \frac{8}{9} s_b(\mathbf{x}),$$

where $s_p(\mathbf{x})$ and $s_b(\mathbf{x})$ are the predictions of the polynomial and RBF model at the point $\mathbf{x}$, respectively.

In the algorithm every possible (one-,) two-, three- and four-model mixture is considered, and therefore the focal elements are {P}, {R}, {K}, and {M} for the single models, {P,R}, {P,K}, {P,M}, {R,K}, {R,M}, and {K,M} for the two-model combinations, {P,R,K}, {P,R,M}, {P,K,M}, and {R,K,M} for the three-model combinations, and {P,R,K,M} for the four-model combination. In case of mixture models BPAs must be calculated based on the cross-validation characteristics of the corresponding mixture.

In the next step, one of the conflict redistribution rules mentioned in Section 2.1 is applied, i.e. the evidences are combined, and for each mixture and single model the corresponding belief, plausibility, and pignistic probability are calculated. Then the (mixture) model with the highest plausibility value is chosen as the new response surface[2].

Next a new sample site must be chosen. The algorithm must guarantee a thorough local as well as global search, i.e. the algorithm must be able to find and explore promising valleys of the objective function, but it must also be prevented from getting stuck in a local optimum. The difficulty is to decide when to switch from the local search to global search and vice versa.

If local minima of the single surrogate models contributing to the mixture are very distant from each other, it might be an indication that there are several regions in the variable domain where local and global optima of the true objective function could be located, and thus sampling at these points may be favorable. However, if the local minima are close together, this approach may lead to a very local search.

In order to prevent the algorithm from excessively sampling in the vicinity of the current best point it is necessary to force the search away from regions where already many samples have been taken. For this purpose the distances $d_{\min}$ of every newly added sample site to the set of already sampled points are recorded, and in case a certain number of sample sites closer to each other than some predefined threshold distance d^* is exceeded, the algorithm must be prevented from adding more points in this already *densely sampled region*. The smaller the number of allowed close points is, the more global

[2]Note that the combination of all models, i.e. the universal set, always has plausibility and belief values of one, and must thus be considered separately.

the search becomes. On the other hand, increasing the maximal allowed number of close samples leads to a longer local search. Similarly, the smaller the threshold distance d^*, the more thorough the local search becomes.

The sample sites where the true function has already been evaluated are clustered according to a k'-means algorithm [89], where the number of clusters k' is determined dynamically. If a large cluster has been found, a densely-sampled area is defined based on the cluster's content. Denote $\mathbf{X}_{c_l} = (\mathbf{x}_{c_l,1}, \mathbf{x}_{c_l,2}, \ldots, \mathbf{x}_{c_l,l})^T$ the matrix of sample sites contained in the considered cluster c_l, where $\mathbf{x}_{c_l,\iota'}$ are column vectors. Lower bounds $\min\{\mathbf{X}_{c_l}(:, i_1)\}$ and upper bounds $\max\{\mathbf{X}_{c_l}(:, i_1)\}$ are defined for the variables $i_1 \in \{1, 2, \ldots, l\}$ that are closest to each other[3], and thus determine the boundary of the densely-sampled area. Note that these bounds do not exist for all variables, which is important especially when problems with very long and steep valleys are considered. In such cases it is of advantage to not define bounds on all variables.

When searching for new prospective regions that may contain local or global optima a target value strategy similar to the one proposed by Holmström *et al.* [64] is applied. A vector $\mathbf{T} = s_{\min} - \boldsymbol{\alpha}(f_{\max} - f_{\min})$ represents desired objective function values, where $\boldsymbol{\alpha}$ is a vector of non-negative scalars (see [64]). $s_{\min}$ stands for the minimum of the response surface, and $f_{\min}$ and $f_{\max}$ denote, respectively, the minimal and maximal objective function value obtained so far. An optimization routine is applied in order to minimize the auxiliary function. To achieve a more global search a penalty term is used to prevent the search from entering the densely-sampled area. Similarly, in the local search phase a penalty term is used to prevent the search from leaving the densely-sampled area. In this way the search can be drawn away from or restricted to the already thoroughly examined area.

For each target value the point where the minimum has been located is recorded. These points are clustered into $\tilde{l}$ groups where $\tilde{l}$ can be varied to increase or decrease the number of desired new sample sites. From each of the $\tilde{l}$ groups only the representative that reached the lowest value when optimizing the auxiliary function is chosen as a new sample site. Problems may potentially arise if the global optimum is inside the densely sampled region during the global search or outside the densely sampled region during the local search. This problem is addressed by setting the penalty of a point that reaches during the optimization of the auxiliary function a significantly

[3] $\mathbf{X}_{c_l}(:, i_1)$ denotes the i_1th column of matrix $\mathbf{X}_{c_l}$.

better function value than any other candidate to zero.

Minimizing the auxiliary function requires itself an optimization routine. Two different approaches have been compared, namely a Hooke-Jeeves pattern search method and the accelerated random search algorithm described by Appel *et al.* [7]. Simulations showed that while both optimization routines lead to approximately the same sample sites especially when the number of sample points is rather low, the ARS algorithm finds much better solutions than the pattern search when the number of sample points is larger. An advantage of the pattern search algorithm over ARS is the lower computation time. However, the solution quality is considered more important. Also if a single function evaluation is computationally expensive, the additional time required by ARS is negligible.

For the accelerated random search approach J different starting solutions are uniformly selected from the parameter domain scaled to $\tilde{\Omega} = [0,1]^k$. The optimization is executed similarly to the description by Appel *et al.* [7]. Let $\|\cdot\|$ denote the sup-norm on $\tilde{\Omega}$, and denote by $B(\mathbf{x},\tau) = \{\boldsymbol{\chi} \in \tilde{\Omega} : \|\mathbf{x} - \boldsymbol{\chi}\| \leq \tau\}$ the closed ball centered at $\mathbf{x}$ with radius τ. With given contraction factor $\gamma > 1$ and precision threshold $\rho > 0$, the following steps are executed:

Algorithm 2 ***Accelerated Random Search***

1. *Set iteration counter* $n = 1$*, radius* $\tau_{1,1} = \tau_{1,2} = \ldots = \tau_{1,J} = 1$ *and generate random vectors* $\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,J}$ *from a uniform distribution on* $\tilde{\Omega}$*.*

2. *Given* $\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,J} \in \tilde{\Omega}$ *and* $\tau_{n,1}, \ldots, \tau_{n,J} \in (0,1]$*, generate random* $\boldsymbol{\chi}_{n,1}, \ldots, \boldsymbol{\chi}_{n,J}$ *from uniform distributions on* $B(\mathbf{x}_{n,1}, \tau_{n,1}), \ldots, B(\mathbf{x}_{n,J}, \tau_{n,J})$*.*

3. *For all* $\jmath = 1, \ldots, J$*:*

   If $s(\boldsymbol{\chi}_{n,\jmath}) < s(\mathbf{x}_{n,\jmath})$*, let* $\mathbf{x}_{n+1,\jmath} = \boldsymbol{\chi}_{n,\jmath}$ *and* $\tau_{n+1,\jmath} = 1$*.*
   Otherwise, let $\mathbf{x}_{n+1,\jmath} = \mathbf{x}_{n,\jmath}$ *and* $\tau_{n+1,\jmath} = \tau_{n,\jmath}/\gamma$*.*
   If $\tau_{n+1,\jmath} < \rho$*, then* $\tau_{n+1,\jmath} = 1$*.*

   Increment $n := n + 1$*, and go to Step 2.*

Here, $s(\cdot)$ denotes the prediction of the response surface. At the end of the predefined number of iterations the best of the J results is accepted as the

new solution and the corresponding point is used as the new sample site. The ARS approach as described above can be implemented in parallel so as to execute the calculations for all J starting points simultaneously. Note that the optimization is done on the response surface, and the computationally expensive objective function is not evaluated during ARS.

After the new sample sites have been determined, the computationally expensive objective function is evaluated and the surrogate models are updated. Cross-validation is used to evaluate the goodness of the models, and DST is applied to either find the best single model or the weights of the models in mixtures. The pseudocode of the numerical procedure is given below.

Algorithm 3 ***SO-M: Mixture Surrogate Model Algorithm***

1. *Find an initial set of sample points using a Latin hypercube design, and evaluate the costly objective function at those points.*

2. *Leave-one-out cross-validation: sequentially leave out one sample site and its corresponding function value at a time, and use the remaining data for fitting different response surfaces. Use every response surface to re-predict the function value at the point left out when building the models.*

3. *Calculate goodness values* CC, MAE, RMSE, *and* MAD *between the true function values and the re-predicted values from Step 2 for each model.*

4. (a) *If using the best single model: use the goodness values as evidences and apply Dempster-Shafer theory to choose the best model.*

   (b) *If using a mixture model: use the goodness values to calculate weights associated with each single model. Use the weights for combining models and do leave-one-out cross-validation. For every sample site and each mixture model, re-predict the function value of each point that has been left out when building the mixture model. Calculate* CC, RMSE, MAD, *and* MAE *for each mixture model, and use the scaled values as evidences. Use Dempster-Shafer theory to combine the evidences of the mixture models and choose the best mixture model.*

5. *Find the new sample site(s) in one of the following ways:*

   (a) Local search 1: use the minimum site of the response surface.

   (b) Local search 2: define variable domains where already many samples have been taken (densely-sampled areas, later on referred to as allowed areas) and apply target value strategy that allows searching only within these regions.

   (c) Global search: define variable domains where already many samples have been taken (densely-sampled areas, later on referred to as forbidden areas) and use target value strategy that allows sampling only outside of these regions.

6. *Evaluate the costly objective function at the new sample site(s).*

7. *Update the response surfaces.*

8. *Go to Step 2.*

The algorithm has been implemented in the following three versions:

- Version 1: Initially the best mixture surrogate model is used in every iteration. After a predefined number of failed improvement trials (i.e. new samples that did not improve the current best function value) the algorithm switches to using only the best single model in every iteration.

- Version 2: Only the best mixture surrogate model is used in every iteration.

- Version 3: Only the best single model is used in every iteration.

Initially the algorithm is set to search globally. The criterion described by Holmström *et al.* [64] (later on referred to as strategy A) has been used to determine whether the surface is fluctuating wildly, in which case the minimum site of the response surface is added as the new sample site. Otherwise, forbidden areas are defined. Target values are calculated and the auxiliary function is minimized taking into account possible forbidden areas.

The algorithm stays in the global search phase as long as function value improvements can be found. If no improvements have been obtained, the algorithm switches to the local search phase where the search for the minimum of the auxiliary function is restricted to within the allowed areas. For both search phases it holds that if no restrictions are given for densely-sampled

regions, the minimum of the auxiliary function is sought over the whole variable domain. Thus, local and global search reach in this case the same results.

Numerical experiments showed that it might also be useful to include the minimum of the response surface as new sample site even if the surface is not fluctuating wildly (later on referred to as strategy B). Therefore, also this approach is examined, and thus three algorithm versions are tested with two different criteria for adding the minimum site of the response surface.

All sample sites must have a sufficiently large distance to each other, thus avoiding repeatedly sampling at the same point. In case no such new sample site can be found, an additional point that maximizes the minimal distance to all other already sampled points is used as new sample site.

2.3 Numerical Results

This section summarizes the results of the described algorithms for six commonly used global optimization test problems from the Dixon and Szegö [39] test suit given in Table 2.1. Although all problems have analytical descriptions and the locations of the global optima are known, the problems have been treated as black-boxes in the numerical experiments. The analytical description, as well as global minima and their locations are given in Appendix A. In order to average out the dependency on the initial experimental design and the random component in ARS 20 trials have been made for each algorithm.

Table 2.1: Test problems for numerical experiments [39].
ID - problem identification, k - problem dimension, LM - local minima, GM - global minima; see Appendix A for further information.

ID	Name	k	#Local/Global Minima	Global minimum
B	Branin	2	3 GM, no other LM	0.40
C	Camelback	2	2 GM, no other LM	-1.03
G	Goldstein-Price	2	1 GM, several LM	3.00
H3	Hartmann	3	1GM, 4 LM	- 3.86
H6	Hartmann	6	1 GM, 4 LM	- 3.32
S10	Shekel 10	4	1 GM, 10 LM	-10.54

The results are presented in the following in tables consisting of three sections with one section for every algorithm version. The abbreviations in the first column of each table reflect the rule for redistributing the global conflict, i.e. D stands for Dempster's [37], Y for Yager's [154], I for Inagaki's [70], and PCR for the proportional conflict redistribution rule [132]. The numbers 1, 2, and 3, respectively, indicate the usage of algorithm version 1, 2, and 3. The columns labeled by "min", "max", and "mean" denote the minimal, maximal, and average relative error between the best solution found by the algorithms and the known global minimum for all 20 trials. The figures corresponding to each test problem illustrate the distribution of the relative errors of all 20 trials in the form of box plots. Also every table and figure consists of parts (a) and (b) that reflect the usage of Holmström's criterion (strategy A) for adding the minimum site of the response surface (tables and figures (a)), and the usage of the second criterion (strategy B) that adds the minimum of the response surface more frequently (tables and figures (b)).

All algorithm versions were stopped as soon as 150 function evaluations were reached. The first version consisted of two phases: at first the model combination was applied until either the current best solution has a relative error of less than 10%, no improvement has been found within 30 consecutive function evaluations, or the true function has been evaluated 150 times. If either of the first two conditions is fulfilled, the algorithm switches to using the best single model in every iteration until the maximal number of 150 function evaluations has been reached.

In the experiments the following issues were of interest:

1. Did the algorithms find the vicinities of the global minima? If so, did they find *all* global minima if more than one existed?

2. Were there significant differences in the results when different conflict redistribution rules were used? If so, was there one rule that worked best for all problems?

3. How did the choice of the algorithm version influence the results?

4. How good were the results with respect to relative errors?

5. Did the conflict redistribution rule or the choice of the algorithm version influence the computation times significantly?

6. Which models were chosen, and if mixture models were used, which weights were assigned to the contributing models?

2.3.1 Test Problem B: Branin Function

The results of the algorithms for this problem are summarized in Tables 2.2(a) and 2.2(b). Every initial experimental design contained four sample points. The results show that the mean relative errors are lower for algorithm versions 2 and 3 for both sampling strategies than for algorithm version 1, indicating that a hybrid of mixture and single surrogate models is less effective for this test problem.

Figures 2.1(a) and 2.1(b) illustrate the distribution of the relative errors according to conflict redistribution rule and algorithm version. It can be seen that, with only few exceptions, the medians of the relative errors are lower in Figure 2.1(a) than the corresponding values in Figure 2.1(b). The figures show that the dispersion of the errors is highest whenever algorithm version 1 was used, which is also indicated by the minimal and maximal relative errors given in Tables 2.2(a) and 2.2(b). Algorithm version 2 achieved in general the lowest medians for each decision rule. Thus, it can be concluded that for the Branin test function algorithm version 2 (choosing in every iteration always the best mixture model) is most effective.

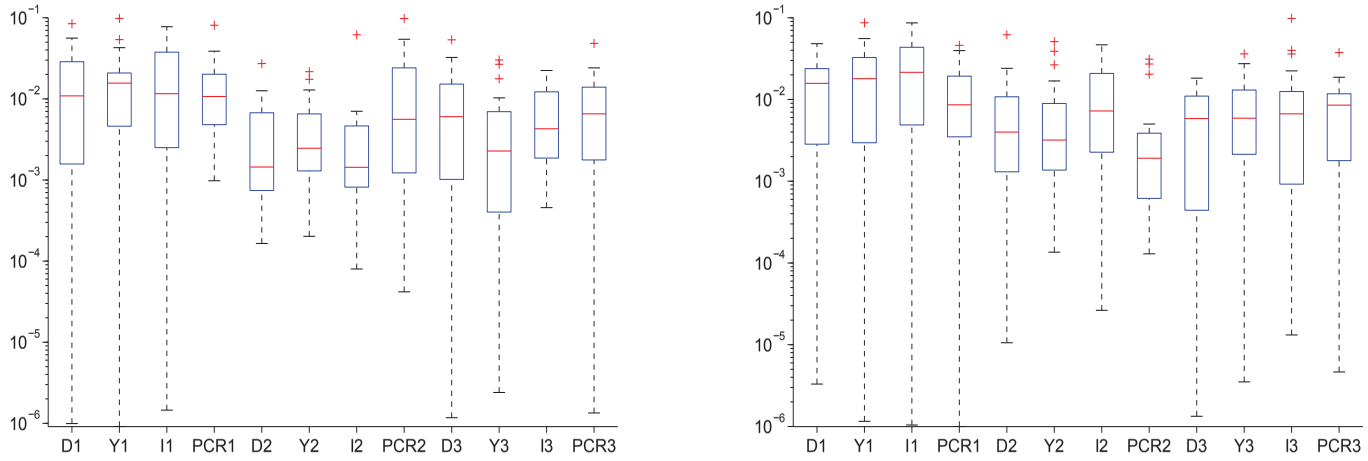
Table 2.2: Branin function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	min	max	mean
D1	$0.0010 \cdot 10^{-3}$	0.0844	0.0186
Y1	$0.0009 \cdot 10^{-3}$	0.0982	0.0200
I1	$0.0015 \cdot 10^{-3}$	0.0775	0.0218
PCR1	$0.9781 \cdot 10^{-3}$	0.0808	0.0158
D2	$0.1649 \cdot 10^{-3}$	0.0272	0.0044
Y2	$0.2017 \cdot 10^{-3}$	0.0216	0.0051
I2	$0.0799 \cdot 10^{-3}$	0.0616	0.0054
PCR2	$0.0418 \cdot 10^{-3}$	0.0978	0.0155
D3	$0.0012 \cdot 10^{-3}$	0.0534	0.0110
Y3	$0.0024 \cdot 10^{-3}$	0.0302	0.0060
I3	$0.4561 \cdot 10^{-3}$	0.0224	0.0073
PCR3	$0.0013 \cdot 10^{-3}$	0.0482	0.0100

(b) Sampling strategy B.

Rule	min	max	mean
D1	$0.0033 \cdot 10^{-3}$	0.0482	0.0171
Y1	$0.0012 \cdot 10^{-3}$	0.0867	0.0220
I1	$0.0010 \cdot 10^{-3}$	0.0865	0.0287
PCR1	$0.0010 \cdot 10^{-3}$	0.0460	0.0130
D2	$0.0106 \cdot 10^{-3}$	0.0618	0.0088
Y2	$0.1355 \cdot 10^{-3}$	0.0511	0.0095
I2	$0.0264 \cdot 10^{-3}$	0.0467	0.0124
PCR2	$0.1294 \cdot 10^{-3}$	0.0311	0.0055
D3	$0.0013 \cdot 10^{-3}$	0.0182	0.0066
Y3	$0.0035 \cdot 10^{-3}$	0.0361	0.0095
I3	$0.0132 \cdot 10^{-3}$	0.0981	0.0135
PCR3	$0.0046 \cdot 10^{-3}$	0.0374	0.0087



(a) Sampling strategy A.

(b) Sampling strategy B.

Figure 2.1: Branin function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

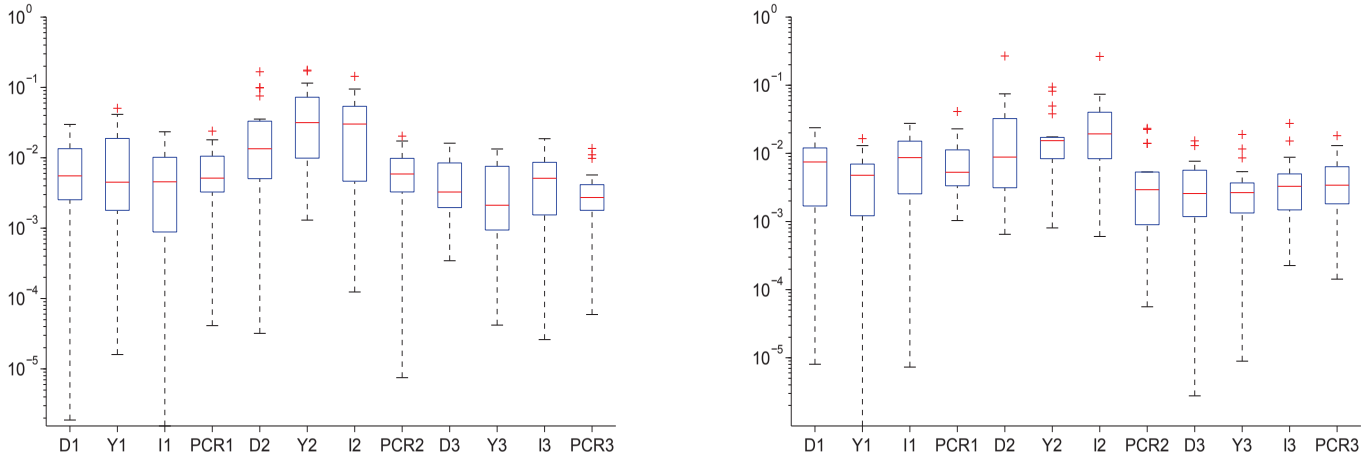
Examining the weights of each model when using the mixture model in algorithm version 1 shows that the kriging model has the highest influence on the response surface. In general all four models contribute at some stage to the combination, but MARS and the polynomial model have usually the lowest weights, indicating that these models were considered worst by DST. After the switch to using in every iteration only the best single model, kriging proved best in almost every iteration. For algorithm version 2 the results were similar. Kriging obtained the highest weights, while RBF, MARS, and the polynomial model had the lowest influence. The polynomial model and RBF had in general about the same weights, while the MARS model had no major impact. When choosing in every iteration only the best single model (algorithm version 3), kriging proved best in over 80% of all choices. MARS and the polynomial model were rarely chosen, and the RBF model was considered best in about 10% of all cases.

2.3.2 Test Problem C: Camelback Function

The results of the various algorithm versions are summarized in Tables 2.3(a) and 2.3(b). Initially four sample sites were used in the experimental designs. The results show that algorithm versions 1 and 3 with Inagaki's or Yager's rule have the lowest mean relative errors for both sampling strategies. The means of the relative errors obtained with algorithm version 2 are lowest when the PCR rule was used for the conflict redistribution, and these results are similar to those of algorithm versions 1 and 3.

The box plots in Figures 2.2(a) and 2.2(b) show that algorithm version 2 has the highest dispersions and medians. In general, algorithm version 3 (choosing in every iteration only the best single model as new response surface) reaches the lowest medians and has the lowest dispersion. Noticeable in Figure 2.2(b) are the higher number and higher values of outliers compared to Figure 2.2(a).

The weights of the models contributing to the mixture model in algorithm version 1 were highest for kriging and RBF, and the influence of the MARS model was lowest. Similar as for the Branin function, the kriging model is used in most iterations after the algorithm switched to using only the best single model. The RBF model was chosen in a few more cases compared to the Branin function. The weights assigned to the single models in algorithm version 2 were highest for kriging. RBF and the polynomial model had about



(a) Sampling strategy A.

(b) Sampling strategy B.

Figure 2.2: Camelback function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

Table 2.3: Camelback function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	min	max	mean
D1	$0.0019 \cdot 10^{-3}$	0.0298	0.0087
Y1	$0.0160 \cdot 10^{-3}$	0.0506	0.0118
I1	$0.0015 \cdot 10^{-3}$	0.0234	0.0059
PCR1	$0.0412 \cdot 10^{-3}$	0.0239	0.0072
D2	$0.0320 \cdot 10^{-3}$	0.1670	0.0317
Y2	$1.3031 \cdot 10^{-3}$	0.1762	0.0510
I2	$0.1240 \cdot 10^{-3}$	0.1437	0.0372
PCR2	$0.0075 \cdot 10^{-3}$	0.0203	0.0069
D3	$0.3443 \cdot 10^{-3}$	0.0161	0.0053
Y3	$0.0419 \cdot 10^{-3}$	0.0133	0.0040
I3	$0.0261 \cdot 10^{-3}$	0.0187	0.0062
PCR3	$0.0592 \cdot 10^{-3}$	0.0136	0.0037

(b) Sampling strategy B.

Rule	min	max	mean
D1	$0.0080 \cdot 10^{-3}$	0.0238	0.0079
Y1	$0.0006 \cdot 10^{-3}$	0.0164	0.0053
I1	$0.0073 \cdot 10^{-3}$	0.0276	0.0097
PCR1	$1.0333 \cdot 10^{-3}$	0.0411	0.0093
D2	$0.6491 \cdot 10^{-3}$	0.2684	0.0298
Y2	$0.8045 \cdot 10^{-3}$	0.0937	0.0222
I2	$0.6032 \cdot 10^{-3}$	0.2641	0.0359
PCR2	$0.0560 \cdot 10^{-3}$	0.0233	0.0055
D3	$0.0028 \cdot 10^{-3}$	0.0152	0.0040
Y3	$0.0089 \cdot 10^{-3}$	0.0189	0.0038
I3	$0.2260 \cdot 10^{-3}$	0.0274	0.0051
PCR3	$0.1425 \cdot 10^{-3}$	0.0182	0.0050

the same influence on the mixture, and the MARS model had again the lowest weights. Also in algorithm version 3 the kriging model was chosen in most cases. The polynomial and MARS model had in comparison to the other models the worst characteristics and were thus rarely chosen.

2.3.3 Test Problem G: Goldstein-Price Function

Since the range of the function values is very large for this test problem, the logarithm of the function values has been used during the calculations. The experimental design contained initially four sample sites.

The results in Tables 2.4(a) and 2.4(b) show that all algorithm versions performed about equally well. For algorithm version 1 Inagaki's rule for conflict redistribution worked best for sampling strategy A, but worst for sampling strategy B. Similarly, the best results for algorithm version 2 were achieved with Yager's rule when sampling strategy A was used, but this rule gave the second worst result when using sampling strategy B. Only for algorithm version 3 are the best results for both sampling strategies achieved with the same conflict redistribution rule (Inagaki's rule).

The box plots in Figures 2.3(a) and 2.3(b) are rather similar for every decision rule, but Figure 2.3(b) contains more outliers. The dispersion of the errors in both figures is about equal. The results suggest that algorithm version 3 with Inagaki's conflict redistribution rule should be favored.

Table 2.4: Goldstein-Price function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

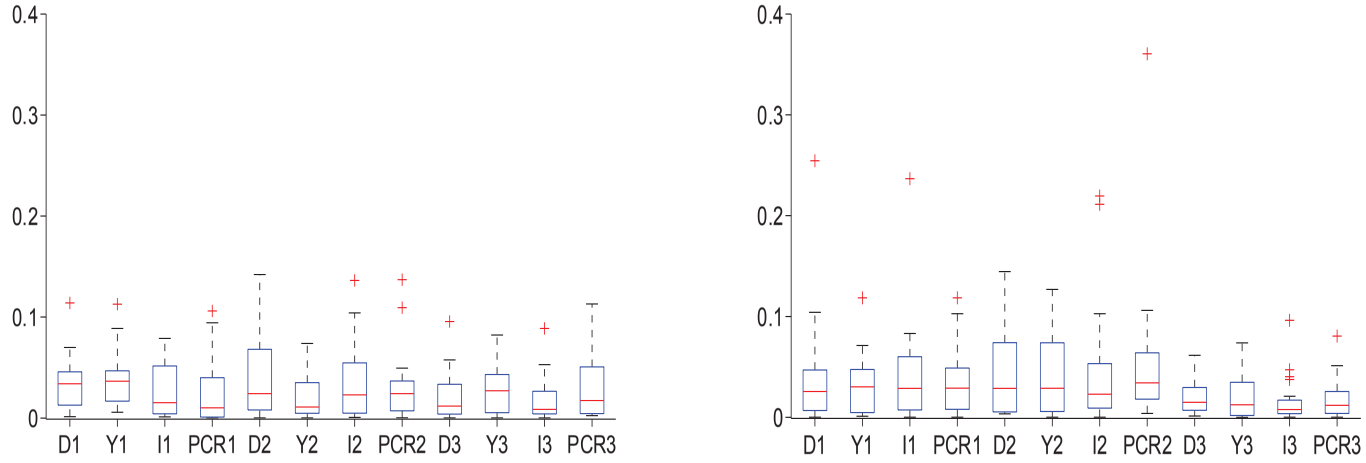
(a) Sampling strategy A.

Rule	min	max	mean
D1	0.0014	0.1140	0.0345
Y1	0.0058	0.1128	0.0394
I1	0.0012	0.0789	0.0263
PCR1	0.0002	0.1061	0.0277
D2	0.0002	0.1421	0.0415
Y2	0.0002	0.0739	0.0212
I2	0.0006	0.1363	0.0372
PCR2	0.0003	0.1370	0.0301
D3	0.0002	0.0956	0.0214
Y3	0.0002	0.0822	0.0303
I3	0.0002	0.0887	0.0182
PCR3	0.0024	0.1130	0.0312

(b) Sampling strategy B.

Rule	min	max	mean
D1	0.0002	0.2545	0.0393
Y1	0.0011	0.1185	0.0327
I1	0.0001	0.2367	0.0413
PCR1	0.0002	0.1185	0.0349
D2	0.0013	0.0615	0.0219
Y2	0.0002	0.1268	0.0416
I2	0.0002	0.2196	0.0492
PCR2	0.0039	0.3605	0.0555
D3	0.0013	0.0615	0.0219
Y3	0.0001	0.0738	0.0220
I3	0.0002	0.0962	0.0161
PCR3	0.0002	0.0806	0.0181

The weights assigned to the models contributing to the mixture differ from those obtained for the Branin and the Camelback function. The highest weights in the mixture model of algorithm version 1 were assigned to the RBF model. The kriging model had the second highest influence while the MARS and polynomial model were rather insignificant in most cases. After the switch to using only the best single model, RBF was chosen in over 90% of all cases. Also in algorithm version 2 the RBF model had significantly higher weights than the other models, and the weights of the kriging and the polynomial model were in many cases about equal. Again, the MARS model did not contribute much to the response surface, indicating that is was considered worst by DST. In algorithm version 3 the RBF model had the best model characteristics and was thus most often chosen as best



(a) Sampling strategy A.

(b) Sampling strategy B.

Figure 2.3: Goldstein-Price function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

single model. The polynomial and MARS model had the worst model characteristics.

2.3.4 Test Problem H3: Three-Dimensional Hartmann Function

The results of the three versions of the algorithm are shown in Tables 2.5(a) and 2.5(b). Five points were used in the initial experimental design. The data shows algorithm version 1 performed in general worse than versions 2 and 3. Sampling strategy A seems to have in general slightly larger relative errors than strategy B. The differences between the distributions of the relative errors illustrated in Figures 2.4(a) and 2.4(b) are rather negligible with respect to dispersion and median values.

Table 2.5: Three-dimensional Hartmann function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

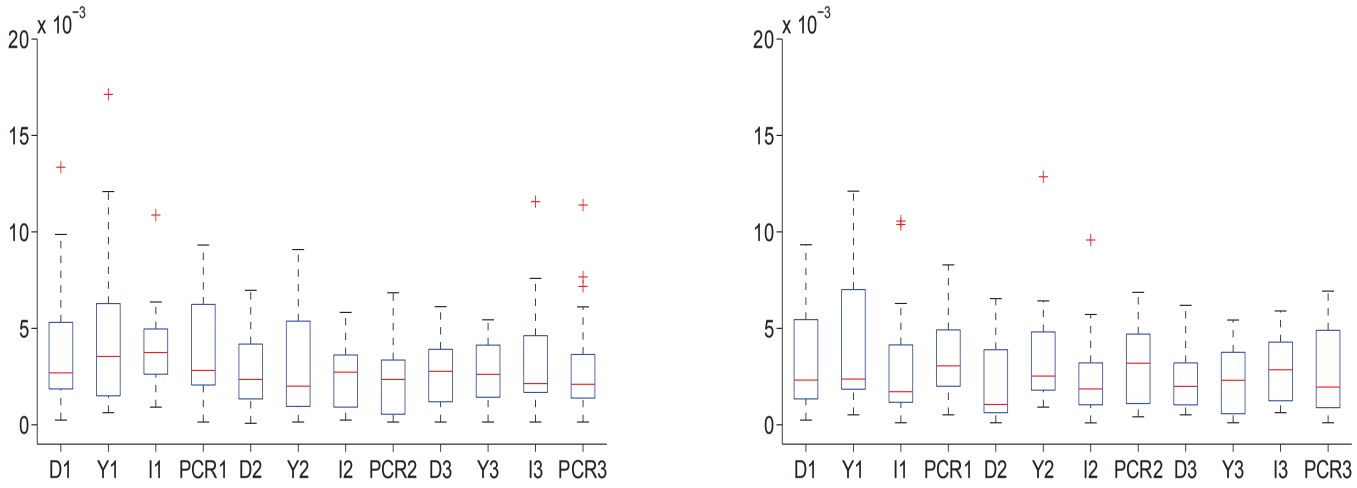
(a) Sampling strategy A.

Rule	min	max	mean
D1	0.0002	0.0134	0.0040
Y1	0.0006	0.0171	0.0047
I1	0.0009	0.0109	0.0040
PCR1	0.0001	0.0093	0.0039
D2	0.0001	0.0070	0.0028
Y2	0.0001	0.0091	0.0032
I2	0.0002	0.0058	0.0026
PCR2	0.0001	0.0068	0.0025
D3	0.0001	0.0061	0.0028
Y3	0.0001	0.0054	0.0028
I3	0.0001	0.0116	0.0033
PCR3	0.0001	0.0113	0.0032

(b) Sampling strategy B.

Rule	min	max	mean
D1	0.0002	0.0093	0.0034
Y1	0.0005	0.0121	0.0043
I1	0.0001	0.0106	0.0030
PCR1	0.0005	0.0083	0.0034
D2	0.0001	0.0065	0.0021
Y2	0.0009	0.0129	0.0035
I2	0.0001	0.0096	0.0025
PCR2	0.0004	0.0069	0.0031
D3	0.0005	0.0062	0.0024
Y3	0.0001	0.0054	0.0023
I3	0.0006	0.0059	0.0028
PCR3	0.0001	0.0069	0.0027

For the three-dimensional Hartmann function the kriging model obtained the highest weights when using mixture models in algorithm versions 1 and 2. For



(a) Sampling strategy A.

(b) Sampling strategy B.

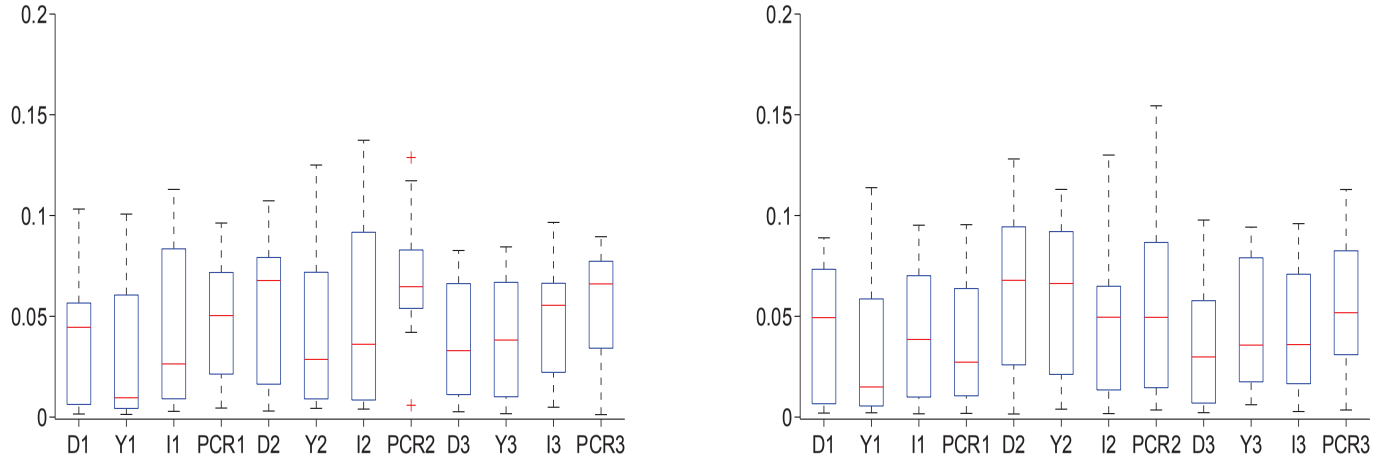
Figure 2.4: Three-dimensional Hartmann function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

algorithm version 1 the weights for the MARS model were in general higher as compared to any other already considered test function. The weights of the polynomial model were lowest. On the other hand, in algorithm version 2 the weight of the MARS model was again comparatively low, and the influence of the kriging model had increased compared to version 1. When only the best single model was used, the kriging model was chosen in over 90% of all cases.

2.3.5 Test Problem H6: Six-Dimensional Hartmann Function

The results for the three algorithm versions are shown in Tables 2.6(a) and 2.6(b), respectively. Initially eight points were used in the experimental design. The results show that algorithm version 1 achieves the lowest mean relative errors for most conflict redistribution rules. Noticeable is that the solution quality has in general decreased compared to the previously considered test problems, indicating that the performance of the algorithms is influenced by the increasing number of variables. Box plots of the relative errors are illustrated in Figures 2.5(a) and 2.5(b). The medians and the dispersion of the relative errors are in general larger than for any other previously considered test problem. For both sampling strategies algorithm version 1 with Yager's conflict redistribution rule performs best.

The weights determined for the models contributing to the mixture models in algorithm version 1 show that the kriging model had the highest influence (50-60%). For this test problem also the MARS model had influence on the response surface in several cases and did not obtain the lowest weights. In the second phase of algorithm version 1 the kriging model was chosen by every decision rule in every iteration. In algorithm version 2 again the kriging model had the highest influence, while MARS and the polynomial model were assigned the lowest weights. In algorithm version 3 the kriging model was chosen in almost 90% of all iterations, and the polynomial and MARS model were rarely used.



(a) Sampling strategy A.

(b) Sampling strategy B.

Figure 2.5: Six-dimensional Hartmann function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

Table 2.6: Six-dimensional Hartmann function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	min	max	mean
D1	0.0015	0.1032	0.0388
Y1	0.0013	0.1007	0.0311
I1	0.0029	0.1130	0.0415
PCR1	0.0045	0.0963	0.0477
D2	0.0030	0.1073	0.0538
Y2	0.0043	0.1251	0.0470
I2	0.0040	0.1374	0.0501
PCR2	0.0058	0.1289	0.0689
D3	0.0027	0.0827	0.0383
Y3	0.0017	0.0844	0.0403
I3	0.0049	0.0966	0.0477
PCR3	0.0012	0.0895	0.0563

(b) Sampling strategy B.

Rule	min	max	mean
D1	0.0020	0.0889	0.0419
Y1	0.0021	0.1138	0.0325
I1	0.0016	0.0952	0.0408
PCR1	0.0018	0.0955	0.0380
D2	0.0015	0.1281	0.0639
Y2	0.0039	0.1130	0.0610
I2	0.0017	0.1300	0.0438
PCR2	0.0035	0.1544	0.0533
D3	0.0022	0.0978	0.0375
Y3	0.0061	0.0942	0.0465
I3	0.0028	0.0960	0.0432
PCR3	0.0035	0.1129	0.0538

2.3.6 Test Problem S10: Shekel Function

In the considered four-dimensional case the optimal function value is $f^* = -10.54$, and the number of local optima is 10. The initial experimental design contained 16 points. All versions of the algorithm had trouble finding the global optimum (see Tables 2.7(a) and 2.7(b)). The minimal relative errors were achieved by algorithm version 1 for both sampling strategies. Figures 2.6(a) and 2.6(b) show the distribution of the relative errors for each algorithm version and conflict redistribution rule. Compared to all other previously considered test problems, the results are much worse, and the dispersion is very large. The box plots show that sampling strategy B has more outliers than strategy A, but otherwise there are no significant differences.

Table 2.7: Shekel function. Relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

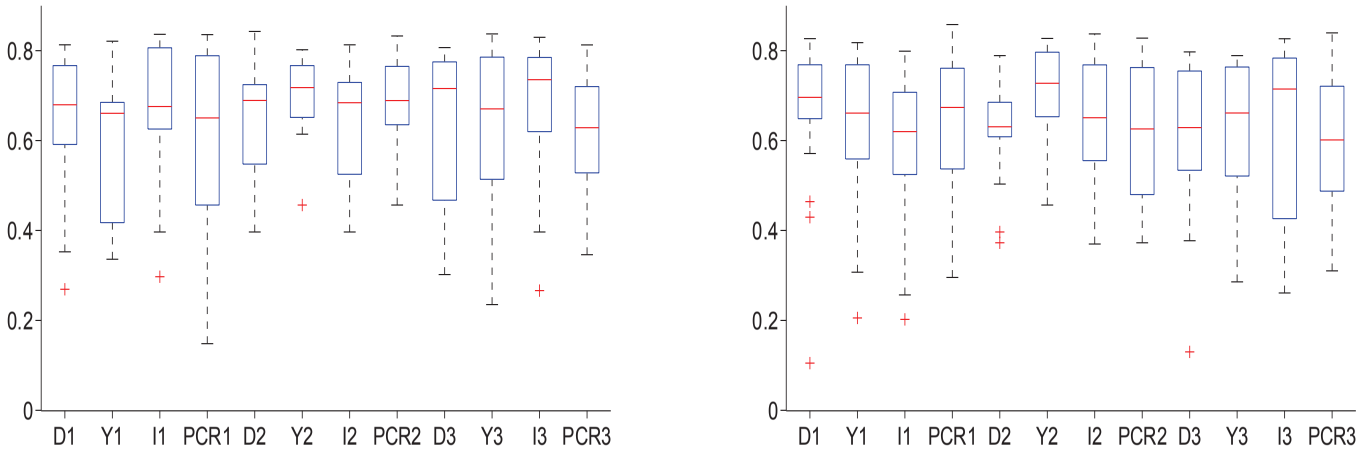
(a) Sampling strategy A.

Rule	min	max	mean
D1	0.2693	0.8130	0.6526
Y1	0.3360	0.8209	0.5927
I1	0.2973	0.8362	0.6705
PCR1	0.1483	0.8357	0.6049
D2	0.3966	0.8427	0.6412
Y2	0.4565	0.8020	0.7035
I2	0.3966	0.8129	0.6418
PCR2	0.4565	0.8326	0.6847
D3	0.3020	0.8069	0.6360
Y3	0.2352	0.8372	0.6296
I3	0.2663	0.8296	0.6736
PCR3	0.3462	0.8126	0.6216

(b) Sampling strategy B.

Rule	min	max	mean
D1	0.1051	0.8267	0.6657
Y1	0.2055	0.8179	0.6305
I1	0.2023	0.7987	0.5808
PCR1	0.2955	0.8581	0.6390
D2	0.3726	0.7888	0.6287
Y2	0.4565	0.8270	0.7106
I2	0.3699	0.8373	0.6464
PCR2	0.3726	0.8278	0.6225
D3	0.1300	0.7971	0.6124
Y3	0.2857	0.7888	0.6371
I3	0.2611	0.8265	0.6189
PCR3	0.3102	0.8396	0.6014

Also for this test function the weights assigned to the kriging model were highest in algorithm version 1. The MARS model obtained the lowest weights. However, the differences between the weights assigned to kriging, RBF, and the polynomial model were much smaller than for the previously considered test problems. After the switch to using only the best single model, kriging was chosen in most cases, but also RBF had in several cases the best model characteristics. The polynomial and MARS model never obtained a better evaluation than kriging or RBF. The same statements hold for algorithm versions 2 and 3. The weights of the models in version 2 were on average as for the first stage of algorithm version 1. In version 3 kriging and RBF were chosen as best models most often.



(a) Sampling strategy A.

(b) Sampling strategy B.

Figure 2.6: Shekel function. Distribution of relative errors for algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

2.3.7 General Results

As can be expected, the computation time is for all test problems highest whenever only the mixture model was used and lowest when using only the best single model. The mixture models require the predictions of more than one model, and finding the best mixture model requires several more calculations. Thus, the computation time is higher than when only the best single model has to be determined, and predictions of only one model are used. With respect to the conflict redistribution rules, there was in general no difference in computation times.

As mentioned in Section 2.2, the set representing complete ignorance, i.e. the combination of all four models, had to be considered separately since it always has belief and plausibility values equal to one. Thus, algorithm versions 1 and 2 were adjusted such that the four-model mixture was used. Comparing the obtained results for all test problems to the reported results in the foregoing sections showed that there were only a few instances in which the four-model mixture achieved better results. In general the four-model mixture resulted in higher maximal relative errors and also the dispersion of the relative errors was larger. The influence of the single models in the combination was highest for kriging and/or RBF, respectively. Compared to the other algorithm versions the weights for MARS and the polynomial model were in general higher, and both models had always about the same influence. It can be concluded that using in every iteration the four-model mixture does not lead to improved results.

In addition to investigating how close the algorithms get to the global optima with respect to relative errors, it is of interest if the global optima could be detected and how many global optima could be detected if there were more than one. In applications it is often of interest how many global optima there are, and where in the variable domain they are located, because some variable adjustments might be in practice easier to realize than others. Tables 2.8(a) and 2.8(b) show how often each algorithm was able to sample in the vicinity of the global minima. If a function has, for example, three global minima, and 20 trials were made with an algorithm, then 100% indicates that samples in the vicinity of all three global optima have been taken in every trial. The vicinity was here defined in terms of the distance of the sample sites to the known locations of global and local optima. The results show that for the Goldstein-Price and the three-dimensional Hartmann function, all algorithm versions found the vicinity of the global minima without getting trapped in local optima. Algorithm version 3 found

the highest number of global minima for all decision rules when the problem dimension is at most three. However, for higher dimensional problems algorithm versions 1 and 2 are more successful.

In general the proportional conflict redistribution rule proved the most successful with respect to the number of global minima found, and has thus a good influence on the global search. Compared to the quality of the results however, the PCR rule was not always the best and did in general not lead to the lowest errors, indicating a negative influence on the local search. For the Shekel function the lowest relative error obtained was 10.51% by Dempster's conflict redistribution rule using algorithm version 1 and sampling strategy B. However, all algorithms found the vicinity of the global minimum in 50-95% of all test cases, but still the relative errors are not very satisfying. Thus, it can be concluded that the algorithms failed in searching thoroughly locally, and thus they missed the very steep minimum of this function. Therefore, the local search strategy of the algorithms must be enhanced so that also the relative errors for such functions can be decreased.

Comparing the results in Tables 2.8(a) and 2.8(b) shows that the number of global minima found was in general higher when using Holmström's criterion (sampling strategy A) for using the minimum site of the response surface as sample point. Based on these and the foregoing results (tables and figures corresponding to sampling strategy A) it seems reasonable to employ Holmström's criterion. Moreover, since algorithm versions 1 and 2 proved more successful with increasing dimensions, they should also be used for problems containing more than six variables.

Table 2.8: Percentage of trials in which basins of global optima were found; B - Branin, C - Camelback, G - Goldstein-Price, H3 - three-dim. Hartmann, S10 - Shekel, H6 - six-dim. Hartmann; algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	B	C	G	H3	S10	H6
D1	91.67	95.00	100.00	100.00	65.00	85.00
Y1	76.67	100.00	100.00	100.00	75.00	95.00
I1	86.67	100.00	100.00	100.00	50.00	80.00
PCR1	83.33	100.00	100.00	100.00	70.00	80.00
D2	86.67	65.00	100.00	100.00	85.00	80.00
Y2	83.33	57.50	100.00	100.00	60.00	90.00
I2	81.67	72.50	100.00	100.00	75.00	90.00
PCR2	81.67	100.00	100.00	100.00	75.00	85.00
D3	96.67	100.00	100.00	100.00	60.00	80.00
Y3	96.67	100.00	100.00	100.00	60.00	80.00
I3	95.00	100.00	100.00	100.00	50.00	85.00
PCR3	100.00	100.00	100.00	100.00	65.00	70.00

(b) Sampling strategy B.

Rule	B	C	G	H3	S10	H6
D1	85.00	100.00	100.00	100.00	55.00	75.00
Y1	81.67	97.50	100.00	100.00	65.00	85.00
I1	85.00	95.00	100.00	100.00	75.00	85.00
PCR1	83.33	100.00	100.00	100.00	65.00	80.00
D2	81.67	77.50	100.00	100.00	80.00	90.00
Y2	81.67	80.00	100.00	100.00	45.00	85.00
I2	25.00	82.50	100.00	100.00	70.00	85.00
PCR2	81.67	100.00	100.00	100.00	60.00	85.00
D3	96.67	100.00	100.00	100.00	60.00	80.00
Y3	96.67	100.00	100.00	100.00	50.00	85.00
I3	95.00	100.00	100.00	100.00	45.00	80.00
PCR3	100.00	100.00	100.00	100.00	65.00	65.00

Tables 2.9(a) and 2.9(b) show the minimum number of objective function evaluations that were necessary to reach relative errors of less than 1% over all 20 trial (note that the Shekel test function is not included since relative errors of less than 1% were not achieved). The numbers show that Inagaki's rule should be used with algorithm version 1. For algorithm version 2 Yager's rule proved the most efficient, and Dempster's rule worked best when algorithm version 3 was used.

Table 2.9: Minimum number of function evaluations to reach less than 1% relative error; B - Branin, C - Camelback, G - Goldstein-Price, H3 - three-dim. Hartmann, H6 - six-dim. Hartmann; algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	B	C	G	H3	H6
D1	26	15	47	30	72
Y1	23	22	52	19	56
I1	12	22	39	20	40
PCR1	27	10	33	16	67
D2	23	16	46	21	70
Y2	12	15	42	14	60
I2	25	14	50	18	63
PCR2	27	9	65	21	58
D3	12	17	29	17	62
Y3	27	25	46	22	96
I3	21	9	29	21	129
PCR3	14	21	42	20	102

(b) Sampling strategy B.

Rule	B	C	G	H3	H6
D1	33	24	22	22	79
Y1	17	10	54	18	101
I1	14	22	19	13	37
PCR1	25	24	29	17	66
D2	17	24	49	24	79
Y2	12	15	40	19	51
I2	27	8	69	20	71
PCR2	17	11	48	20	88
D3	12	15	30	19	51
Y3	25	16	29	28	62
I3	13	22	36	20	89
PCR3	15	23	22	22	64

Tables 2.10(a) and 2.10(b) show the number of trials (out of 20) for which each method could approximate the global optimum with less than 1% relative error. For example, with sampling strategy A the method D1 found for test problem B in 10 out of 20 trials the global optimum with an error of less than 1%. For algorithm version 1 Inagaki's rule is most successful, while Dempster's or Yager's rule are recommendable when using algorithm version 2. For algorithm version 3 Dempster's rule works best. An overall comparison of the results of all conflict redistribution rules shows that Yager's and the PCR rule have the best performance. Together with the results from Tables 2.9 it seems promising to use Inagaki's rule when algorithm

version 1 is used, Yager's rule in algorithm version 2, and Dempster's rule with algorithm version 3 in order to obtain the best results.

Table 2.10: Number of trials (out of 20) in which solutions were found that have less than 1% relative error; B - Branin, C - Camelback, G - Goldstein-Price, H3 - three-dim. Hartmann, H6 - six-dim. Hartmann; algorithm versions 1-3, sampling strategies A and B; conflict redistribution rules: D - Dempster, Y - Yager, I - Inagaki, PCR - proportional conflict redistribution.

(a) Sampling strategy A.

Rule	B	C	G	H3	H6
D1	10	14	4	19	7
Y1	17	8	5	20	3
I1	13	15	10	20	5
PCR1	6	13	3	18	10
D2	16	5	10	20	6
Y2	16	18	6	20	5
I2	9	15	9	19	6
PCR2	19	7	8	20	5
D3	13	16	11	19	3
Y3	9	13	10	20	2
I3	13	15	7	20	1
PCR3	13	18	7	19	2

(b) Sampling strategy B.

Rule	B	C	G	H3	H6
D1	7	11	8	20	7
Y1	14	10	7	20	2
I1	15	18	6	20	6
PCR1	9	16	6	18	9
D2	16	7	7	19	2
Y2	11	16	7	20	2
I2	7	12	4	18	5
PCR2	12	7	6	20	4
D3	14	18	13	20	2
Y3	11	13	6	20	5
I3	17	16	4	20	2
PCR3	13	16	10	20	4

2.4 Conclusions

In this chapter the application of Dempster-Shafer theory to surrogate model choice and surrogate model combination in global optimization problems has been presented. Model characteristics obtained from cross-validation reflect good and bad properties of the considered models and their combinations, respectively. Different rules, namely Dempster's, Yager's, Inagaki's, and the proportional conflict redistribution rule, for redistributing the conflict arising when one model has good and bad characteristics simultaneously have been compared. Two sampling procedures that use different criteria for adding the minimum site of the response surface as sample point have been examined. The influence of mixture models, single models, and a hybrid of both, and the importance of the sampling strategy have been assessed

in numerical experiments on a subset of the Dixon and Szegö [39] global optimization test problem suit.

The advantage of the presented Dempster-Shafer theory method is that in contrast to the approach by Goel *et al.* [53] no parameters need to be adjusted for emphasizing or restricting the influence of good and bad models in the mixture. In comparison to the method of Viana and Haftka [144] no auxiliary optimization problem has to be solved for computing the model weights, and the Dempster-Shafer theory approach guarantees that all weights are non-negative and at most one.

The numerical results showed that the proposed approach is successful in finding the global minima of most test problems. The exploration of the variable domain had a good ratio of local and global search, and thus the vicinities of global minima were detected, and the actual global minimum could, with the exception of the Shekel test function, be approximated with average relative errors of less than 1% within 150 function evaluations. All algorithms had difficulties finding a good approximation of the global minimum only for the Shekel test function, which is characterized by a very steep global minimum and several local minima. Although the vicinity of the global minimum has been detected and several samples have been taken in that region, the algorithms failed to search thoroughly enough to find the actual location of the minimum. This implies that the used local search procedure could still be improved in order to detect also very steep minima. In connection with this issue is also the adjustment of parameters determining for example when to invoke the local search, how many points should be sampled, or the definition of the search area when optimizing the auxiliary function. A more random sampling strategy may improve the local search.

The results also indicated that the success of the algorithms in finding all global minima was to some extent dependent on the used conflict redistribution rule and the algorithm version. In general, the proportional conflict redistribution rule led to the best variable domain exploration (global search). On the other hand, this rule did not always lead to the lowest relative errors, and thus the possibility of linking the conflict redistribution rule to the search phase (local or global) could be examined in further experiments. With respect to variable domain exploration and relative errors the results showed that with increasing problem dimension the algorithms applying model combinations become more favorable.

The number and type of model characteristics used for the evaluation of the goodness of the single surrogate models should in general be derived from the specific application problem, i.e. in some cases certain model characteristics may be more important than others and should thus be emphasized in the model evaluation. For real-world application problems an extension of the algorithms to handling linear, nonlinear, and integrality constraints, which are often present in engineering problems, is necessary.

With respect to computation times the numerical experiments showed that there are in general no major differences when using different conflict redistribution rules. Differences arise obviously when using single or mixture models. For mixture models parameters of more models have to be computed, which is computationally more expensive. In general it can however be assumed that evaluating the true function is computationally much more expensive than building the surrogate models, and thus the algorithm's own computation time becomes negligible. Moreover, as the algorithm advances, it may be advantageous to use the leave-one-out cross-validation only on a subset of the data in the vicinity of the best point found so far to determine the locally best surrogate model. Also, making the weights w_r in equation (2.2) dependent on the variable $\mathbf{x}$ may emphasize or restrict the influence of single models depending on the point in the variable domain.

Lastly, it should be mentioned that the described algorithms are suitable for distributed computing. For example, the cross-validation of the single models is an embarrassingly parallel computation, as is the procedure for computing the surrogate model parameters, and the method for minimizing the auxiliary function. The cross-validation becomes time consuming as the number of models in the mixture and the number of sample sites increases. Therefore, switching to a so-called $\tilde{k}$-fold cross-validation strategy should be considered. It may also be of advantage to re-evaluate and choose the best (mixture) model only every, say, tth iteration to save computation time. An important issue for this approach is however the determination of t, and thus how often the (mixture) model should be re-adjusted while keeping the solution quality at a high level. These extensions exceed however the topic of this chapter.

Chapter 3

Influence of Surrogate Model Choice and Sampling Strategies

Abstract

This chapter describes a computational study of extensions of SO-M to solving larger problems with up to 30 dimensions. The leave-one-out cross-validation is replaced by a $\tilde{k}$-fold strategy, which leads to significant computation time savings as the number of sample sites and dimensions increases. Two sampling procedures have been examined, namely a random sampling strategy (algorithm SO-M-c), and a strategy where the minimum point of the response surface is used as the new sample point in every iteration. The surrogate models contributing to the mixture may not change and only their weights are adjusted between iterations in order to better examine which (mixture) surrogate models work generally well or should be avoided. The performance of SO-M-c and SO-M-s with various (mixture) surrogate models are compared in numerical experiments to examine the influence of the sampling strategy on the results. Also the efficient global optimization algorithm (EGO) [74], and Gutmann's radial basis function algorithm [57] are included in the comparison. The algorithms have been compared on 13 literature test problems, and two application problems. One application deals with groundwater bioremediation, and the other one arises in energy generation using tethered kites. The results of the numerical experiments show that when the problem dimension is low (at most six), then SO-M-s performs on average better than SO-M-c. For problems with 12 and more dimensions, SO-M-c performs in general better than SO-M-s. EGO performed better for low-dimensional problems than for large-dimensional ones. Gutmann's

method is except for one convex problem outperformed by all other algorithms. Moreover, the numerical experiments showed that mixtures with the cubic RBF model work in general well, whereas using only a polynomial regression model should be avoided. Mixture surrogate model algorithms should be used whenever it is a priori unclear which surrogate model performs best because using a mixture prevents selecting the worst model, and often leads to better results than when only a single model is used.

Abbreviations and Nomenclature

DACE	Design and analysis of computer experiments
DST	Dempster-Shafer Theory
EGO	Efficient Global Optimization, [74]
G	Gutmann's radial basis function algorithm [57]
GM	Global minima
K	Kriging model
LM	Local minima
M, MARS	Multivariate adaptive regression spline
P	Polynomial regression model
R, RBF	Radial basis function surrogate model
SEM	Standard error of means
SO-M	Surrogate Optimization - Mixture
SO-M-c	Surrogate Optimization - Mixture - candidate sampling
SO-M-s	Surrogate Optimization - Mixture - surface minimum sampling

$\mathbf{x}$	Continuous variable vector
$\mathbf{x}^T$	Transpose of $\mathbf{x}$
x_i	ith continuous variable, see equation (2.1c)
k	Problem dimension
n_0	Number of points in the initial experimental design
n	Number of function evaluations obtained so far
$\tilde{k}$	Number of points in the validation set for cross-validation
$f_{\min}$	Minimal objective function value found so far
$s_k(\cdot)$	Kriging surrogate model
Ω	Box-constrained variable domain
$s_b(\cdot)$	Radial basis function interpolant
w_r	Weight of the rth model in the mixture
$\mathcal{S}$	Set of already evaluated sample points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
$\boldsymbol{\chi}_\jmath$	$\jmath$th candidate point, $\jmath = 1, \ldots, t$
$\mathbf{x}_{\text{best}}$	Best point found so far
$\mathbb{P}$	Perturbation probability of each variable, see equation (3.4)
V_D	Scaled distance criterion, see equation (3.6)
V_R	Scaled response surface criterion, see equation (3.7)
V	Weighted score, see equation (3.8)
ω_R	Weight for response surface criterion, see equation (3.8)
ω_D	Weight for distance criterion, see equation (3.8)

3.1 Introduction and Motivation

In this chapter the same type of box-constrained continuous global optimization problems as in Chapter 2 (equations (2.1a)-(2.1c)) are considered. The goal of this chapter is (a) to further develop the mixture surrogate model algorithm SO-M introduced in Chapter 2 by applying a random sampling method when selecting the next sample site, (b) to examine the influence of the (mixture) surrogate model and the strategy for iteratively selecting sample points for doing the expensive simulation on the results, and (c) to improve the mixture surrogate model algorithm introduced in Chapter 2 such that larger dimensional problems can be solved efficiently. Within this scope the efficient global optimization algorithm (EGO) [74] and the radial basis function algorithm by Gutmann [57] are compared to the improved mixture model algorithm on 13 global optimization test problems from the literature, a 12-dimensional groundwater bioremediation application problem, and a 13-dimensional application problem arising in energy generation using kites.

The remainder of this chapter is structured as follows. Section 3.2 briefly reviews the surrogate model algorithms used in the numerical experiments [57, 74, 98]. In Section 3.3 the improved mixture surrogate model algorithm SO-M-c (Surrogate Optimization - Mixture - candidate sampling) is described. The setup for the numerical experiments is described in Section 3.4, and the results are discussed in Section 3.5. Section 3.6 concludes this chapter.

Within the scope of this study, a Matlab toolbox has been developed that allows the user to choose between different strategies for building the initial experimental design, (mixture) surrogate models, and sampling strategies. The toolbox can be obtained on request from the author or on Matlab File Exchange ("Surrogate Model Optimization Toolbox", File ID #38530).

3.2 Review of Surrogate Model Algorithms

The common basic structure of surrogate model algorithms has already been described in Section 1.2.1. Recall that the algorithms are iterative. In the first step an initial experimental design is generated, and the costly objective function is evaluated at these points. Secondly, the chosen response surface is fit to the given data, and in the third step the next sample site is chosen according to some strategy that is based on objective function

value predictions by the response surface. The costly objective function is evaluated at the chosen point, and in step four, the response surface parameters are updated with the new data. The algorithm iterates through steps three and four until a predefined stopping criterion has been reached. Surrogate model algorithms differ in general with respect to the method of generating the initial experimental design, the surrogate model, and the strategy for choosing the next sample site.

Two well-known and widely used surrogate model algorithms, namely EGO [74] and Gutmann's radial basis function algorithm [57], are briefly reviewed in the following sections. Also, a recap of the mixture surrogate model algorithm SO-M [98] that uses Dempster-Shafer theory for computing the weights of the models in the mixture is provided.

3.2.1 Efficient Global Optimization (EGO)

The initial experimental design in the EGO algorithm [74] is created by generating a Latin hypercube design in k dimensions, so that all one- and two-dimensional projections are nearly uniformly covered. Initially, with slight deviations, $n_0 = 10k$ points are generated.

A kriging model (see Section 1.2.5) is used as response surface [32, 92]. The advantage of the kriging model is that an uncertainty estimate is computed when making predictions. These uncertainty estimates are then used by the EGO algorithm for determining the next sample site. A disadvantage of the kriging model is the computation of the model parameters. The number of parameters depends on the problem dimension, and kriging is known to be plagued by the curse of dimensionality, not just in computer memory and time but also in ill-conditioning.

EGO determines the next sample point by maximizing the expected improvement

$$\mathbb{E}\left[I(\mathbf{x})\right] = \mathbb{E}\left[\max(f_{\min} - Y, 0)\right], \tag{3.1}$$

where $f_{\min}$ is the best function value found so far, and Y is a random variable. The closed form solution can be shown to be

$$\mathbb{E}\left[I(\mathbf{x})\right] = \left(f_{\min} - s_k(\mathbf{x})\right) \Phi\left(\frac{f_{\min} - s_k(\mathbf{x})}{\zeta}\right) + \zeta\phi\left(\frac{f_{\min} - s_k(\mathbf{x})}{\zeta}\right), \tag{3.2}$$

for $\zeta > 0$, where $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard normal density and distribution function, respectively, and ζ is the root mean squared error of

the prediction at the point $\mathbf{x} \in \Omega$ obtained from the kriging model. The expected improvement function is however not unimodal, and to find the global maximum, a global optimization routine must be used. The advantage is that the computationally expensive objective function is not involved when maximizing the expected improvement. Finding the global optimum of this auxiliary optimization problem remains, however, a challenging task.

Jones *et al.* [74] solved the subproblem of maximizing the expected improvement with a branch and bound algorithm. The chosen stopping criterion for the EGO algorithm was a maximal expected improvement of less than 1%. EGO has been tested in [74] on problems of at most six dimensions, whereas an 11-dimensional problem has been solved with EGO in [64].

3.2.2 Gutmann's Radial Basis Function Algorithm

Gutmann's algorithm [57] uses an RBF interpolant (see equation (1.9)). The next sample site is determined by minimizing a "bumpiness" measure

$$g_n(\mathbf{x}) = (-1)^{m_0+1} \mu_n(\mathbf{x}) \left[f_n^* - s_b(\mathbf{x})\right]^2, \tag{3.3}$$

where m_0 depends on the chosen radial basis function and is either -1, 0, or 1, f_n^* is a target value for the objective function in iteration n, $s_b(\cdot)$ is the RBF interpolant as defined in equation (1.9), and $\mu_n(\mathbf{x})$ is the coefficient corresponding to $\mathbf{x}$ of the Lagrangian function L that satisfies $L(\mathbf{x}_\iota) = 0$, $\iota = 1, \ldots, n$, and $L(\mathbf{x}) = 1$ in iteration n. Gutmann's algorithm has been applied to test problems of at most ten dimensions in [57], and the author of this paper remarked that optimizing the auxiliary function becomes computationally more complex as the number of iterations increases. Another disadvantage is that the algorithm may have difficulties finding very steep minima (as in the case of the Shekel test function [39], for example), and sometimes converges slowly to the global minimum because it does not search locally [120].

3.2.3 SO-M: Mixture Surrogate Model Algorithm Based on Dempster-Shafer Theory

Recall that the mixture surrogate model algorithm SO-M [98] described in Section 2.2 uses $n_0 = k + 2$ initial sample points generated by a Latin hypercube design. Model characteristics such as correlation coefficients and various error measures are computed by leave-one-out cross-validation, and

represent how well each individual model fits the data. Dempster-Shafer theory (DST) uses this information to compute the weights w_r of the models in the mixture in equation (2.2). The next sample site is chosen by solving an auxiliary optimization problem that uses a target value strategy similar to that proposed by Holmström *et al.* [64]. The optimization of the auxiliary problem is done by multistart accelerated random search with some restrictions on the search space that force the algorithm to switch between local and global search phases. The mixture model is updated in every iteration, and thus the models contributing to the mixture may change. The algorithm has been used to solve problems of up to six dimensions.

The advantage of using a mixture model is that if it is a priori unknown which surrogate model performs best for a given problem, the influence of "bad" models can be restricted and the influence of "good" models can be emphasized. The disadvantage of the mixture model approach is that finding the next sample point by optimizing an auxiliary function becomes more complex as the number of variables increases, and finding very steep minima (as in the case of the Shekel function) may be difficult. Furthermore, the leave-one-out cross-validation becomes computationally expensive as the number of sample sites increases, and if a kriging model is involved, the same drawbacks as in the EGO algorithm are encountered.

3.3 SO-M-c: A Stochastic Mixture Surrogate Model Algorithm

The mixture surrogate model algorithm SO-M [98] has been further developed with respect to two major aspects, namely the method of choosing the next sample site, and the computation of the model characteristics. Computing the model characteristics is done by leave-one-out cross-validation in the algorithm described in Chapter 2. This approach becomes a computational burden as the number of sample sites and the problem dimension increase because the models have to be fit for every sample site that is left out. The complexity of computing the model parameters depends on the problem dimension and the number n of already sampled points. Therefore, the leave-one-out cross-validation is applied only while the number of sample sites is less than 50. Thereafter, the $\tilde{k}$-fold cross-validation is applied, where $\tilde{k}$ is adjusted dynamically depending on the number of sample points already

evaluated.

In the $\tilde{k}$-fold cross-validation, iteratively a subset of $\tilde{k}$ sample points is taken out of the total set $\mathcal{S}$ of already sampled points. This subset is called the validation set. The remaining $n - \tilde{k}$ points constitute the training set. Using the points in the training set, all surrogate models are fit to the data. Each surrogate model is used to re-predict the objective function values of the sample points in the validation set. This is done for each subset, and based on the re-predicted and the true function values, correlation coefficients, maximum absolute errors, median absolute deviations, and root mean squared errors are computed for each model. These values are then transformed into basic probability assignments as described in Section 2.2, and DST is used to determine the weights of the individual models in the mixture in equation (2.2).

The second change made in the SO-M algorithm was to replace the method for finding the next sample site. Since studies with RBF models showed that a stochastic sampling strategy may be more successful than optimizing an auxiliary function [119], a randomized approach is used. The SO-M algorithm uses either the minimum of the response surface or a target value strategy. In the target value strategy, the search is either global (the best point is searched outside a densely sampled area of the variable domain), or local (the target value strategy is applied within the densely sampled area). Thus, SO-M is able to escape from local optima and explore other promising regions of the variable domain. In every iteration several target values are used, and therefore several auxiliary optimization problems have to be solved. For low-dimensional problems this is computationally inexpensive, but the computational complexity increases with the problem dimension. Therefore, a randomized sampling procedure similar to the candidate point approach by Regis and Shoemaker [119] has been used.

In the randomized sampling procedure candidate points (denoted by $\boldsymbol{\chi}_\jmath$, $\jmath = 1, \ldots, t$) for the next sample site are generated as follows. One group of candidates consists of points that are uniformly selected from the whole variable domain Ω. The points in the second group are generated by perturbing the best point found so far, i.e. $\mathbf{x}_{\text{best}} = \text{argmin}_{\iota=1,\ldots,n} f(\mathbf{x}_\iota)$. In contrast to [119] where *all* variables are perturbed with the *same perturbation range*, every variable is perturbed with probability

$$\mathbb{P} = \begin{cases} \max\{0.1, 5/k\} & \text{if } k > 5 \\ 1 & \text{otherwise} \end{cases}, \tag{3.4}$$

and three perturbation ranges have been used to obtain large, medium, and small perturbations. Thus, when generating the candidate points in the second group, randomly chosen variable values of $\mathbf{x}_{\text{best}}$ are perturbed by randomly adding or subtracting small, medium, or large perturbations. This allows the generation of a broader range of candidate points for which the magnitude of the perturbations of all variables may be different. Thus, a larger diversity of points in the vicinity of $\mathbf{x}_{\text{best}}$ is generated as compared to the approach in [119].

Two criteria are then used to find the "best" candidate point [119]. The first criterion is determined based on the distance of every candidate point to the set of already sampled points $\mathcal{S}$ ("distance criterion"). The second criterion is based on the objective function value predicted by the mixture surrogate model ("response surface criterion"). A score is computed for each candidate point as a weighted sum of both criteria. The candidate point with the best score becomes the point for doing the next expensive function evaluation.

A global search can be achieved by giving candidate points that are in relatively unexplored regions of the variable domain preference, i.e. by assigning a high weight to the distance criterion. On the other hand, once a promising point has been found, its vicinity should be explored more thoroughly. A local search can be achieved by giving a larger weight to the response surface criterion because the response surface is likely to predict lower objective function values for points in the vicinity of $\mathbf{x}_{\text{best}}$.

By repeatedly cycling through a weight pattern for the criteria, a repeated transition from global to local search is achieved [119]. The algorithm starts by assigning a large weight to the distance criterion and a low weight to the response surface criterion. The weight for the distance criterion is iteratively decreased, and the weight for the response surface criterion is increased. After the distance criterion weight has reached its minimum (and the weight for the response surface criterion has reached its maximum), it is re-initialized to the maximal value (minimal value). The advantage of this randomized sampling strategy is that no subproblem has to be optimized to find the next sample site, and savings in computation times are possible. The algorithm is implemented such that it satisfies the convergence conditions stated in [119] and convergence follows from Theorem 1 in [119].

Figure 3.1 illustrates the two criteria and the weighted scoring function for a one-dimensional problem (note that illustrated are the values scaled to [0,1]). The scoring function (solid green line) is not unimodal, and finding the

global minimum would require a global optimization algorithm. Numerical experiments on a set of test problems in Chapter 5 showed, however, that trying to find the global minimum of the scoring function and using the corresponding point as the next sample site does not lead to better results than the candidate point approach.

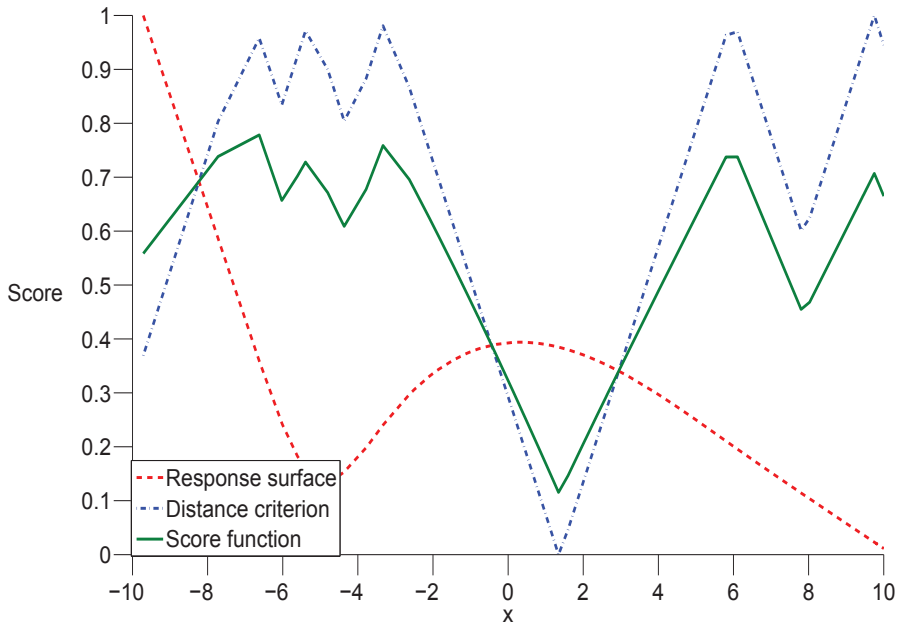


Figure 3.1: A one-dimensional example showing the response surface and distance criterion together with the weighted sum (score function) of both criteria.

A third alteration of the SO-M algorithm in Chapter 2 is that the surrogate models contributing to the mixture are fixed at the beginning of the algorithm, and during the optimization phase only their weights are adjusted, as opposed to the original implementation where the surrogate models contributing to the mixture could change throughout the optimization phase. By this approach it is possible to better examine the influence of single models on the mixture. The specific steps of the stochastic mixture surrogate model algorithm SO-M-c (**S**urrogate **O**ptimization - **M**ixture - **c**andidate sampling) are given in Algorithm 4.

Algorithm 4 ***SO-M-c: Mixture Surrogate Model Algorithm With Candidate Point Sampling Strategy***

1. *Create an initial experimental design using a symmetric Latin hypercube sampling strategy. Evaluate the computationally expensive objective function at the generated points.*

2. *Apply cross-validation to compute the characteristics of the desired surrogate models in the mixture:*

   - *If $n \leq 50$, use the leave-one-out cross-validation.*
   - *If $n > 50$, use the $\tilde{k}$-fold cross-validation, where $\tilde{k}$ increases with the number of function evaluations ($\tilde{k} = 10$ for $50 < n \leq 100$, $\tilde{k} = 20$ for $100 < n \leq 150$, $\tilde{k} = 30$ for $150 < n \leq 200$, etc.)*

3. *Use Dempster-Shafer Theory to determine the weights w_r of the chosen models in the mixture in equation (2.2).*

4. *Compute the parameters for each model contributing to the mixture.*

5. *Generate candidate points $\boldsymbol{\chi}_\jmath$, $\jmath = 1, \ldots, t$, by uniformly selecting points from the variable domain and by perturbing $\mathbf{x}_{\text{best}}$ as follows. With $\delta = \max_{i=1,\ldots,k}\{x_i^u - x_i^l\}$, the perturbations are computed as $g\delta\Upsilon$, where the standard deviation $g \in \{0.1, 0.01, 0.001\}$ is chosen randomly, and $\Upsilon \sim \mathcal{N}(0,1)$ is a random variable drawn from the standard normal distribution. Every variable of $\mathbf{x}_{\text{best}}$ is perturbed by adding $g\delta\Upsilon$ with probability $\mathbb{P}$ as defined in equation (3.4) to its value.*

6. *Compute the distance*

$$\Delta(\boldsymbol{\chi}_\jmath) = \min_{\iota=1,\ldots,n} \|\boldsymbol{\chi}_\jmath - \mathbf{x}_\iota\|_2, \tag{3.5}$$

   where $\|\cdot\|_2$ is the Euclidean norm in $\mathbb{R}^k$, and $\mathbf{x}_\iota \in \mathcal{S}$. Scale the values to the interval $[0,1]$:

$$V_D(\boldsymbol{\chi}_\jmath) = \begin{cases} \frac{\Delta_{\max} - \Delta(\boldsymbol{\chi}_\jmath)}{\Delta_{\max} - \Delta_{\min}} & \text{if } \Delta_{\min} \neq \Delta_{\max} \\ 1 & \text{otherwise} \end{cases}, \tag{3.6}$$

   where $V_D(\boldsymbol{\chi}_\jmath)$ is the scaled distance value for candidate $\boldsymbol{\chi}_\jmath$, $\Delta_{\max} = \max_{\jmath=1,\ldots,t}\{\Delta(\boldsymbol{\chi}_\jmath)\}$, and $\Delta_{\min} = \min_{\jmath=1,\ldots,t}\{\Delta(\boldsymbol{\chi}_\jmath)\}$. Eliminate candidates that are already contained in $\mathcal{S}$ from further consideration.

7. *Use the mixture model in equation (2.2) to predict the objective function values* $s_{\text{mix}}(\boldsymbol{\chi}_j)$ *of the candidate points, and scale these values to the interval* $[0, 1]$*:*

$$V_R(\boldsymbol{\chi}_j) = \begin{cases} \frac{s_{\text{mix}}(\boldsymbol{\chi}_j) - s_{\min}}{s_{\max} - s_{\min}} & \text{if } s_{\min} \neq s_{\max} \\ 1 & \text{otherwise} \end{cases}, \tag{3.7}$$

where $V_R(\boldsymbol{\chi}_j)$ *is the scaled predicted objective function value for candidate* $\boldsymbol{\chi}_j$*,* $s_{\max} = \max_{j=1,\ldots,t}\{s_{\text{mix}}(\boldsymbol{\chi}_j)\}$*, and* $s_{\min} = \min_{j=1,\ldots,t}\{s_{\text{mix}}(\boldsymbol{\chi}_j)\}$*.*

8. *Compute the weighted scores of the candidate points*

$$V(\boldsymbol{\chi}_j) = \omega_R V_R(\boldsymbol{\chi}_j) + \omega_D V_D(\boldsymbol{\chi}_j),\ j = 1, \ldots, t, \tag{3.8}$$

where $\omega_R + \omega_D = 1$*, and* $\omega_R \geq 0$ *is the weight for the response surface criterion, and* $\omega_D \geq 0$ *is the weight for the distance criterion. Set* $\omega_D = 1$ *in the first iteration and decreases the value for 0.1 in every following iteration until* $\omega_D = 0$*. Reinitialize the value to* $\omega_D = 1$*, and decrease the value anew, etc. Choose the candidate point with the best score (the smallest value for* V*) as next sample site.*

9. *Do the costly function evaluation at the chosen point.*

10. *If the maximum number of allowed function evaluations has not been reached, go to Step 2. Otherwise, return the best point found.*

3.4 Experimental Setup

The performance of the algorithms briefly reviewed in Section 3.2 (EGO, Gutmann's RBF algorithm) are compared to SO-M-c with various mixture models for a range of literature test problems and two applications that are described in Section 3.4.2. The parameter settings for the individual algorithms are summarized in Section 3.4.1.

3.4.1 Algorithms and Parameter Settings

A symmetric Latin hypercube design is used to generate the initial experimental design for each algorithm. In general any other Latin hypercube sampling could be used. The design contains $2(k + 1)$ points, which is twice the minimum number of data points needed for fitting a cubic RBF model with a linear polynomial tail for a k-dimensional problem. This number has been found to work better than the $k+2$ points used in SO-M. Furthermore,

$10k$ points as suggested for the EGO algorithm [74] are in general too many points because with a 30-dimensional problem, 300 points would be used, which is for computationally expensive global optimization problems already a very large number of evaluations. Each algorithm uses the same initial experimental design in order to obtain a comparison after an equal number of function evaluations.

An implementation of the EGO algorithm has been obtained from [48]. A kriging model with Gaussian correlation function is used as surrogate for the computationally expensive objective function. The parameters of the kriging model in the given implementation are computed by maximum likelihood estimation for which a genetic algorithm is applied. A genetic algorithm is also used for determining the point that maximizes the expected improvement in equation (3.2) for determining the next sample site.

In Gutmann's algorithm a cubic RBF interpolant has been used as surrogate model. A range of target values has been defined as suggested by [64], and the algorithm cycles through these target values during the optimization routine. The optimization subproblem of determining the minimum of the bumpiness measure for finding the next sample site has been solved with the dynamically dimensioned search algorithm [141]. The stopping criterion for this optimization subroutine is $1000k$ bumpiness measure evaluations, or a lack of improvement within 20 consecutive iterations. The goal is to find a good approximation of the optimum of the auxiliary function and to not spend too much time on the optimization subroutine which may cause the algorithm to become itself computationally expensive.

The SO-M-c algorithm has been used with different options for the surrogate models in the mixture in order to better examine how the influence of "bad" models, i.e. models that do not lead to good results in comparison with the best surrogate model, can be restricted by using mixtures with "good" surrogate models. In the following, the kriging model is abbreviated by "K", the RBF interpolant is abbreviated by "R", the polynomial regression model is denoted by "P", and the multivariate adaptive regression spline (MARS) is denoted by "M". Mixtures of these individual models are denoted by concatenating letters of the individual models. For example, "RM" denotes a mixture of the RBF model and the MARS model. To indicate which mixture model is used in the SO-M-c framework, the surrogate model abbreviation is given together with the sampling strategy abbreviation. For example K-c indicates that only the kriging model is used within SO-M-c. RKM-c denotes that SO-M-c uses a mixture of RBF, kriging, and MARS.

Note that if only the RBF model is used within the SO-M-c framework, then R-c is except for the altered sampling procedure and the cubic instead of the thin-plate spline RBF the same as the G-MSRBF algorithm by Regis and Shoemaker [119].

Furthermore, a sampling procedure that uses the minimum site of the response surface in every iteration to determine the next sample point has been used as alternative sampling strategy. The dynamically dimensioned search algorithm [141] with the same stopping criterion as used when minimizing the bumpiness measure described above is applied for minimizing the response surface. The algorithms using this sampling strategy are in the following denoted by SO-M-s (**S**urrogate **O**ptimization - **M**ixture - **s**urface minimum). Using the minimum site of the surface is often done in practice, but may lead to a search that is too local. If the response surface is unimodal, sample points are selected only in the vicinity of the surface minimum point, and the surface is thus refined only locally. The global fit of the surface may not be improved, and there is a potential risk that the algorithm may get trapped in a local optimum of the response surface (which does not necessarily have to be a local optimum or even a stationary point of the true objective function). On the other hand, if the response surface is multimodal, then, depending on the optimization routine used for finding the minimum of the surface, chances are that the next sample site is derived from a local minimum. If the surface has several local and/or global minima, the sample sites selected in each iteration may be in very different areas of the variable domain, and therefore a global exploration is possible. Hence, when using the minimum site of the response surface as sampling strategy, the multimodality of the surface and the optimization routine for finding the surface minimum play an important role and determine whether the search is local or global. Regis and Shoemaker [124] also showed that using a sampling criterion that is based on the various local and global minima of the response surface may lead to improved results.

Within the SO-M-c and SO-M-s algorithms, the Matlab toolbox DACE [88] has been used for computing the parameters of the kriging model. Furthermore, the Matlab toolbox ARESLab [71] has been used for adjusting the MARS model. The various mixtures used within SO-M-c and SO-M-s in the numerical experiments and their abbreviations are given in Table 3.1.

With the two sampling strategies, 22 versions of the mixture surrogate model algorithms have been compared, where 11 versions use the candidate point

Table 3.1: Individual and mixture surrogate models used within SO-M-c (candidate point sampling) and SO-M-s (minimum site of response surface sampling).

Individual models	
P-c/P-s	Full cubic polynomial regression model
R-c/R-s	Cubic RBF
K-c/K-s	Kriging with Gaussian correlation function
M-c/M-s	MARS
Two-model mixtures	
RK-c/RK-s	Cubic RBF and kriging with Gaussian correlation function
RM-c/RM-s	Cubic RBF and MARS
KM-c/KM-s	Kriging with Gaussian correlation function and MARS
RP-c/RB-s	Cubic RBF and full cubic polynomial regression model
KP-c/KP-s	Kriging with Gaussian correlation function and full cubic polynomial regression model
MP-c/MP-s	MARS and full cubic polynomial regression model
Three-model mixtures	
RKM-c/RKM-s	Cubic RBF, kriging with Gaussian correlation function, MARS

sampling procedure described in Section 3.3 (SO-M-c), and the remaining 11 versions use the minimum point of the response surface as next sample site (SO-M-s). In addition to these 22 algorithms, also EGO and Gutmann's RBF method have been included in the comparison. Therefore, all together 24 algorithms have been compared in the numerical experiments. 30 trials have been made for every test problem and every algorithm. The algorithms used the same initial experimental design for the same trial of a test problem in order to obtain a conclusive comparison. The maximum number of allowed function evaluations has been restricted to 400.

3.4.2 Test Problems

In total 13 test problems from the literature have been used to compare the algorithms. The problem dimensions range between 2 and 30. All problems

are box-constrained, i.e. except for the finite lower and upper bounds, there are no other constraints. The characteristics of the test problems are given in Table 3.2. The column "ID" contains the problem identification with which each problem will be referred to later on. The column "k" contains the problem dimension. The column "#Local/Global Minima" contains information about the number of local minima (LM) and global minima (GM). These 13 test problems are not really computationally expensive, but have characteristics typically encountered with black-box global optimization problems and are therefore suitable for comparing global optimization algorithms. The mathematical description of the generic test problems is given in Appendix B.

Furthermore, a 12-dimensional *in situ* groundwater bioremediation problem has been solved with the 24 algorithms. The objective is to purify contaminated groundwater at minimum cost using microorganisms to degrade pollutants. The bioremediation is stimulated by pumping water and electron acceptors (for example oxygen) or nutrients via injection wells into the contaminated groundwater to increase the ability of the microorganisms to degrade the contaminant in the aquifer. Injection and monitoring wells for measuring contamination are installed at fixed locations. The planning horizon is divided into management periods, and the goal is to determine the pumping rates at the beginning of each management period such that the contamination concentration at the end of the period is below a given threshold while minimizing the total costs of the clean-up. To determine the effect of the pumping policy, the biological transformation of the substances must be simulated numerically. Only after the simulation can it be determined whether the contamination constraints are satisfied. The contamination constraints are incorporated with a penalty approach in the objective function as done in [158]. This problem has multiple local minima, but the number of local and global minima is unknown. The reported global minimum in Table 3.2 is the best solution found during the numerical experiments.

A second application problem that has been used to compare the algorithms arises from energy generation using tethered kites [8, 22, 45, 67]. Kites exploit high-altitude winds to generate energy. Energy is produced by lifting and pulling forces of the kite, and then transmitted to an object at lower altitudes by the tether. A drum is used to reel the tether in and out. If the tether is let out at high tension, power is generated, whereas power is used when reeling the tether back in. The goal is to control the kite optimally so that the tension in the tether is much lower when reeling the kite in than when letting it out, and thus generating the maximum net amount of

power possible. Various variables can be adjusted to control the tension in the tether, for example, increasing the angle of attack of the kite increases the tension. The control variables in the considered application problem are the period of control of the kite, the rate of change of the period, the kite control angle, the tether release speed, and the initial direction angle. For computing the power generated by the kite, a set of differential equations has to be solved. Also for this problem the number of local and global optima is unknown, and the reported global minimum in Table 3.2 is the best solution found during the numerical experiments.

Table 3.2: Test problems for numerical experiments.
ID - problem identification, k - problem dimension,
LM - local minima, GM - global minima, (a) - best solution found;
see Appendix B for further information.

ID	Name	k	#Local/Global Minima	Global minimum
B	Branin [39]	2	3 GM, no LM	0.40
H3	Hartmann [39]	3	1 GM, 4 LM	- 3.86
H6	Hartmann [39]	6	1 GM, 4 LM	- 3.32
S5	Shekel 5 [39]	4	1 GM, 5 LM	-10.15
S7	Shekel 7 [39]	4	1 GM, 7 LM	-10.40
S10	Shekel 10 [39]	4	1 GM, 10 LM	-10.54
A15	Ackley [5]	15	1 GM, several LM	-22.72
Sc17	Schoen [128]	17	1 GM, several LM	23.01
L20	Levy [83]	20	1 GM, several LM	0
P24	Powell [109]	24	1 GM, no LM	0
M25	Michalewicz [93]	25	1 GM, 25! LM	-16.49 (a)
Sp27	Sphere [34]	27	1 GM, no LM	0
R30	Rastrigin [97, 142]	30	1 GM, several LM	-30.00
GW	Groundwater [158]	12	black-box	279.20 (a)
Kite	Kite	13	black-box	2501120 (a)

3.5 Numerical Results

The algorithms are in the following compared with respect to the relative errors between the optimal solution (or the best known solution) and the average function value over 30 trials after 400 evaluations found by the

algorithms. The test problems are subdivided into four groups, namely low-dimensional ($2 \leq k \leq 6$), mid-sized ($15 \leq k \leq 20$), large-dimensional ($24 \leq k \leq 30$), and application problems. For some test problems algorithm trials have failed, either due to ill-conditioning of the sample site matrix (DACE toolbox failed) or due to limited computer memory. The number of failed trials for each algorithm are summarized in Table 3.3. If a test problem is not mentioned in this table, it means that all 30 trials were successful for each algorithm. A dash in the table denotes that all 30 trials of the corresponding algorithm were successful for the respective problem. If there were failed trials for a test problem, the corresponding average relative errors as reported in the following sections are computed based on the successful trials.

Tables 3.4, 3.5, 3.7, and 3.10 show the mean relative errors for every problem in each class, and the relative errors averaged over all problems in each class (column "Mean"). The column "Algorithm" shows the number of the algorithm together with its abbreviation as defined in Table 3.1. The numbers are used in the box plots in Figures 3.2-3.5 for identifying the corresponding algorithms. There are three problems (L20, P24, Sp27) whose optimal values are zero, and thus the relative errors are not defined. For these problems, the objective function values averaged over all successful trials are reported for each algorithm in Tables 3.6, 3.8, and 3.9.

3.5.1 Low-dimensional Problems

The relative errors for the low-dimensional problems are shown in Table 3.4. Note that the DACE toolbox failed for few problem instances due to ill-conditioning of the sample site matrix when computing the parameters of the kriging model (see Table 3.3). The results show that EGO achieves the lowest relative error averaged over all problems. For the Branin and the Hartmann test functions, the SO-M-c algorithms lead to better solutions than SO-M-s. For the Shekel test functions, however, SO-M-s leads in general to slightly better results than SO-M-c.

A box plot of the relative errors over all test problems for each algorithm is shown in Figure 3.2. EGO has the lowest median, but the most outliers. Among the SO-M-c algorithms (algorithms 1-11), the RBF model (algorithm 1) and mixtures including the RBF model (algorithms 5, 6, 8, 9) attain results for which the medians of the relative errors are lowest. The same statement holds for the SO-M-s algorithms (algorithms 14-24). The plot shows that the strategy of using the minimum of the response surface

Table 3.3: Number of failed trials per problem and algorithm.

Algorithm	B	H3	L20	P24	M25	Sp27	R30
1 R-c	-	-	-	-	-	-	-
2 K-c	1	1	-	-	-	-	-
3 M-c	-	-	-	-	-	-	-
4 P-c	-	-	-	-	-	-	-
5 RK-c	1	1	-	4	23	24	24
6 RM-c	-	-	-	-	-	-	-
7 KM-c	1	1	-	-	-	-	-
8 RKM-c	1	1	-	4	23	24	25
9 RP-c	-	-	-	-	-	-	-
10 KP-c	1	1	-	-	-	-	-
11 MP-c	-	-	-	-	-	-	-
12 G	-	-	5	-	-	9	-
13 EGO	-	-	-	-	-	-	-
14 R-s	-	-	-	-	-	-	-
15 K-s	1	1	-	-	-	-	-
16 M-s	-	-	-	-	-	-	-
17 P-s	-	-	-	-	-	-	-
18 RK-s	1	1	-	9	25	21	27
19 RM-s	-	-	-	-	-	-	-
20 KM-s	1	1	-	-	-	-	-
21 RKM-s	1	1	-	9	25	21	27
22 RP-s	-	-	-	-	-	-	-
23 KP-s	1	1	-	-	-	-	-
24 MP-s	-	-	-	-	-	-	-

(14-24) leads to better results (lower medians and smaller spread) than when using the candidate point approach (algorithms 1-11), but the number of outliers is larger. For both sampling strategies, the polynomial model (algorithms 4 and 17) and the mixture of the polynomial and the MARS model (algorithms 11 and 24) performed worst. The figure also shows that the mixture models perform better or at least equally well as when using single models (except for the mixtures of the polynomial and MARS). Thus, if it is a priori unknown which model will be the best for a given problem,

Table 3.4: Mean relative errors for low-dimensional problems; best result is marked by boxes.

Algorithm	B	H3	H6	S5	S7	S10	Mean
1 R-c	0.0025e-01	0.0000e-02	0.1094e-01	4.9528e-01	5.3215e-01	4.1087e-01	2.4158e-01
2 K-c	0.0022e-01	0.0000e-02	0.1078e-01	4.6915e-01	4.8925e-01	4.1450e-01	2.3065e-01
3 M-c	0.0997e-01	0.0766e-02	0.9254e-01	6.4081e-01	6.9793e-01	7.2039e-01	3.6040e-01
4 P-c	0.9944e-01	0.4818e-02	1.1502e-01	6.7663e-01	8.0138e-01	8.1193e-01	4.1820e-01
5 RK-c	0.0034e-01	0.0001e-02	0.1198e-01	4.8750e-01	4.7486e-01	4.2010e-01	2.3246e-01
6 RM-c	0.0069e-01	0.0068e-02	0.2012e-01	5.1087e-01	4.0532e-01	4.6461e-01	2.3361e-01
7 KM-c	0.0121e-01	0.0046e-02	0.1873e-01	4.9989e-01	4.8816e-01	4.6596e-01	2.4567e-01
8 RKM-c	0.0068e-01	0.0015e-02	0.1237e-01	4.7104e-01	4.3645e-01	4.6870e-01	2.3154e-01
9 RP-c	0.0061e-01	0.0104e-02	0.1863e-01	3.9704e-01	5.1969e-01	4.4486e-01	2.3015e-01
10 KP-c	0.0241e-01	0.0082e-02	0.1364e-01	4.9246e-01	5.3617e-01	4.5067e-01	2.4924e-01
11 MP-c	0.0571e-01	0.1986e-02	1.2036e-01	6.6858e-01	7.0808e-01	7.7802e-01	3.8046e-01
12 G	2.2293e-01	3.2546e-02	2.6735e-01	1.9879e-01	1.9616e-01	2.1628e-01	1.8901e-01
13 EGO	0.1229e-01	0.0002e-02	0.0993e-01	0.3371e-01	0.0039e-01	1.6603e-01	0.3706e-01
14 R-s	1.8913e-01	2.7638e-02	2.4374e-01	0.3318e-01	0.1691e-01	0.0001e-01	0.8510e-01
15 K-s	1.9122e-01	2.8333e-02	2.4674e-01	3.7286e-01	1.1843e-01	0.6785e-01	1.7090e-01
16 M-s	3.7455e-01	2.7638e-02	2.4374e-01	6.3093e-01	2.6170e-01	1.9991e-01	2.8974e-01
17 P-s	1.9441e-01	2.7638e-02	2.5116e-01	5.1267e-01	5.7986e-01	4.4185e-01	3.3460e-01
18 RK-s	1.7178e-01	2.6884e-02	2.3703e-01	0.4976e-01	0.1691e-01	0.3818e-01	0.9009e-01
19 RM-s	1.7498e-01	2.7871e-02	2.3703e-01	0.3318e-01	0.4138e-01	0.0001e-01	0.8574e-01
20 KM-s	1.7178e-01	2.6884e-02	2.3887e-01	2.4031e-01	1.3546e-01	0.6830e-01	1.4694e-01
21 RKM-s	1.7178e-01	2.6884e-02	2.3703e-01	0.4976e-01	0.0001e-01	0.0001e-01	0.8091e-01
22 RP-s	1.6940e-01	2.7871e-02	2.3703e-01	0.3318e-01	0.0001e-01	0.0001e-01	0.7792e-01
23 KP-s	1.7178e-01	2.6884e-02	2.4090e-01	2.4881e-01	0.1691e-01	0.1697e-01	1.2038e-01
24 MP-s	2.0393e-01	2.7871e-02	2.3728e-01	6.7495e-01	5.3425e-01	4.0165e-01	3.4665e-01

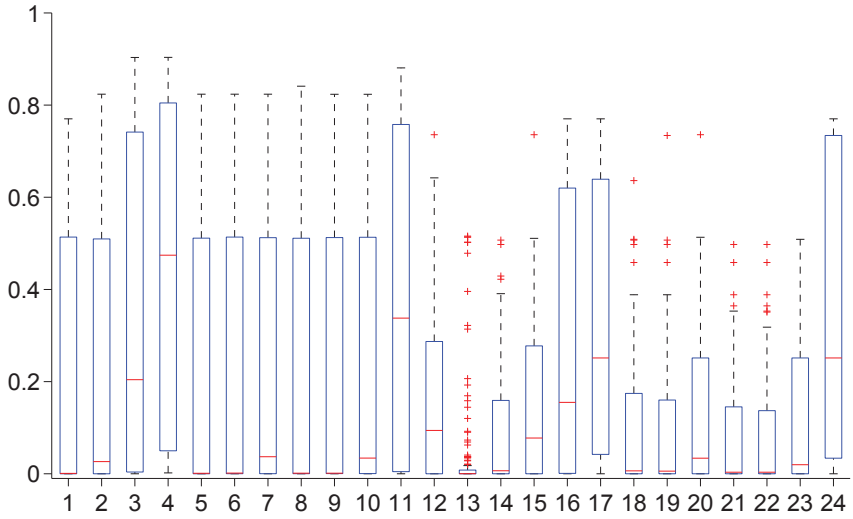


Figure 3.2: Distribution of relative errors for low-dimensional problems.

using a mixture that contains an RBF model is advisable.

3.5.2 Medium-sized Problems

The relative errors for the medium-sized problems are shown in Table 3.5. Note that for the 20-dimensional Levy problem the relative errors are not defined because the optimum has the objective function value 0. For this test function the means of the objective function value over 30 trials after 400 function evaluations for each algorithm are given in Table 3.6. Gutmann's algorithm failed for five trials for this problem, and therefore the given results are averaged over the remaining 25 successful trials. For the Levy function, the results show that EGO outperforms all other algorithms.

The results in Table 3.5 show that the average of the relative errors over the Ackley and Schoen function is lowest when a mixture of RBF and the polynomial is used together with the candidate point sampling strategy. For the 17-dimensional Schoen function, the SO-M-c algorithms (algorithms 1-11) perform better than the SO-M-s algorithms (algorithms 14-24). Also the mean of the relative errors over both problems is, except for the polynomial model, lower for SO-M-c. This fact is also reflected in the box plot in Figure 3.3. For the SO-M-c algorithms, the RBF model (algorithm 1) and

the mixture of RBF and the polynomial (algorithm 9) perform best. EGO (algorithm 13) has a slightly higher median than R-c and RP-c. Among the SO-M-s algorithms, using only the RBF model as response surface performs best (lowest median, algorithm 14). Thus also for this problem class using a (mixture) surrogate that contains an RBF model should be chosen.

Table 3.5: Mean relative errors for medium-sized problems; best result is marked by boxes. For results of L20, see Table 3.6.

Algorithm	A15	Sch17	Mean
1 R-c	0.0996	0.1261	0.1128
2 K-c	0.6779	0.0009	0.3394
3 M-c	0.6189	1.6183	1.1186
4 P-c	0.8395	2.7743	1.8069
5 RK-c	0.3668	0.0990	0.2329
6 RM-c	0.1779	0.3763	0.2771
7 KM-c	0.6626	0.3315	0.4971
8 RKM-c	0.5293	0.2751	0.4022
9 RP-c	0.0839	0.0549	0.0694
10 KP-c	0.6662	0.0298	0.3480
11 MP-c	0.6329	1.6021	1.1175
12 G	0.5917	2.7088	1.6503
13 EGO	0.1072	0.2334	0.1703
14 R-s	0.0074	2.7086	1.3580
15 K-s	0.6598	2.7086	1.6842
16 M-s	0.5610	2.7101	1.6355
17 P-s	0.8384	2.8155	1.8270
18 RK-s	0.1416	2.7087	1.4252
19 RM-s	0.0470	2.7086	1.3778
20 KM-s	0.5526	2.7086	1.6306
21 RKM-s	0.2946	2.7086	1.5016
22 RP-s	0.0847	2.7086	1.3966
23 KP-s	0.6709	2.7086	1.6898
24MP-s	0.8028	2.7094	1.7561

Table 3.6: Levy-20 test problem, $k = 20$; 1 global minimum, several local minima, minimization problem; best result is marked by box; Standard error of means (SEM) in italic.

Algorithm	Statistic		Algorithm	Statistic	
1 R-c	mean	259.46	14 R-s	mean	149.73
	SEM	*36.28*		*SEM*	*20.04*
2 K-c	mean	550.78	15 K-s	mean	787.43
	SEM	*27.85*		*SEM*	*34.48*
3 M-c	mean	408.05	16 M-s	mean	146.27
	SEM	*19.28*		*SEM*	*4.20*
4 P-c	mean	2105.82	17 P-s	mean	1475.62
	SEM	*70.48*		*SEM*	*12.09*
5 RK-c	mean	451.37	18 RK-s	mean	196.90
	SEM	*32.98*		*SEM*	*27.21*
6 RM-c	mean	272.01	19 RM-s	mean	130.70
	SEM	*27.75*		*SEM*	*18.16*
7 KM-c	mean	517.76	20 KM-s	mean	533.97
	SEM	*27.45*		*SEM*	*16.84*
8 RKM-c	mean	492.73	21 RKM-s	mean	144.07
	SEM	*31.02*		*SEM*	*22.41*
9 RP-c	mean	198.63	22 RP-s	mean	313.83
	SEM	*31.52*		*SEM*	*27.24*
10 KP-c	mean	583.74	23 KP-s	mean	760.40
	SEM	*30.80*		*SEM*	*41.38*
11 MP-c	mean	397.47	24 MP-s	mean	836.30
	SEM	*27.87*		*SEM*	*14.89*
12 G	mean	364.32			
	SEM	*16.86*			
13 EGO	mean	31.70			
	SEM	*1.35*			

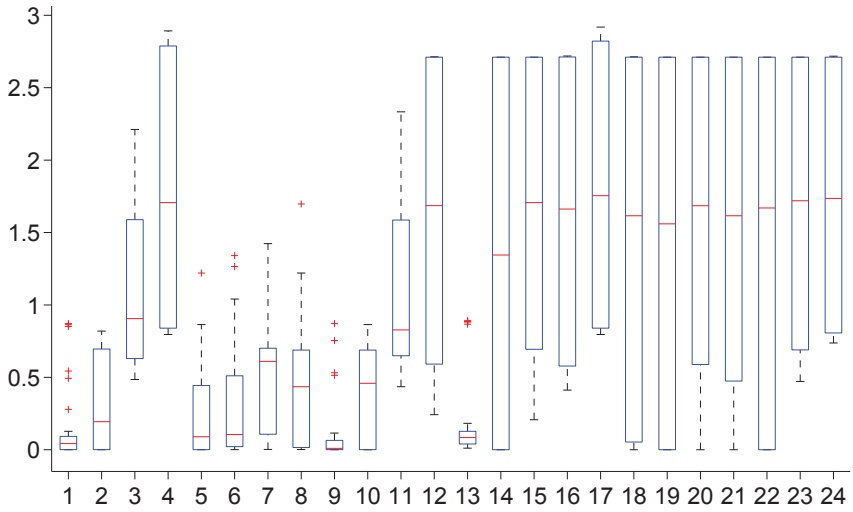


Figure 3.3: Distribution of relative errors for medium-sized problems.

3.5.3 Large-dimensional Problems

The relative errors over the large-dimensional problems are shown in Table 3.7. Note that the optimal values of the Powell and the Sphere function are zero, and therefore the relative errors are not defined. The average objective function values and the standard errors of the means reached by the algorithms for these problems after 400 function evaluations are given in Tables 3.8 and 3.9, respectively. The algorithms involving a mixture of RBF and kriging (algorithms 5, 8, 18, and 21) failed for several trials due to limited computer memory. The results in Table 3.7 show that the mixture of the RBF model and the MARS model when using the minimum of the response surface as sampling strategy (algorithm 19) performs on average better than all other algorithms. RM-s also performs best for the Powell function, which is in particular a unimodal function. EGO performs for the large-dimensional problems on average worse than the other algorithms, except when the MARS (algorithm 3), the polynomial (algorithms 4 and 17), or a mixture of MARS and polynomial (algorithms 11 and 24) are used as response surface.

Figure 3.4 shows the box plots of the relative errors for each algorithm. The figure shows that using only the polynomial regression model or a mixture of the polynomial and MARS perform for both sampling strategies

worst (algorithms 4, 11, 17, 24). Also for this problem class it can be seen that using the RBF model (algorithms 1 and 14), or a mixture including the RBF model (algorithms 6, 9, 19, 22) performs in general better than other mixtures. For the large-dimensional problems also Gutmann's method (algorithm 12) outperforms EGO (algorithm 11).

Table 3.7: Mean relative errors for large-dimensional problems; best result is marked by boxes. For results of P24 and Sp27, see Tables 3.8 and 3.9, respectively.

Algorithm	M25	R30	Mean
1 R-c	0.3911	0.5189	0.4550
2 K-c	0.2979	0.8435	0.5707
3 M-c	0.5109	1.1617	0.8363
4 P-c	0.6437	2.2174	1.4305
5 RK-c	0.3986	0.5988	0.4987
6 RM-c	0.3783	0.5734	0.4759
7 KM-c	0.2890	0.8591	0.5740
8 RKM-c	0.2821	0.6395	0.4608
9 RP-c	0.4227	0.5411	0.4819
10 KP-c	0.3501	0.8316	0.5908
11 MP-c	0.5723	1.2493	0.9108
12 G	0.5652	0.0744	0.3198
13 EGO	0.4530	1.0409	0.7469
14 R-s	0.5614	0.0333	0.2974
15 K-s	0.6326	0.9744	0.8035
16 M-s	0.5481	0.8978	0.7229
17 P-s	0.6545	2.2363	1.4454
18 RK-s	0.5544	0.1333	0.3439
19 RM-s	0.5591	0.0144	0.2868
20 KM-s	0.6172	0.4667	0.5420
21 RKM-s	0.5710	0.1333	0.3522
22 RP-s	0.5803	0.1389	0.3596
23 KP-s	0.6140	0.8222	0.7181
24 MP-s	0.6304	2.1681	1.3993

Table 3.8: Powell-24 test problem, $k = 24$, 1 global minimum, no other local minima, minimization problem; best result is marked by box; Standard error of means (SEM) in italic.

Algorithm	Statistic		Algorithm	Statistic	
1 R-c	mean	504.10	14 R-s	mean	89.93
	SEM	*79.09*		*SEM*	*12.57*
2 K-c	mean	2330.65	15 K-s	mean	4031.00
	SEM	*157.73*		*SEM*	*519.888*
3 M-c	mean	4618.12	16 M-s	mean	4495.25
	SEM	*393.68*		*SEM*	*210.42*
4 P-c	mean	10665.02	17 P-s	mean	12091.29
	SEM	*482.43*		*SEM*	*745.62*
5 RK-c	mean	1491.18	18 RK-s	mean	281.29
	SEM	*104.05*		*SEM*	*55.57*
6 RM-c	mean	1112.35	19 RM-s	mean	81.47
	SEM	*170.86*		*SEM*	*11.32*
7 KM-c	mean	2487.67	20 KM-s	mean	1284.43
	SEM	*175.44*		*SEM*	*239.18*
8 RKM-c	mean	1898.06	21 RKM-s	mean	544.43
	SEM	*174.68*		*SEM*	*152.58*
9 RP-c	mean	578.80	22 RP-s	mean	147.23
	SEM	*128.97*		*SEM*	*21.42*
10 KP-c	mean	2515.37	23 KP-s	mean	1830.13
	SEM	*139.69*		*SEM*	*407.59*
11 MP-c	mean	4673.35	24 MP-s	mean	12093.97
	SEM	*317.60*		*SEM*	*746.34*
12 G	mean	241.73			
	SEM	*317.60*			
13 EGO	mean	363.05			
	SEM	*28.26*			

Table 3.9: Sphere-27 test problem, $k = 27$; 1 global minimum, no other local minima, minimization problem; best result is marked by box; Standard error of means (SEM) in italic.

Algorithm	Statistic		Algorithm	Statistic	
1 R-c	mean	1.32	14 R-s	mean	1.47
	SEM	*0.87*		*SEM*	*0.33*
2 K-c	mean	26.44	15 K-s	mean	71.40
	SEM	*1.33*		*SEM*	*5.56*
3 M-c	mean	45.87	16 M-s	mean	80.07
	SEM	*2.91*		*SEM*	*1.88*
4 P-c	mean	154.78	17 P-s	mean	154.78
	SEM	*2.73*		*SEM*	*2.73*
5 RK-c	mean	12.07	18 RK-s	mean	3.33
	SEM	*0.77*		*SEM*	*0.82*
6 RM-c	mean	3.77	19 RM-s	mean	0.37
	SEM	*1.44*		*SEM*	*0.15*
7 KM-c	mean	29.82	20 KM-s	mean	19.83
	SEM	*1.94*		*SEM*	*3.03*
8 RKM-c	mean	17.65	21 RKM-s	mean	9.67
	SEM	*3.91*		*SEM*	*2.69*
9 RP-c	mean	0.45	22 RP-s	mean	2.20
	SEM	*0.04*		*SEM*	*0.42*
10 KP-c	mean	29.83	23 KP-s	mean	38.37
	SEM	*1.15*		*SEM*	*5.48*
11 MP-c	mean	50.08	24 MP-s	mean	154.78
	SEM	*3.44*		*SEM*	*2.73*
12 G	mean	0.00			
	SEM	*0.00*			
13 EGO	mean	13.68			
	SEM	*0.70*			

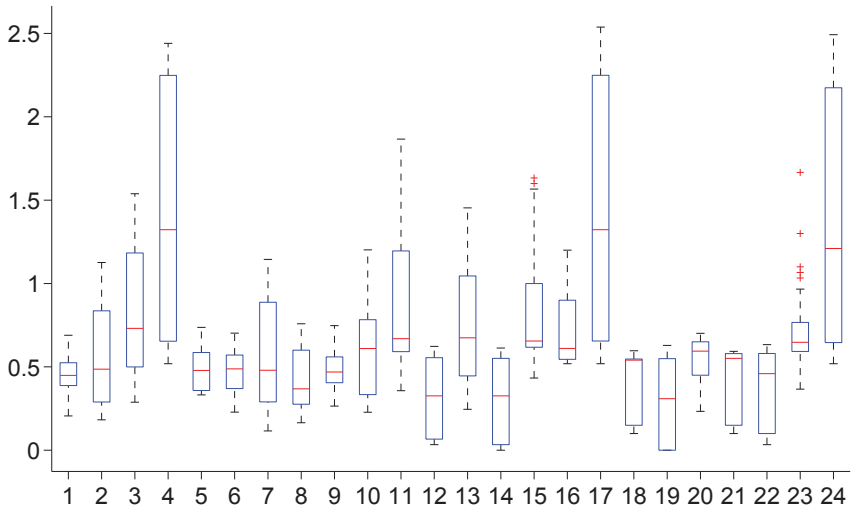


Figure 3.4: Distribution of relative errors for large-dimensional problems.

3.5.4 Application Problems

The relative errors and the mean of the relative errors for both problems are given in Table 3.10. For the groundwater bioremediation application the RBF model with the candidate point sampling approach (algorithm 1) leads to the best results. For the kite application the kriging model together with the candidate point sampling (algorithm 2) attain on average the lowest relative errors. R-c, K-c, and RP-c perform almost equally well. EGO performs on average worse than most SO-M-c algorithms (except when only the polynomial regression model is used (algorithm 4)). Gutmann's RBF method performs worse than all other algorithms. The box plot in Figure 3.5 also shows that the candidate point sampling strategy (algorithms 1-11) leads in general to better results than when using the minimum of the response surface (algorithms 14-24).

Table 3.10: Mean relative errors for application problems; best result is marked by boxes.

Algorithm	GW	Kite	Mean
1 R-c	0.2298	0.5319	0.3808
2 K-c	0.2985	0.4802	0.3894
3 M-c	0.3683	0.5250	0.4467
4 P-c	1.1301	0.5251	0.8276
5 RK-c	0.2749	0.5509	0.4129
6 RM-c	0.3731	0.5687	0.4709
7 KM-c	0.4094	0.4926	0.4510
8 RKM-c	0.3807	0.5183	0.4495
9 RP-c	0.2736	0.5203	0.3969
10 KP-c	0.4038	0.5103	0.4571
11 MP-c	0.4694	0.5929	0.5311
12 G	2.7298	0.5708	1.6503
13 EGO	1.1159	0.5007	0.8083
14 R-s	2.1496	0.5631	1.3564
15 K-s	2.2332	0.5387	1.3859
16 M-s	2.1506	0.5600	1.3553
17 P-s	2.6970	0.6161	1.6565
18 RK-s	1.9660	0.5401	1.2531
19 RM-s	1.9660	0.5496	1.2578
20 KM-s	2.0348	0.4845	1.2596
21 RKM-s	1.9660	0.5035	1.2348
22 RP-s	1.9660	0.5222	1.2441
23 KP-s	2.0062	0.5066	1.2564
24 MP-s	1.9909	0.5287	1.2598

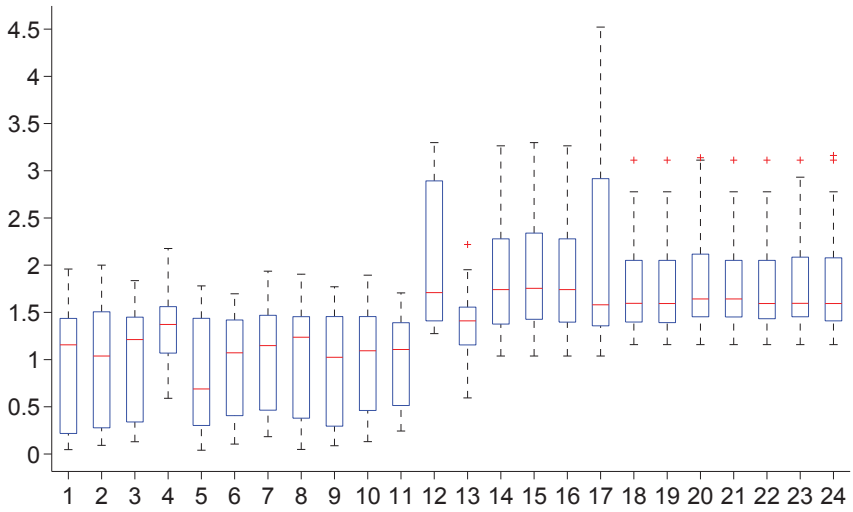


Figure 3.5: Distribution of relative errors for application problems.

3.6 Conclusions

In this chapter the mixture surrogate model algorithm SO-M introduced in Chapter 2 has been further developed with respect to two aspects. Firstly, the strategy for selecting the next sample site in each iteration has been changed to using a stochastic approach (algorithm SO-M-c). In every iteration candidate points for the next sample site are generated by uniformly selecting points from the whole variable domain, and by perturbing the variable values of the best point found so far. Scoring criteria are used to determine the best candidate, which in turn is used as sample site for doing the expensive function evaluation. By changing the weights of the criteria a repeated transition from global to local search could be achieved.

The second improvement has been made with respect to computing the characteristics of the models in the mixture. Since the leave-one-out cross-validation in SO-M becomes computationally too expensive as the number of sample points and the problem dimension increase, a $\tilde{k}$-fold cross-validation has been applied, where $\tilde{k}$ is adjusted dynamically throughout the algorithm and depends on the number of sample sites.

The algorithm SO-M-c has been compared in numerical experiments to the efficient global optimization algorithm (EGO) and Gutmann's RBF

method. Moreover, a second sampling strategy that uses the minimum point of the response surface has been used within the mixture model algorithm (algorithm SO-M-s). Several mixture surrogate models have been used in the comparison. As opposed to SO-M, however, the models contributing to the mixtures in SO-M-c and SO-M-s were fixed, and thus they could not change between iterations. Only their weights were re-adjusted in every iteration. By this approach it was possible to examine which surrogate models perform in general well, and which models should be avoided.

The comparison of the various algorithms, (mixture) surrogate models, and sampling strategies showed that there is in general not one single algorithm that outperforms all other algorithms (no free lunches [151]). The numerical experiments showed that for the low-dimensional problems ($2 \leq k \leq 6$) the SO-M-s algorithms perform on average better than the SO-M-c algorithms. EGO had the best average performance for these low-dimensional problems and Gutmann's RBF method was among the worst. For the medium sized problems ($15 \leq k \leq 20$) and both application problems the SO-M-c algorithms proved in general superior as compared to using the SO-M-s algorithms. For the large-dimensional problems ($24 \leq k \leq 30$) mixture models performed better than single models for both sampling strategies.

Mixture models containing a cubic radial basis function interpolant performed for most test problems best, while using only a cubic polynomial regression model, MARS, or a mixture of the two often led to poor results. However, in combinations with RBF or kriging, the performance of MARS and the polynomial model could be significantly improved, and these mixtures often outperformed even the best individual model contributing to the mixture. Thus, if it is a priori unknown which surrogate model should be used for a given black-box optimization problem, mixture surrogate models that contain an RBF model should be chosen.

The problem of increasing computation times of the mixture model algorithms can easily be avoided by parallel or distributed implementations. The parameters and cross-validation for each model in the mixture can be done on different processors, and thus the computation time can be reduced to almost the same time required if only a single model was used. Gutmann's method on the other hand cannot be sped up by using parallelization because minimizing the bumpiness measure is the driving computational expense, and this is an iterative procedure.

In conclusion, a mixture surrogate model algorithm should be chosen if nothing about the behavior of the black-box objective function is a priori known, especially if there is the possibility of using several processors. Regarding the sampling strategy the candidate point approach is very promising especially for problems with larger dimensions ($k \geq 12$). A more dynamic adjustment of the perturbation probability when generating candidate points from the best point found so far as done in the DYCORS algorithm [123] could be examined in the future in connection with mixture models. Moreover, the scoring criteria of the candidate point approach for mixture models containing kriging could be changed such that the expected improvement becomes a criterion. Since the uncertainty of predictions is computed along with the predictions in kriging, this information could be exploited in addition to the response surface and distance criterion to determine the best candidate point. These extensions are, however, left for future research and are thus not further discussed in this thesis.

Chapter 4

SO-MI: A Surrogate Model Algorithm for Computationally Expensive Nonlinear Mixed-Integer Black-Box Global Optimization Problems

Abstract

This chapter introduces a surrogate model based algorithm for computationally expensive *mixed-integer* black-box global optimization problems with both binary and non-binary integer variables that may have computationally expensive constraints. The goal is to find accurate solutions with relatively few function evaluations. A radial basis function surrogate model (response surface) is used to select candidates for integer and continuous decision variable points at which the computationally expensive objective and constraint functions are to be evaluated. In every iteration multiple new points are selected based on different methods, and the function evaluations are done in parallel. The algorithm converges to the global optimum almost surely. The performance of this new algorithm, SO-MI, is compared to a branch and bound algorithm for nonlinear problems, a genetic algorithm, and the NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search, version 3.5.0) algorithm for mixed-integer problems on sixteen test problems from the literature (constrained, unconstrained, unimodal, and multimodal problems), as well as on two application problems arising from structural

optimization, and three application problems from optimal reliability design. The numerical experiments show that SO-MI reaches significantly better results than the other algorithms when the number of function evaluations is very restricted (200 to 300 evaluations).

Abbreviations and Nomenclature

B&B	Branch and bound algorithm
FEA	Finite element analysis
GA	Genetic algorithm
LC	Linear constraints
MINLP	Mixed-integer nonlinear programming
MM	Multimodal
MTTF	Mean time to failure
NLC	Nonlinear constraints
NOMAD	Nonsmooth Optimization by Mesh Adaptive Direct Search
RBF	Radial basis function
SEM	Standard error of the means
SO-MI	Surrogate Optimization - Mixed Integer
UM	Unimodal
$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integer numbers
$\mathbf{w}$	Mixed-integer decision variable vector, see equation (4.1e)
$\mathbf{w}^T$	Transpose of $\mathbf{w}$
$\mathbf{x}$	Continuous decision variables, see equation (4.1e)
$\mathbf{u}$	Discrete decision variables, see equation (4.1e)
i_1	Index for continuous variables, see equation (4.1c)
i_2	Index for discrete variables, see equation (4.1d)
$x^l_{i_1}$, $x^u_{i_1}$	Lower and upper variable bounds for continuous variables, see equation (4.1c)
$u^l_{i_2}$, $u^u_{i_2}$	Lower and upper variable bounds for discrete variables, see equation (4.1d)
$f(\cdot)$	Objective function, see equation (4.1a)
$c_j(\cdot)$	jth constraint function, see equation (4.1b)
k_1	Dimension of the continuous variables, see equation (4.1c)
k_2	Dimension of the discrete variables, see equation (4.1d)
k	Problem dimension ($k = k_1 + k_2$)
m	Number of constraints
Ω_b	Box-constrained variable domain
Ω^D	Discrete box-constrained variable domain
Ω^C	Continuous box-constrained variable domain
Ω	Variable domain defined by box-constraints, linear, nonlinear, and integrality constraints
n_0	Number of points in initial experimental design

$\mathcal{S}$	Set of n already evaluated points
$\mathbf{w}_\iota$	ιth already sampled point, $\iota = 1, \ldots, n$
n	Number of already sampled points
$v(\cdot)$	Constraint violation function, see equation (4.2)
$f_p(\cdot)$	Objective function augmented with penalty term, see equation (4.3)
p	Penalty factor, see equation (4.3)
$\tilde{f}_p(\cdot)$	Objective function augmented with penalty term, see equation (4.5)
$v_s(\cdot)$	Scaled constraint violation function, see equation (4.4)
$s_b(\cdot)$	Radial basis function interpolant, see equation (1.9)
$\mathbf{w}_{\min}$	Best feasible point found so far
$f_{\min}$	Best feasible objective function value found so far
$f_{\max}$	Worst feasible objective function value found so far
$f(\mathbf{x}\|\mathbf{u})$	Objective function when the discrete variables are fixed to $\mathbf{u}$, in that case f is dependent only on continuous variables $\mathbf{x}$
$\boldsymbol{\chi}_\jmath$	$\jmath$th candidate point, $\jmath = 1, \ldots, t$
h_1, h_2	Standard deviations of perturbations
$V_D(\cdot)$	Score for distance criterion, see equation (3.6)
$V_R(\cdot)$	Score for response surface criterion, see equation (3.7)
$V(\cdot)$	Weighted score, see equation (3.8)
H_0, H_1	Null hypothesis and alternative hypothesis

4.1 Introduction and Motivation

In many application areas such as logistics, engineering design, portfolio optimization or energy generation problems may be encountered where integrality constraints are imposed on several variables. The algorithms developed and compared in Chapters 2 and 3 are suitable only for continuous optimization problems, and other algorithms for solving computationally expensive mixed-integer problems are needed. Mixed-integer optimization problems are in general NP-hard and difficult to solve. Most algorithms for solving mixed-integer optimization problems are based on branch and bound methods, or some evolutionary strategy such as genetic algorithms [31, 50, 59] or ant colony optimization (for example MIDACO [127]). While several software packages are available for solving problems with convex objective and constraint functions (for example DICOPT [145] or BONMIN [20]), algorithms for solving nonconvex problems are rather scarce and are usually based on problem reformulation and convexification strategies [105, 106, 133]. However, when dealing with black-box objective and constraint functions, where all that is known about the problem is the deterministic output for a given input variable vector, reformulating the problem is not possible. Moreover, if a single function evaluation requires a time consuming simulation (from several minutes to several hours or even days), the number of function evaluations for finding a good approximation of the global optimum has to be as low as possible.

The branch and bound algorithm is based on recursively dividing the set of possible solutions into smaller sets (nodes) during the so-called branching step, and a tree structure evolves. During the bounding step an upper and lower bound on the objective function value is calculated for every solution subset. This requires minimizing the costly objective function at least once because in order to obtain the lower bound, a relaxed problem (where certain integer variables are assumed to be continuous) must be minimized. For obtaining an upper bound, the variables in the solution of the relaxed problem can be rounded, for example, to the closest integers (which does not necessarily yield a feasible solution). In case the feasible upper bound of a node, say N_1, is lower than the feasible lower bound of node N_2, the node N_2 can be deleted from the search (pruning). While the feasible upper bounds on the solution at the nodes are non-increasing, the feasible lower bounds are non-decreasing, and this is typically used to show convergence to the global optimum.

For solving computationally expensive black-box functions, however, branch and bound might not be suitable because to obtain valid lower bounds for multimodal problems the global optimum of the relaxed problem must be found, which requires the application of a global optimization algorithm. Furthermore, some application problems may not allow a relaxation of the integer variables because this may cause the code evaluating the objective function to fail.

Evolutionary algorithms such as genetic algorithms [63] mimic the natural process of evolution by survival of the fittest. An initial population of individuals (solutions) is generated randomly. The fittest individuals are chosen for reproduction. Their offspring are generated by random crossover and mutation operations. The advantage of this algorithm type is that it is possible to escape from a local optimum. In general, a large number of function evaluations must be done to find a good approximation of the global optimum due to the number of individuals in the population and the number of generations. Thus, genetic algorithms may not be suitable for solving computationally expensive optimization problems.

Other algorithms for mixed-integer problems include cutting plane methods [54, 147, 148], and outer approximation methods [43, 76]. BARON [137, 125] is a widely used algorithm for global optimization of algebraic nonlinear and mixed-integer nonlinear problems. However, these algorithms require the knowledge of the algebraic formulation of the problems, or that the problems are of a specific form (for example, convexity of the objective function and constraints, or linearity in integer variables) and are thus not suitable for solving black-box problems.

A derivative-free algorithm often referred to in the literature is NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search, here version 3.5.0 is used) [2, 3, 4]. NOMAD uses a mesh adaptive direct search algorithm designed for solving constrained black-box optimization problems. The mesh adaptive direct search is an extension of generalized pattern search algorithms for which superior convergence properties can be shown [1].

NOMAD is applicable for mixed-variable problems, and by using the variable neighborhood search [58, 95] option in the C++ implementation, it is able to escape from local minima [10]. The mixed-variable pattern search can be shown to guarantee first-order optimality conditions with respect to the continuous variables, and local optimality in the mixed-integer case is defined by user-specified neighboring points [11]. Although the C++

implementation of NOMAD is able to use surrogate models during the search, the user has to implement the desired model him/herself.

Furthermore, there are no extensive numerical studies of using NOMAD for solving constrained mixed-integer problems. Liuzzi *et al.* [87] compared their derivative-free algorithms to NOMAD on many low-dimensional mixed-integer local optimization problems and few global optimization test problems with at most 20 dimensions that have only bound constraints, but application problems were not considered.

Despite their success in solving continuous optimization problems, surrogate model algorithms have only started to be considered for solving mixed-integer optimization problems [14, 33, 60, 64, 116]. Davis and Ierapetritou [33] suggest an algorithm that is able to handle problems with noisy data, and that have continuous and binary variables. A branch and bound framework is coupled with kriging response surfaces and response surface methodology to obtain a balance of local and global search while using a very restricted number of function evaluations.

Holmström *et al.* [64] describe in their paper an adaptive radial basis function algorithm for solving mixed-integer optimization problems. The algorithm uses mixed-integer subsolvers from the commercial TOMLAB optimization environment to solve an auxiliary problem to determine the points where to evaluate the expensive objective function in every iteration of the algorithm. Holmström *et al.* [64] considered mainly low dimensional problems (one test problem with 11 dimensions and eight integer variables, and other test problems with six or fewer dimensions with one to four integer variables). The response surface is not necessarily unimodal, and solving the auxiliary problem may require itself the application of a global optimization algorithm. Furthermore, constraints have to either be computationally cheap, or otherwise added as penalty term to the objective function value. The adjustment of the penalty factor is however a delicate task, and there are no recommendations on how to set it.

Similarly, Rashid *et al.* [116] use a MINLP subsolver to solve two auxiliary optimization problems for determining the next sample site(s). In addition to the computationally expensive objective function, Rashid *et al.* [116] consider problems that may have computationally inexpensive as well as expensive constraints.

In this chapter a new surrogate model based algorithm is introduced that solves larger dimensional mixed-integer computationally expensive black-box problems *without* using mixed-integer subsolvers, and where the integer variables may take on a wide range of values rather than only binary values. Using mixed-integer subsolvers may, as for the continuous case, lead to difficulties finding the global optimum of the auxiliary function to be optimized, and the algorithms may thus get trapped in local optima of the response surface which, if the surrogate model is inaccurate, may not be a local optimum of the true objective function. The random sampling strategy proposed here aims at avoiding this difficulty. A theoretical analysis of convergence properties is also presented.

The remainder of this chapter is structured as follows. The mixed-integer problem formulation is given in Section 4.2. The new algorithm, SO-MI, for solving mixed-integer optimization problems that may have computationally expensive constraints is described in Section 4.3, and its theoretical properties are described in Theorem 1. The algorithm uses a radial basis function interpolant (which can in general be replaced by any of the other (mixture) surrogate models described in Chapters 2 and 3). In Section 4.4 the setup of the numerical experiments is described. The performance of SO-MI is compared to a genetic algorithm, a branch and bound algorithm, and NOMAD on 16 test problems from the literature, and five application problems that are briefly described in Section 4.5. A more thorough description of the test problems is given in Appendix C. The numerical results are summarized in Section 4.6, and conclusions are drawn in Section 4.7.

4.2 Mixed-Integer Optimization Problem

In the following let $f : \mathbb{R}^{k_1} \times \mathbb{Z}^{k_2} \mapsto \mathbb{R}$ denote the costly black-box objective function that may in general be nonlinear or multimodal. Denote the decision variables by $\mathbf{w}^T = (\mathbf{x}^T, \mathbf{u}^T)$, where $\mathbf{x} = (x_1, x_2, \ldots, x_{k_1})^T \in \mathbb{R}^{k_1}$ denote the continuous variables, and $\mathbf{u} = (u_1, u_2, \ldots, u_{k_2})^T \in \mathbb{Z}^{k_2}$ denote the discrete variables. The mixed-integer optimization problem can then be stated as

$$\begin{aligned}
\text{minimize} \quad & f(\mathbf{w}) && \text{(4.1a)}\\
\text{s.t.} \quad & c_j(\mathbf{w}) \leq 0, \ \forall j = 1, \ldots, m && \text{(4.1b)}\\
& -\infty < x^l_{i_1} \leq x_{i_1} \leq x^u_{i_1} < \infty, \ \forall i_1 = 1, \ldots, k_1 && \text{(4.1c)}\\
& -\infty < u^l_{i_2} \leq u_{i_2} \leq u^u_{i_2} < \infty, \ \forall i_2 = 1, \ldots, k_2 && \text{(4.1d)}\\
& \mathbf{x} \in \mathbb{R}^{k_1}, \mathbf{u} \in \mathbb{Z}^{k_2}, \mathbf{w}^T = (\mathbf{x}^T, \mathbf{u}^T) && \text{(4.1e)}
\end{aligned}$$

where $x_{i_1}^l$ and $x_{i_1}^u$ denote the lower and upper bounds on the continuous variables x_{i_1}, $i_1 = 1, \ldots, k_1$, respectively, and where $u_{i_2}^l$ and $u_{i_2}^u$ denote the lower and upper bounds on the discrete variables u_{i_2}, $i_2 = 1, \ldots, k_2$, respectively. The jth computationally expensive constraint is denoted by $c_j(\mathbf{w})$. It is assumed that the feasible region is nonempty. Inequalities (4.1c) and (4.1d) are the box constraints. Denote the box-constrained discrete variable domain by $\Omega^D \subset \mathbb{Z}^{k_2}$, and the box-constrained continuous variable domain by $\Omega^C \subset \mathbb{R}^{k_1}$. The mixed-integer box-constrained variable domain is denoted by $\Omega_b = \Omega^C \times \Omega^D$, and it holds that $|\Omega^D| = \kappa < \infty$, i.e. Ω^D is a finite set. The dimension of the mixed-integer optimization problem is denoted by $k = k_1 + k_2$. Throughout this chapter it is assumed that every point $\mathbf{w}$ satisfies the integrality constraints imposed on $\mathbf{u}$, and that $f(\mathbf{w})$ and $c_j(\mathbf{w}), j = 1, \ldots, m$, are continuous when the discrete variables are fixed. Moreover it is assumed that each integer variable can assume at least two values, otherwise the variable would be fixed, and the problem dimension would reduce.

The objective functions of the optimization problems considered in this chapter may in general look as illustrated in Figure 4.1 (constraints left out for simplicity). Illustrated are the graphs of a function with one discrete and one continuous variable when the discrete variable is fixed to three different values. As can be seen, the function depends now only on the continuous variable and may be multimodal.

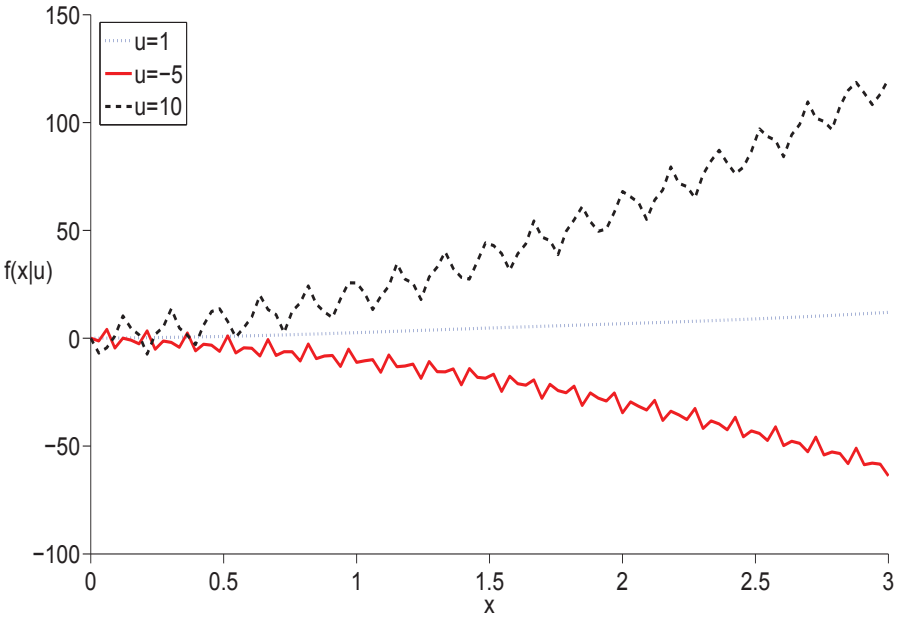


Figure 4.1: Illustrated is the function $f(x, u) = ux + u\sin^3(u^3x) + ux^2$ for three different fixed values of u. The function may be multimodal depending on the discrete variable values.

4.3 SO-MI: Surrogate Model Algorithm for Mixed-Integer Black-Box Optimization Problems

The surrogate model algorithm presented in this chapter has been developed for finding (near) optimal solutions of mixed-integer optimization problems. It will in the following be referred to as SO-MI (**S**urrogate **O**ptimization - **M**ixed-**I**nteger). The algorithm uses a cubic radial basis function (RBF) surrogate model as described in Section 1.2.4. The single steps of SO-MI are described in detail in the following sections.

4.3.1 The Initial Experimental Design and Penalty Functions

The initial experimental design consists of $2k + 1$ points that are generated using a symmetric Latin hypercube design over the box-constrained region where the integer constraints are satisfied by rounding the corresponding variable values to the closest integers. It is assumed that one feasible point is known, and this point is added to the design. In practice, it can be assumed that at least one feasible solution is known due to experience and a general understanding of the problem at hand. Thus, totally $n_0 = 2(k + 1)$ points are in the initial experimental design, which is twice the minimum number of points required to fit the cubic RBF model of dimension k. Note that there is a potential problem of the rank of the matrix $\mathbf{P}$ in equation (1.10) being less than $k+1$ when rounding the integer variables. This has not been a problem in the computational experiments, but, in order to circumvent this problem, the rank of the matrix $\mathbf{P}$ is computed, and if $\text{rank}(\mathbf{P}) < k+1$, a new initial experimental design is generated until $\text{rank}(\mathbf{P}) = k + 1$. Next, the computationally expensive objective function and the constraints are evaluated at these points.

The constraints $c_j(\mathbf{w})$ in equation (4.1b) are integrated during the optimization phase in the objective function by a penalty term. A constraint violation function

$$v(\mathbf{w}) = \sum_{j=1}^{m} \left[\max\left\{0, c_j(\mathbf{w})\right\}\right]^2, \tag{4.2}$$

is used, where it is assumed that $\mathbf{w} \in \Omega_b$, i.e. only points within the box-constrained domain are considered. If $v(\mathbf{w}) = 0$, the point $\mathbf{w}$ satisfies every constraint $c_j(\mathbf{w}), j = 1, \ldots, m$. The penalty for constraint violations is in-

corporated in two different ways. At the beginning of the algorithm when there are possibly only one or very few feasible points known, it is preferable to stay within the feasible region and explore it, i.e. to find further feasible points, and thus improve the accuracy of the response surface within the feasible region. To achieve this goal the following penalty augmented objective function is used

$$f_p(\mathbf{w}) = \begin{cases} f_{\max} + pv(\mathbf{w}), & \text{if } \mathbf{w} \text{ is not feasible} \\ f(\mathbf{w}), & \text{otherwise} \end{cases}, \tag{4.3}$$

where p denotes the penalty factor and $f_{\max}$ is the worst feasible objective function value found so far. This definition guarantees that the penalty augmented function values of the infeasible points are larger than the worst feasible objective function value. Otherwise, if defining the penalty function for example by $f(\mathbf{w}) + pv(\mathbf{w})$, it is possible that for a constant penalty factor p infeasible points reach despite penalty a better objective function value than the feasible points, and the search might be drawn into the infeasible region and many unnecessary objective function evaluations could be done. To prevent numerical instabilities, high function values f_p are replaced by the median. The data $(\mathbf{w}_\iota, f_p(\mathbf{w}_\iota)), \iota = 1, \ldots, n_0$, from the initial experimental design are then used to solve the linear system of equations (1.10) in Section 1.2.4 to obtain the response surface parameters. Note that although some variables have to be integers, all variables are assumed to be continuous in order to obtain a smooth response surface. The sample points are however generated such that the integrality constraints are satisfied.

Since the goal is to find accurate approximations of the global optimum within as few function evaluations as possible, it is desired to do only very few function evaluations at infeasible points. By adding large values to infeasible points as done in equation (4.3) the response surface is likely to predict high objective function values for points in the vicinity of infeasible points. The response surface is used to approximate $f_p(\mathbf{w})$, and the penalty term acts similarly to a barrier method with the goal to stay initially within the feasible region. The disadvantage is however that the accuracy of the response surface near the boundary of the feasible domain is decreased. To overcome this drawback the penalty term is changed after 100 function

evaluations[1].

The penalty augmented objective function used for the remaining function evaluations is defined as follows. First the sum of squared constraint violations (4.2) is scaled to the interval $[0, 1]$:

$$v_s(\mathbf{w}) = \frac{v(\mathbf{w}) - v_{\min}}{v_{\max} - v_{\min}}, \tag{4.4}$$

if $v_{\min} \neq v_{\max}$, and $v_s(\mathbf{w}) = 0$ otherwise (if $v_{\min} = v_{\max}$, then there are no infeasible points). Here $v_{\min} = \min\{v(\mathbf{w}_\iota), \iota = 1, 2, \ldots, n\}$ and $v_{\max} = \max\{v(\mathbf{w}_\iota), \iota = 1, 2, \ldots, n\}$. Thus, feasible points (including the points on the boundary) obtain the value $v_s = 0$, whereas points that violate the constraints obtain positive values for v_s depending on how much these points violate the constraints. The new penalty augmented objective function values are then defined by

$$\tilde{f}_p(\mathbf{w}) = f(\mathbf{w}) + v_s(\mathbf{w}) f_{\max}. \tag{4.5}$$

Thus, the worst feasible objective function value $f_{\max}$ is added to the function value of the point with the largest constraint violation, whereas only a fraction of $f_{\max}$ is added as penalty to the objective function values of points with lower constraint violations. This penalty function does not guarantee that infeasible points have worse objective function values than the best feasible point. The parameters of the response surface are then computed using the data $(\mathbf{w}_\iota, \tilde{f}_p(\mathbf{w}_\iota))$, and the surrogate model may predict better objective function values for infeasible points than for feasible points, and the probability of sampling points at the boundary of the feasible region is larger than when using the definition in equation (4.3).

4.3.2 Selecting the Next Sample Site

Just as for the algorithms for continuous optimization problems described in Chapters 2 and 3, there are several possibilities for determining the next sample site. However, if optimizing an auxiliary function, now a mixed-integer subsolver has to be used. The EGO algorithm by Jones *et al.* [74] could, for example, be adapted by maximizing the expected improvement with respect to the mixed-integer constrains. The Matlab implementation of EGO [48] uses a genetic algorithm for solving the auxiliary optimization problem.

[1]Numerical experiments showed that for the problems in Section 4.5 100 evaluations were sufficient to find several feasible points. The number may be changed depending on the size of the initial experimental design and the total number of allowed function evaluations.

The genetic algorithm in Matlab 2011b and newer Matlab versions is able to deal with integer constraints, and thus this would be a straightforward adaption of EGO for solving mixed-integer problems. Similarly, Gutmann's bumpiness measure [57] could be minimized with respect to the integer constraints with an appropriate mixed-integer subsolver. In this chapter the goal is to develop a random sampling strategy that is able to select several sample points in one iteration, and exploits the possibilities of parallelization.

In every iteration SO-MI chooses four new sample sites for evaluating the expensive objective and constraint functions. These four points are determined by generating four groups of candidate points. The first group of points is generated by perturbing only the continuous variables of the best feasible point found so far (this point is in the following denoted by $\mathbf{w}_{\min}$). The discrete variable values of $\mathbf{w}_{\min}$ are kept constant. Randomly chosen continuous variables of $\mathbf{w}_{\min}$ are perturbed by randomly adding or subtracting small, medium and large perturbations. If the dimension $k > 5$, then each variable is perturbed with a probability of $\max\{0.1, 5/k\}$. Otherwise, every variable is perturbed. Considering high-dimensional problems, this is a practical approach because the goal of a local search is to stay within the vicinity of $\mathbf{w}_{\min}$, and perturbing each variable of, for example, a 30-dimensional problem may cause candidate points to be far away from $\mathbf{w}_{\min}$ [118]. By perturbing only the continuous variables it is possible to find the minimum of the function $f(\mathbf{x}|\mathbf{u}$ fixed), which is at least a local optimum of $f(\mathbf{w})$. The candidate points in this first group are generated such that they satisfy the Global Search Condition 1 in [117, Theorem 1, page 1189].

For generating the candidate points in the second group, the continuous variables of $\mathbf{w}_{\min}$ are kept constant, and only randomly chosen discrete variables are perturbed by randomly adding or subtracting small, medium or large discrete perturbations that depend on the minimum range of the variables, and that are at least one unit. Thus, it is possible to explore how much the objective function values change when transitioning from one integer variable vector to another (compare for example the three objective function values of the three graphs in Figure 4.1 when the continuous variable is kept constant, for example, at $x = 2$). If, for example, the global optima for each function $f(\mathbf{u}|\mathbf{x}$ fixed) are at the same point $\mathbf{x}$ for all $\mathbf{u}$, it is possible to find the global optimum of $f(\mathbf{w})$ very efficiently.

The candidate points in the third group are generated by perturbing randomly chosen discrete *and* continuous variables of $\mathbf{w}_{\min}$ by randomly adding

or subtracting small, medium and large perturbations that are defined as for the first two groups. If any candidate point in the first three groups exceeds the lower or upper variable bounds $x^l_{i_1}, u^l_{i_2}$ or $x^u_{i_2}, u^u_{i_2}$ in equations (4.1c) and (4.1d), respectively, then these variable values are set to the value of the corresponding bound that has been exceeded. The candidate points in the fourth group are generated by uniformly selecting points from the whole box-constrained variable domain Ω_b, and thus every point in Ω_b may become a candidate and therefore also sample point. The integrality constraints are satisfied by rounding the corresponding variables to the closest integers.

The standard deviations of the perturbations of the continuous variables have been set to $h_1 = \delta \cdot \min\{\min\{x^u_{i_1} - x^l_{i_1}, i_1 = 1, \ldots, k_1\}, \min\{u^u_{i_2} - u^l_{i_2}, i_2 = 1, \ldots, k_2\}\}$, where $\delta = 0.1$ for large perturbations, $\delta = 0.01$ for medium perturbations, and $\delta = 0.001$ for small perturbations. The standard deviations of the perturbations for the integer variables have been defined as $h_2 = \max\{1, [h_1]\}$, where $[h_1]$ is the value of h_1 rounded to the closest integer. The random perturbations ζh_1 and ζh_2, where $\zeta \sim \mathcal{N}(0, 1)$, are then added to the variables chosen to be perturbed, and integer variables are subsequently rounded to the closest integer value. Three standard deviations for the perturbations have been used in order to increase the variability of the generated candidate points.

Each group contains $500k$ candidate points, which is sufficient to create a large diversity of points. The four groups have been generated with the incentive of obtaining points that are close to $\mathbf{w}_{\min}$ (local search), and points that are randomly selected from the variable domain (global search). Thus, it is possible to explore the vicinity of $\mathbf{w}_{\min}$ more thoroughly, as well as search globally for other promising regions of the variable domain.

If it is a priori known that the optimization problem is unimodal and that the location of the minimum is independent of the discrete variable values, it might be sufficient to use only candidate points from the third group, and later switching to using candidate points only from the first group to further refine the search since the perturbations are in general much smaller (perturbing an integer variable results in a perturbation of at least one unit). In general, it is, however, unknown how many local and/or global optima are present. The candidate points in the fourth group allow the exploration of the whole variable domain, and thus it is possible to escape from local optima.

The two criteria described in Section 3.3 are used to determine the best candidate point in each group, i.e. one point from each of the four groups

is chosen for doing the next expensive objective and constraint function evaluations[2]. It is assumed that objective and constraint function values are the output of the same computationally expensive black-box simulation model, i.e. whenever the objective function is evaluated, also the constraint function values are obtained. The evaluations at the chosen points can thus be done in parallel because they are independent of each other. Of course, the parallelization can be extended to more than four points if one wishes to select more than four candidate points for doing function evaluations[3].

After all new function values have been obtained, the new points are added to the set of already sampled points $\mathcal{S}$, $f_{\max}$ is updated if necessary, and, depending on the stage of the algorithm, either f_p in equation (4.3) or $\tilde{f}_p$ in equation (4.5) is computed for each point in $\mathcal{S}$. The parameters of the RBF response surface in equation (1.9) are updated using either the data $(\mathbf{w}_\iota, f_p(\mathbf{w}_\iota))$ or $(\mathbf{w}_\iota, \tilde{f}_p(\mathbf{w}_\iota))$, $\iota = 1, \ldots, n$ (depending on the stage of the algorithm), and the algorithm iterates through generating candidate points, calculating scores, and updating the response surface until a given maximal number of function evaluations has been reached.

4.3.3 SO-MI Algorithm

The specific steps of the surrogate model algorithm SO-MI for computationally expensive mixed-integer black-box optimization problems is given below.

Algorithm 5 ***SO-MI: Surrogate Optimization - Mixed Integer***

1. *Repeatedly generate an initial experimental Latin hypercube design with* $2k + 1$ *distinct points, round the discrete variables to the closest integers, and add a known feasible point to the design until* $\text{rank}(\mathbf{P}) = k + 1$*, where* $\mathbf{P}$ *is defined in equation (1.11). Denote the points by* $\mathbf{w}_1, \ldots, \mathbf{w}_{n_0}$*, where* $n_0 = 2(k + 1)$*.*

2. *Do the costly function evaluations to obtain* $f(\mathbf{w}_\iota)$*, and* $c_j(\mathbf{w}_\iota), \iota = 1, \ldots, n_0, j = 1 \ldots, m$*.*

[2]If two or more selected points are the same, then only one of them is used for doing the expensive function evaluation, i.e. less than four points may be selected.

[3]If objective and constraint function values are the output of separate black-box simulation models, it might be favorable (depending on the number of constraints and their computational demand) to use $m+1$ processors, and do objective and constraint function evaluations for one point at a time in parallel rather than using one processor for each point.

3. *Find the best feasible point* $\mathbf{w}_{\min} = \arg\min\{f(\mathbf{w}_\iota)$ *where* $c_j(\mathbf{w}_\iota) \leq 0, \forall j = 1\ldots,m\}$ *with lowest function value* $f_{\min}$*, and determine the worst feasible objective function value* $f_{\max} = \max\{f(\mathbf{w}_\iota)$ *where* $c_j(\mathbf{w}_\iota) \leq 0, \forall j = 1\ldots,m\}$.

4. *Compute* $f_p(\mathbf{w}_\iota), \iota = 1,\ldots,n_0$*, the adjusted objective function values according to equation (4.3).*

5. *Use the data* $(\mathbf{w}_\iota, f_p(\mathbf{w}_\iota)), \iota = 1,\ldots,n_0$*, to calculate the RBF model parameters by solving system (1.10).*

6. *Iterate until the maximal number of allowed function evaluations has been reached:*

   (a) *Create four groups of candidate points by randomly (i) perturbing only continuous variable values of* $\mathbf{w}_{\min}$*, (ii) perturbing only discrete variable values of* $\mathbf{w}_{\min}$*, (iii) perturbing continuous and discrete variable values of* $\mathbf{w}_{\min}$*, and (iv) uniformly selecting points from* Ω_b*, and make sure that the integrality constraints are satisfied. Eliminate all candidates that are already contained in* $\mathcal{S}$ *from further consideration.*

   (b) *Calculate the scoring criteria for every candidate point in each group.*

      - *Predict the objective function value of each candidate point* $\boldsymbol{\chi}_\jmath, \jmath = 1,\ldots,t$*, using the response surface* s_b *defined in equation (1.9), and compute* $V_R(\boldsymbol{\chi}_\jmath)$ *in equation (3.7), where* s_{mix} *must be replaced by* s_b*.*
      - *Determine the distance of each candidate point* $\boldsymbol{\chi}_\jmath, \jmath = 1,\ldots,t$*, to* $\mathcal{S}$*, and compute* $V_D(\boldsymbol{\chi}_\jmath)$ *in equation (3.6).*
      - *Compute the weighted score* $V(\boldsymbol{\chi}_\jmath)$ *in equation (3.8) for the candidates in each group.*

   (c) *Choose from each group the candidate point with the best score* V*.*

   (d) *Do the expensive function evaluations at these points (in parallel).*

   (e) *If necessary, update the best feasible point found so far* $\mathbf{w}_{\min}$*. Update also the worst feasible objective function value* $f_{\max}$ *if necessary, and adjust the objective function values according to equation (4.3) or equation (4.5), depending on the stage of the algorithm.*

   (f) *Update the RBF model parameters using the penalty augmented objective function values from Step 6(e).*

7. Return the best feasible solution found $\mathbf{w}_{\min}$.

4.3.4 Convergence of SO-MI

The algorithm SO-MI is convergent, specifically asymptotically complete [104]. It is assumed that f is a deterministic real-valued function on the compact set $\Omega \subseteq \Omega_b$ defined by the box constraints, and the linear and nonlinear constraints $c_j, j = 1, \ldots, m$, if they exist. It is assumed that f and $c_j, j = 1, \ldots, m$, are continuous when the integer variables are fixed. Denote $f^* = \inf_{\mathbf{w}\in\Omega} f(\mathbf{w}) > -\infty$ the feasible global minimum. Let $\mathbf{w}^*$ be a feasible global minimizer of f over Ω and suppose that f is continuous at $\mathbf{w}^*$ when the integer variables are fixed. Moreover, the candidate points in the first group (perturbation of only continuous variables) are generated such that the Global Search Condition 1 in [117, Theorem 1, page 1189] is satisfied.

Theorem 1 *The algorithm SO-MI is asymptotically complete, i.e. assuming an indefinitely long run-time and exact computations, a global minimum of the optimization problem (4.1a)-(4.1e) will be found with probability one.*

Proof 1 *By definition of the mixed-integer problem, the cardinality* $|\Omega^D| = \kappa$ *is finite, i.e. there is only a finite number of integer variable vectors* $\mathbf{U}^1, \mathbf{U}^2, \ldots, \mathbf{U}^\kappa$. *Therefore, there is only a finite number of possible integer vectors that can be realized by the random vectors* $\mathbf{W}^T = (\mathbf{X}^T, \mathbf{U}^T)$. *If each integer vector is considered as a "bin", then each random vector* $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_n$ *belongs to exactly one such bin. For example, if a problem has only one integer variable* U *with possible values* $U \in \{1, 2, \ldots, 10\}$, *there are totally 10 such bins, and* $|\Omega^D| = 10$.

Each function evaluation point $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_n$ *can thus be assigned to the bin it belongs to according to its integer variable values. Denote the points in each bin as follows.*

$$\textit{Bin 1:}\ \left\{\mathbf{W}_{1(\mathbf{U}^1)}, \mathbf{W}_{2(\mathbf{U}^1)}, \ldots, \mathbf{W}_{n_{\mathbf{U}^1}(\mathbf{U}^1)}\right\}, \tag{4.6}$$

$$\textit{Bin 2:}\ \left\{\mathbf{W}_{1(\mathbf{U}^2)}, \mathbf{W}_{2(\mathbf{U}^2)}, \ldots, \mathbf{W}_{n_{\mathbf{U}^2}(\mathbf{U}^2)}\right\}, \tag{4.7}$$

$$\vdots$$

$$\textit{Bin } \kappa\textit{:}\ \left\{\mathbf{W}_{1(\mathbf{U}^\kappa)}, \mathbf{W}_{2(\mathbf{U}^\kappa)}, \ldots, \mathbf{W}_{n_{\mathbf{U}^\kappa}(\mathbf{U}^\kappa)}\right\}, \tag{4.8}$$

where $\mathbf{W}_{\tilde{i}(\mathbf{U}^{\tilde{j}})} = (\mathbf{X}_{\tilde{i}(\mathbf{U})}, \mathbf{U})$ *with* $\mathbf{U} = \mathbf{U}^{\tilde{j}}$, *and* $\mathbf{X}_{\tilde{i}}$ *is the* $\tilde{i}$*th point for which* $\mathbf{U} = \mathbf{U}^{\tilde{j}}$, *i.e. the* $\tilde{i}$*th point in bin* $\mathbf{U}^{\tilde{j}}$. *For each such bin a subsequence of best*

feasible points found so far can be defined. It is assumed that in the following the points $\mathbf{W}_{\tilde{i}(\mathbf{U}^{\tilde{j}})}, \tilde{i} = 1, 2, \ldots, n_{\mathbf{U}^{\tilde{j}}}$, $\tilde{j} = 1, 2, \ldots, \kappa'$, $\kappa' \leq \kappa$, *satisfy the linear and nonlinear constraints if applicable (there may be integer sequences for which the constraints cannot be satisfied regardless of the continuous variable values). Thus, in general there are at most* $\kappa' \leq \kappa$ *such subsequences of best feasible points found so far, where the* $\tilde{i}$*th subsequence with integer variable values* $\mathbf{U}^{\tilde{i}}$ *is denoted as*

$$\mathbf{W}^*_{1(\mathbf{U}^{\tilde{i}})} = (\mathbf{X}^*_{1(\mathbf{U}^{\tilde{i}})}, \mathbf{U}^{\tilde{i}}) = \mathbf{W}_{1(\mathbf{U}^{\tilde{i}})} = (\mathbf{X}_{1(\mathbf{U}^{\tilde{i}})}, \mathbf{U}^{\tilde{i}}) \tag{4.9a}$$

$$\mathbf{W}^*_{n_{\mathbf{U}^{\tilde{i}}}(\mathbf{U}^{\tilde{i}})} = (\mathbf{X}^*_{n(\mathbf{U}^{\tilde{i}})}, \mathbf{U}^{\tilde{i}}) = \begin{cases} \mathbf{W}_{n_{\mathbf{U}^{\tilde{i}}}(\mathbf{U}^{\tilde{i}})} & \text{if } f(\mathbf{W}_{n_{\mathbf{U}^{\tilde{i}}}(\mathbf{U}^{\tilde{i}})}) < f(\mathbf{W}^*_{n_{\mathbf{U}^{\tilde{i}}-1}(\mathbf{U}^{\tilde{i}})}) \\ \mathbf{W}^*_{n_{\mathbf{U}^{\tilde{i}}}-1(\mathbf{U}^{\tilde{i}})} & \text{otherwise} \end{cases}, \tag{4.9b}$$

for $\tilde{i} = 1, 2, \ldots, \kappa'$.

According to the sampling procedure of SO-MI, it is possible to generate points in each of the κ *bins (recall that candidate point groups 2, 3, and 4 perturb the integer variables/uniformly generate candidate points from the whole variable domain). By assumption, the objective function* f *is continuous when the integer variables are fixed. If SO-MI is able to converge to the feasible global minimum for each continuous subproblem that arises when fixing the integer variable values (i.e. the global minimum of each bin), a finite number (assuming there is a finite number of global minima in each bin) of local minima for the mixed-integer problem can be found. The point amongst these local minima with the lowest objective function value is then the global minimum of the mixed-integer problem.*

Denote $\mathbf{X}^*_{U^{\tilde{i}}}, \tilde{i} = 1, 2, \ldots, \kappa'$, *the feasible global minimizer of the objective function* f *when the integer variables are fixed to the values* $\mathbf{U}^{\tilde{i}}, \tilde{i} = 1, 2, \ldots, \kappa'$, *i.e. the feasible global minimizer of the* $\tilde{i}$*th bin. Denote the feasible global minimum of the continuous objective function by* $f^*_{U^{\tilde{i}}} = f(\mathbf{X}^*_{U^{\tilde{i}}}|\mathbf{U}^{\tilde{i}})$, $\tilde{i} = 1, 2, \ldots, \kappa'$, *where* $(\cdot|\mathbf{U}^{\tilde{i}})$ *means that the integer variables are fixed to* $\mathbf{U}^{\tilde{i}}$. *Then the global minimum of the mixed-integer problem is* $f^* = \min\{f^*_{U^1}, f^*_{U^2}, \ldots, f^*_{U^{\kappa'}}\}$. *Therefore, it remains to show that, assuming indefinitely long run-time and exact computations, the algorithm SO-MI converges to the feasible global minimum for each bin* $\tilde{i} = 1, 2, \ldots, \kappa'$.

Note that it is not necessary to first find the global minimum of some bin before moving on to finding the global minimum of some other bin. According to

the sampling procedure of SO-MI it is possible to return to a bin where sample points have been taken already in previous iterations (candidate points in groups 2, 3, and 4). Thus, it is only necessary to show that SO-MI converges to the global minimum of each bin, which (when considering sample points in the $\tilde{i}$th bin $\mathbf{U}^{\tilde{i}}$ only) is a continuous global optimization problem. This requires showing that the subsequence defined in equations (4.9a) and (4.9b) converges to the minimum $\mathbf{X}^*_{U^{\tilde{i}}}$ of the $\tilde{i}$th bin. However, this proof directly follows along the lines of the proof of Theorem 1 by Regis [117].

4.4 Numerical Experiments

The SO-MI algorithm has been implemented and tested in Matlab 2010a. The performance of SO-MI is in the following analysis compared in numerical experiments to a branch and bound algorithm for solving nonlinear mixed-integer optimization problems, a genetic algorithm, and the mixed-integer option of the C++ implementation of NOMAD [3, 10].

The branch and bound algorithm for nonlinear problems is based on the implementation by Kuipers [78] and applies the trust-region-reflective algorithm (Matlab optimization toolbox function fmincon) when solving the relaxed subproblems in the tree nodes. The maximum number of function evaluations to solve one subproblem has been kept at the default value $100k$. The branch and bound implementation uses a depth-first search with backtracking. The branching variable is chosen such that $|f(x_1,\ldots,x_{k_1},u_1,\ldots,u_{i_2},\ldots,u_{k_2}) - f(x_1,\ldots,x_{k_1},u_1,\ldots,[u_{i_2}],\ldots,u_{k_2})|$ is maximized over all $i_2 \in \{1,\ldots,k_2\}$, where $[u_{i_2}]$ is the value of the i_2th discrete variable rounded to its closest integer.

The branch and bound algorithm has been included in the comparison because it is a widely used algorithm for solving mixed-integer optimization problems. However, as argued before, the performance of branch and bound cannot be expected to be good on multimodal problems due to the computation of the lower bounds in the tree nodes that would require a global optimization algorithm. The comparison with branch and bound is included to show that although the algorithm is suitable for solving mixed-integer problems with special characteristics as, for example, convexity, it is not a feasible option for finding (near) optimal solutions to black-box problems where the algebraic structure of the objective function is unknown and can thus not be exploited.

The genetic algorithm has a population of 20 individuals, 20 generations, and, following [61], uses real-coded chromosomes. The parents for creating the next generation's individuals are chosen based on their objective function value f_p. Crossover and mutation operations have been applied for generating the offspring. In the crossover operation two parents are chosen and the crossover point is selected randomly. The mutation operator randomly selects variables of the parent and adds a value $d\zeta$, where ζ is a random variable drawn from the normal distribution $\mathcal{N}(0, 1)$, and $d = 1$ initially and increases if the mutation does not result in new offspring. The discrete variables are rounded to the closest integer values, and variables exceeding any upper or lower bounds are replaced by the respective value of the bound that has been exceeded.

As suggested by the literature [135, 162] and in order to help the genetic algorithm to perform well within a limited number of function evaluations, a dynamic adjustment of the crossover and mutation probabilities is applied. The probability of using crossover decreases as the number of generations increases, whereas the probability of using mutation increases with the generation number. Hence, a transition from a rather global to a more local search can be achieved as the algorithm advances. The linear and nonlinear constraints have been incorporated in the objective function with a penalty term. The branch and bound algorithm and the genetic algorithm were implemented and tested with Matlab 2011b[4].

NOMAD 3.5.0 has been obtained from [3] and can solve mixed-integer problems [10]. The C++ implementation has been used because it incorporates the variable neighborhood search (the setting VNS_SEARCH 0.75 as suggested in the user manual has been used) that enables the algorithm to escape from local optima. The constraints are treated with the progressive barrier approach (setting PB). Although this version of the user manual states that NOMAD can be used with a surrogate model, the software does not include an implementation of a surrogate model. Therefore, NOMAD has not been used with surrogate models in the numerical experiments.

In the SO-MI algorithm, the penalty factor in equation (4.3) has been set to $p = 100$. The factor guarantees that infeasible points will have worse function values than the worst feasible point. By subsequently

[4]Note that the genetic algorithm in Matlab 2011b and newer versions is able to deal with integer constraints. This algorithm has been tried on the test problems, but delivered significantly worse results than the suggested implementation.

replacing large function values with the median, numerical instabilities can be prevented.

The maximum number of allowed function evaluations has been set to 300 since SO-MI is intended for problems for which only a limited number of function evaluations can be done, and in many real life applications even 100 function evaluations are already computationally infeasible. 30 trials have been made with every algorithm for every problem, and every algorithm was given the same feasible point for the same trial of a given test problem. This feasible point was either the starting point for the algorithm (NOMAD and branch and bound), or it was added to the initial experimental design/initial generation (SO-MI and genetic algorithm).

The mixed-integer global optimization solver *arbfMIP* contained in the commercial TOMLAB optimization environment [64] has not been included in the comparison for the following reasons. TOMLAB has been developed for unconstrained problems and problems that have computationally cheap constraints. If the constraints are computationally expensive, they have to be incorporated in the objective function with a penalty term. There are, however, no recommendations on how to adjust the penalty factor. Moreover, the objective and constraint function values must be computed in separate Matlab files, and thus there are difficulties applying the penalty method used in SO-MI as described in Section 4.3.1. A conclusive comparison between SO-MI and TOMLAB for constrained problems is therefore not possible.

Furthermore, *arbfMIP* has several parameters that need to be adjusted (for example, global search type, cycle length, global and local solver, etc.), which may have to be adjusted according to the test problem, and which is without further knowledge of the problem at hand difficult to do such that the comparison with the other algorithms would be fair.

The mixed-integer global optimization solver *arbfMIP* has been applied to a five- and a 30-dimensional unconstrained test problem with computationally cheap objective functions. The calculations for the five-dimensional problem have been interrupted after more than two hours computation time and the algorithm was not close to having done 300 computationally cheap function evaluations. For the 30-dimensional problem, the algorithm required more than two hours for finding only 50 points for doing the computationally cheap function evaluations, i.e. the *arbfMIP* tends to become itself a computational burden (SO-MI needs less than a tenth of that time for 300 evaluations).

Assuming an allowed maximum number of 300 function evaluations, where each evaluation would take about two minutes, the total time for solving the problem with TOMLAB would almost double for the 30-dimensional problem. Thus, the TOMLAB optimization environment seems to be efficient only for problems where the function evaluation requires considerably more time. Also the solver *glcSolve* could be used, which is significantly faster than *arbfMIP*. For *glcSolve* it is however not possible for the user to define a starting point (the given feasible point). Thus, also for this solver a conclusive comparison of the algorithms would not be possible. For the reasons stated above, TOMLAB has not been included in the computational experiments.

4.5 Test Problems

4.5.1 Generic Test Problems

Totally, 16 test problems (which are modifications of literature problems) have been used to examine the efficiency and solution quality of SO-MI compared to branch and bound, the genetic algorithm, and NOMAD. Four test problems used by Koziel and Michalewicz [77] (test problems 5-8, Table 4.1), that are originally continuous global optimization test problems, have been used and integrality constraints have been imposed on some of the continuous variables.

Four test problems from the Mixed-Integer Nonlinear Programming models library MINLPLib [23] (test problems 3, 9, 11, 12, Table 4.1), five problems that have only box constraints (test problems 2, 10, 13, 14, 15, Table 4.1), and three test problems from [6] have been used to compare the algorithms (see also [17, 47, 159], integer variables are binary variables in these cases, test problems 1, 4, 16, Table 4.1). The mathematical description of all test problems is given in Appendix C together with their best known solutions. Although some of the problems have convex objective functions, the constraints determine the number of local optima. Many problems have therefore several local optima. Furthermore, some problems have flat regions, i.e. regions where several points of the variable domain have the same function value. These generic test problems are computationally inexpensive, but they have characteristics often found in real world application problems, and are thus suitable for comparing the algorithms.

4.5.2 Structural Design Applications

Two applications from structural design have been examined. Mixed-integer optimization problems are often encountered in this application area, for example when designing truss structures.

The goal is in general to minimize structural costs such as the total volume of the structure while satisfying constraints such as limits on the maximal nodal displacements when loads are applied. A finite element analysis has to be done to determine these nodal displacements, and depending on the number of elements involved and the type of finite element analysis required, the computation times may become a considerable burden [36, 72, 115, 126, 136].

The need for integer decision variables arises because technological requirements do not allow the production of truss members with arbitrary cross sectional areas. Rather, catalogues are used from which commercially available member sizes may be chosen. Therefore, discrete variables enter the design problem. On the other hand, as the length of the truss element is variable (one can always weld truss members together), continuous variables are encountered. Although for this problem type the integer constraints could be relaxed during the optimization, there are application problems which do not allow continuous values for integer variables because this may cause the simulation function to fail (see Chapter 5). Therefore, it is assumed that integer variables cannot be relaxed in general.

The two structures shown in Figures 4.2 and 4.3, respectively, are considered. The first example (Figure 4.2, test problem 17, Table 4.1) is a plane truss and is an alteration of the example in [25, Chapter 4]. The structure has 11 elements (an element is a truss member located between two nodes). There are three load cases, $F_1 = 280\,\mathrm{kN}$ acting on nodes 1 and 5, $F_2 = 210\,\mathrm{kN}$ acting on node 3, and $F_3 = 310\,\mathrm{kN}$ acting on node 7. The coordinates of nodes 1, 3, 5 and 7 are fixed, but the height x of the structure is variable. The goal is to minimize the total mass of the structure while satisfying the condition that the maximal displacement is at most 8 mm for all nodes. Therefore, the cross sectional areas of all truss elements have to be determined, as well as the height x of nodes 2, 4 and 6, where each node may have a different value for x. It is assumed that the steel bars have cross-sections of equal-sided angle type, and therefore one discrete variable is associated with every truss element, i.e. there are totally 11 discrete variables describing the side lengths of the bars. There are three continuous

variables describing the vertical locations of nodes 2, 4 and 6.

The second example (Figure 4.3, test problem 18, Table 4.1) is a three-dimensional truss dome. The coordinates of nodes 8-13 are fixed, and the height of nodes 1-7 has to be determined (continuous variables). There are two load cases. Load $F_1 = 20\,\mathrm{kN}$ acts vertically on nodes 2-7, and load $F_2 = 40\,\mathrm{kN}$ acts vertically on node 1. The truss consists of 24 tubular elements that all have the same outer diameter and whose inner diameter has to be determined (discrete variables). The goal is to minimize the total mass while limiting the displacement of node 1 to at most 10 mm. The geometry data of both trusses is given in Appendix C.

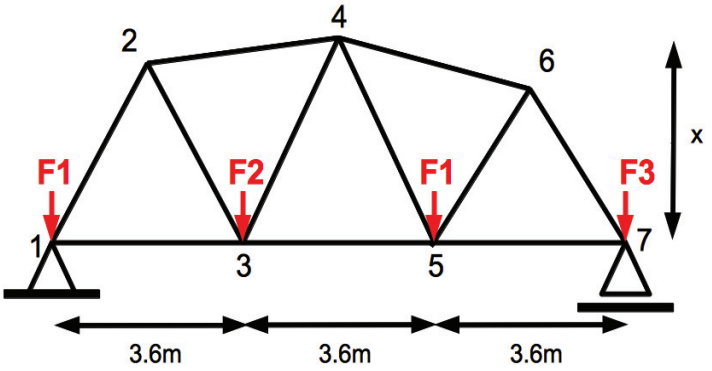
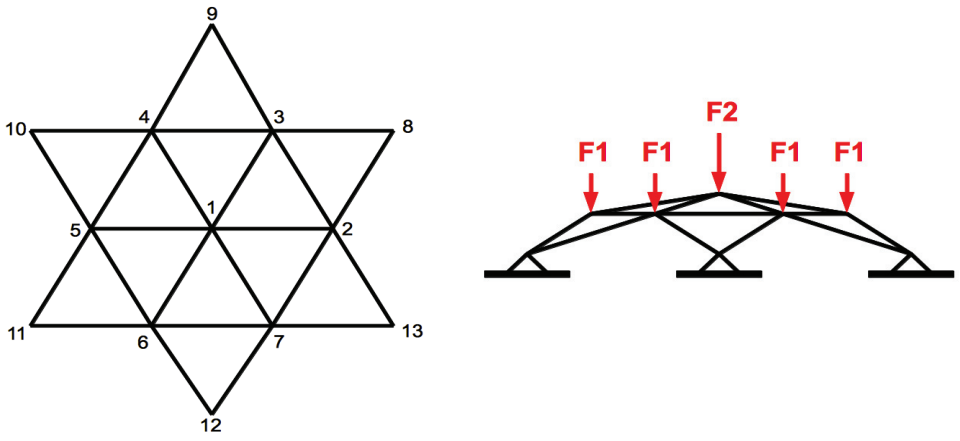


Figure 4.2: Eleven element plane truss (test problem 17, Table 4.1): The goal is to minimize the weight of the structure while satisfying nodal displacement constraints. The geometry data is given in Appendix C.



(a) Top view (b) Side view

Figure 4.3: Truss dome (test problem 18, Table 4.1): The goal is to minimize the weight of the truss dome while satisfying displacement constraints for node 1. The geometry data is given in Appendix C.

4.5.3 Reliability-Redundancy Allocation Problems

Three application problems arising from reliability engineering are examined. Reliability engineering is an important topic in fields such as system, mechanical, electronics, or software engineering. Depending on the system under consideration different levels of reliability must be guaranteed. The consequences of the failure of a system's reliability may vary significantly between different applications (compare, for example, the crash of an airplane and the malfunctioning of a coffee machine). Although the reliability requirements for different systems are in general very different, the common goal is to maximize the total system reliability.

The reliability of a system is defined as the probability that a system or device will perform its intended function for a specified time period under given restrictions such as production costs, for example. The most commonly used system reliability measure is the mean time to failure (MTTF). The higher the MTTF is, the higher the system's reliability.

The engineer has in general two options for increasing the reliability of a system. On the one hand, it is possible to increase the reliability of single components, and on the other hand redundancy can be provided at various stages of the system. Because the component cost often increases exponentially when the component reliability exceeds a certain limit, it

might be cheaper to use components of lower reliability and to provide redundancy, i.e. to add more components of the same kind of lower reliability to the system. Although additional components also incur costs, the cost increase might be less compared to the costs caused by increasing the component reliability. A tradeoff between component reliability increments and component redundancy arises. This problem type is in the literature referred to as reliability-redundancy allocation (see for example [80]).

Since reliability is a probability, it is in practice difficult to test a system's reliability. A single test of the system is in general not representative, and performing multiple tests or tests of systems with a high MTTF may be too expensive. Thus, when adjusting reliability and redundancy parameters for maximizing the total system reliability, it is important to strictly limit the number of reliability tests to a minimum.

Reliability-redundancy allocation leads to a mixed-integer nonlinear optimization problem where the discrete variables represent the levels of redundancy, and the continuous variables are the component reliabilities that are assumed to be known. The general mathematical formulation of such reliability-redundancy allocation problems is as follows [80].

$$\max_{\mathbf{x},\mathbf{u}} \; R_s(\mathbf{x},\mathbf{u}) \tag{4.10a}$$

$$\text{s.t. } c_j(\mathbf{x},\mathbf{u}) \leq b_j, \; j = 1,\ldots,m \tag{4.10b}$$

$$u_{i_2} \in \{u_{i_2}^l,\ldots,u_{i_2}^u\}, \; x_{i_1} \in [x_{i_1}^l, x_{i_1}^u], \; i_1, i_2 = 1,\ldots,k_1, \tag{4.10c}$$

where $R_s(\mathbf{x},\mathbf{u})$ is the system reliability that depends on the system configuration, and the constraints (4.10b) can be, for example, cost constraints or weight restrictions. The continuous variables $\mathbf{x}$ reflect the reliability of each component, whereas the integer variables $\mathbf{u}$ describe the redundancy for each component.

In the literature, different heuristic algorithms [55, 138], as well as algorithms based on branch and bound [79] and evolutionary algorithms [27, 68] have been developed for solving reliability-redundancy allocation problems. However, these methods require in general many function evaluations and may become infeasible in practice.

Three reliability-redundancy allocation problems have in the following been examined, namely a series-parallel configuration, a bridge system, and an overspeed protection system (see [26, 38, 62, 157]). Block diagrams for a simple bridge and a series-parallel configuration, and the configuration

specific data are given in Appendix C.

4.5.4 Overview of Test Problems

A summary of all test problems is shown in Table 4.1. The column "ID" shows the problem identification number each problem is referred to with in the following sections. The column "k_2" denotes the number of discrete variables, and the column "Notes" gives information about the origin of the problem. The column "Characteristics" contains information about the problem constraints, where NLC stands for nonlinear constraints, and LC for linear constraints, and indicates whether the problem is unimodal (UM) or multimodal (MM). FEA means that a finite element analysis has to be done for evaluating the constraints.

4.6 Numerical Results

In the following the genetic algorithm will be referred to as GA, and the branch and bound algorithm for nonlinear problems will be abbreviated by B&B. The test problems have been divided into four groups as follows. The results for the test problems that have only box constraints are discussed in Section 4.6.1. The results of the algorithms for problems that have in addition to the box constraints linear and/or nonlinear constraints are summarized in Section 4.6.2. Section 4.6.3 shows the numerical results for the structural design application problems, and the results of the reliability-redundancy allocation problems are given in Section 4.6.4 (note that the latter are maximization problems).

Each of the Tables 4.2, 4.4, 4.6, and 4.8 shows the problem ID as defined in Table 4.1, and the feasible function value achieved by every algorithm after 100, 200, and 300 function evaluations, respectively, averaged over 30 trials together with the corresponding standard errors of the mean (SEM) in columns "100 eval.", "200 eval.", "300 eval.". The reported numbers are all for feasible points only. Note that if a test problem has constraints, then the constraints are evaluated if and only if the objective function is evaluated, i.e. every objective function evaluation is followed by the evaluation of all constraints, and constraints are not evaluated at a point without the objective function being evaluated. In many real-life application problems

Table 4.1: Summary of test problems; NLC - nonlinear constraints, LC - linear constraints, FEA - finite element analysis, k -problem dimension, k_2 - dimension of integer variables, UM - unimodal, MM - multimodal; ID - problem identification number; see Appendix C for further information
(a) http://www.aridolan.com/ga/gaa/MultiVarMin.html
(b) reliability-redundancy allocation problem.

ID	k	Domain	k_2	Notes	Characteristics
1	11	$[0,1]^8 \times [0,0.997] \times [0,0.9985] \times [0,0.9988]$	4	[17]	3 NLC, 4 LC, MM
2	8	$[-10,10]^8$	4	convex	no constraints, UM
3	5	$[0,10]^3 \times [0,1]^2$	2	[23]	2 NLC, 3 LC, UM
4	3	$[0,1] \times [0.2,1] \times [-2.22554,-1]$	1	[46]	1 NLC, 2 LC, MM
5	25	$[0,10]^{25}$	6	[77]	1 NLC,1 LC, MM
6	5	$[78,102] \times [33,45] \times [27,45]^3$	2	[77]	6 NLC, UM/flat
7	2	$[13,100] \times [0,100]$	1	[77]	2 NLC, MM
8	7	$[-10,10]^7$	3	[77]	4 NLC, MM
9	5	$[0,10]^3 \times [0,1]^2$	3	[23], linear	3 LC, UM
10	5	$[-100,100]^5$	2	(a)	no constraints, MM
11	10	$[3,9]^{10}$	5	[23]	no constraints, UM
12	10	$[3,99]^{10}$	5	[23]	no constraints, UM
13	12	$[-1,3]^{12}$	5	[142]	no constraints, MM
14	12	$[-10,30]^{12}$	5	[142]	no constraints, MM
15	30	$[-1,3]^{30}$	10	[142]	no constraints, MM
16	11	$[0,1]^8 \times [0,10]^3$	4	[159]	4 NLC, 9 LC, MM
17	14	$[10,60]^{11} \times [2000,3200]^3$	11	2-dim. truss	FEA
18	31	$[1,10]^{24} \times [0,1000]^7$	24	3-dim. truss	FEA
19	10	$[1,10]^5 \times [0,0.999999]^5$	5	(b), bridge	3 NLC
20	8	$[1,10]^4 \times [0,0.999999]^4$	4	(b), overspeed	3 NLC
21	10	$[1,10]^5 \times [0,0.999999]^5$	5	(b), series-parallel	3 NLC

objective and constraint function values are the output of the same computationally expensive black-box simulation, and therefore this assumption has been made in the numerical experiments. The columns "dim." and "$|\Omega^D|$" in the tables give information about the problem dimension and the cardinality of the discrete variable domain. The last column indicates whether the problem is unimodal (UM) or multimodal (MM).

It has to be emphasized that the algorithms are compared with respect to the number of function evaluations needed to find improvements, and that the computation times of the algorithms are neglected because in applications where a function evaluation may take up to several hours or even days, the algorithms' computation times become insignificant. The objective function values reported in the tables are for feasible points only. It shall be noted at this point that the performance of NOMAD, GA, and B&B may be improved by combining these algorithms with surrogate models. However, publicly available codes are rather scarce, and as in the case of the C++ implementation of NOMAD, the developers ask the user to implement the surrogate model him/herself.

In order to better compare the solution quality of the algorithms for the four problem classes, the "score" rows of Tables 4.2, 4.4, 4.6, and 4.8 summarize how much each algorithm deviates from the best feasible solution averaged over 30 trials (numbers in %). The value is computed for each column (100 eval., 200 eval., and 300 eval.) and each algorithm as

$$\frac{1}{|\mathcal{N}|}\sum_{\pi\in\mathcal{N}}\left|\frac{f^n_{\mathcal{A}}(\pi)-f^n_{\text{best}}(\pi)}{f^n_{\text{best}}(\pi)}\right|\cdot 100, \tag{4.11}$$

where $\mathcal{N}$ is the set of problems in each category (box-constrained, constrained, structural, reliability-redundancy), π denotes a certain problem within the category, $f^n_{\text{best}}(\pi)$ is the best average feasible objective function value reached for problem π after n function evaluations (n is 100, 200, or 300), and $f^n_{\mathcal{A}}(\pi)$ is the average objective function value reached by algorithm $\mathcal{A}$ ($\mathcal{A}$ is SO-MI, GA, B&B, or NOMAD) after n evaluations for problem π. The best algorithm for a given problem receives therefore the score 0, whereas all other algorithms with worse results obtain a positive score. Thus, by this definition, the smaller the reported score is, the better is the algorithm's performance because the lower is the deviation from the best solution.

In addition, Tables 4.3, 4.5, 4.7, and 4.9 contain information of hypothesis testing for differences in means between SO-MI and the other algorithms after 100, 200, and 300 function evaluations

4.6.1 Box-Constrained Problems

The results for the box-constrained problems in Table 4.2 show that SO-MI and NOMAD outperform B&B and GA on all problems of this group. Comparing the function values averaged over all 30 trials, shows that SO-MI is better than NOMAD for five out of seven problems after 100 function evaluations (column 3), indicating that SO-MI finds improvements faster. SO-MI is also superior after 200 and 300 function evaluations for four and five out of the seven problems, respectively.

B&B did not find any improvements during 300 function evaluations for test problems 11, 12, 13, 14 and 15. GA found improvements for all problems, but could not outperform SO-MI or NOMAD. B&B has in general the worst performance, and is outperformed by *all* other algorithms for five out of seven problems including the unimodal test problems. This fact indicates that the optimization of the subproblems in the tree nodes of the branch and bound algorithm consumes too many function evaluations, and feasible solutions with respect to the integer constraints cannot be found efficiently.

Problems 13 and 14 are essentially the same problems, but problem 14 has a larger variable domain. The results show that for problem 13 the results found by SO-MI and NOMAD after 300 evaluations are about equal, whereas the results of NOMAD for problem 14 are for all evaluations significantly worse. B&B found for problem 13 only very slight improvements of the initially given point within 300 evaluations, and was not able to find any improvements when the variable domain was increased. Also the performance of GA is significantly worse for the problem with larger variable domain.

Similarly, problems 11 and 12 have the same structure, but problem 12 has a larger variable domain. The results show that SO-MI is able to find for both problems a near optimal solution within fewer than 100 function evaluations, whereas NOMAD reaches competitive results after 300 evaluations. B&B did not find any improvements of the initially given points for either problem. GA performs slightly worse for problem 12 (the final solution is about 40% worse than the best solution after 300 evaluations) than for problem 11 (the final solution is about 22% worse than the best solution after 300 evaluations), indicating that the performance of GA decreases when the variable domain gets larger. These four problems (11, 12, 13, and 14) show that the performance of SO-MI is less affected by the size of the

variable domain than the other algorithms.

The scores for each algorithm in the table are computed without taking the convex problem 2 into consideration because SO-MI was developed for non-convex and multimodal problems, and was not expected to work so well on a convex problem. The convex problem was included only to examine how well SO-MI might do on this problem class. The scores in the table show that SO-MI performs better than all other algorithms and has the smallest deviations from the best solution found. Although SO-MI does not continuously perform best on all problems, the cases where it is outperformed by NOMAD show that the differences between the results found by NOMAD and SO-MI are much smaller than the differences for the problems where SO-MI outperforms NOMAD. Compared to GA and B&B, SO-MI and NOMAD perform significantly better.

The standard errors of the means (SEM) given for each algorithm in Table 4.2 are for SO-MI in general lowest (except for test problem 10). The hypothesis testing for differences in means between the different algorithms in Table 4.3 shows that except for problems 10, 13, and 15 the average objective function values of SO-MI are significantly lower than those of GA, NOMAD, and B&B at the 1% significance level. NOMAD reaches significantly lower values than SO-MI only for test problems 10 and 15 after 200 and more function evaluations, and for problem 13 for up to 100 evaluations. The SEM values reported for B&B for test problems 11, 12, 13, 14, and 15 are computed based on the initially given points since B&B was not able to find any improvements for these test problems.

Figures 4.4 and 4.5 illustrate the development of the objective function value averaged over 30 trials versus the number of function evaluations for all four compared algorithms for test problems 12 and 15. The error bars show the standard deviations. Since the problems were about minimization, the best algorithm is the one that achieves the fastest reduction of the objective function value. Figure 4.4 shows that SO-MI improves the objective function value within the lowest number of function evaluations as compared to GA, NOMAD and B&B. NOMAD eventually finds a solution that is very similar (less than 4% difference) to the one found by SO-MI, but comparing the graphs of SO-MI and NOMAD after 100 and 200 function evaluations, for example, shows that SO-MI performs significantly better. GA also finds improvements, but is not able to perform nearly as well as SO-MI. Figure 4.5 illustrates the results found by the algorithms for test problem 15. SO-MI is outperformed by NOMAD after about 100 function evaluations, and both

algorithms outperform GA and B&B. B&B did not find any improvements within 300 function evaluations for either test problem.

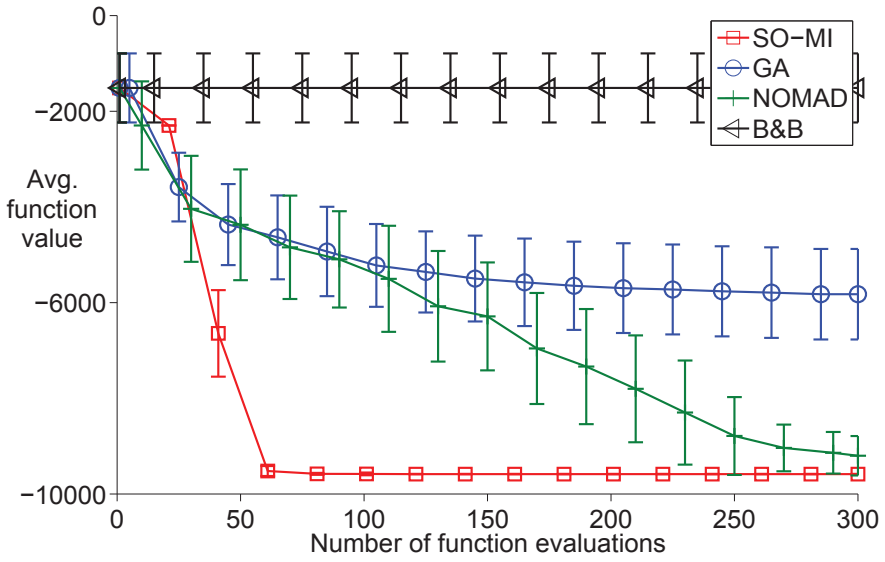


Figure 4.4: Box-constrained minimization problem, test problem 12. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 4.2: Box-constrained problems. Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. MM - multimodal, UM - unimodal.

ID	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	dim.	$\lvert\Omega^D\rvert$	MM/UM
10	SO-MI	mean	-386.33	-420.91	-432.58	5	101^2	MM
		SEM	*16.74*	*14.87*	*15.17*			
	GA	mean	-240.21	-300.64	-306.96			
		SEM	*18.78*	*18.54*	*18.04*			
	B&B	mean	644.33	-120.76	-357.94			
		SEM	*107.72*	*95.97*	*19.42*			
	NOMAD	mean	-380.20	-460.05	-479.98			
		SEM	*14.49*	*12.41*	*9.37*			
11	SO-MI	mean	-42.92	-42.99	-42.99	10	7^5	UM
		SEM	*0.19*	*0.13*	*0.13*			
	GA	mean	-17.00	-26.31	-33.73			
		SEM	*0.78*	*0.76*	*0.77*			
	B&B	mean	4.23	4.23	4.23			
		SEM	*1.08*	*1.08*	*1.08*			
	NOMAD	mean	-26.78	-32.99	-38.26			
		SEM	*1.03*	*1.01*	*0.87*			
12	SO-MI	mean	-9581.32	-9584.62	-9584.62	10	97^5	UM
		SEM	*5.26*	*0.79*	*0.79*			
	GA	mean	-4931.94	-5648.29	-5825.53			
		SEM	*170.55*	*168.37*	*172.75*			
	B&B	mean	-1513.54	-1513.54	-1513.54			
		SEM	*131.72*	*131.72*	*131.72*			
	NOMAD	mean	-5280.95	-7436.47	-9201.87			
		SEM	*184.13*	*222.49*	*75.23*			

13	SO-MI	mean	-4.89	-8.48	–9.63	12	5^5	MM
		SEM	*0.33*	*0.30*	*0.23*			
	GA	mean	2.69	-1.71	-3.74			
		SEM	*0.60*	*0.34*	*0.28*			
	B&B	mean	23.88	23.88	23.88			
		SEM	*2.05*	*2.05*	*2.05*			
	NOMAD	mean	-7.51	-8.66	-9.27			
		SEM	*0.43*	*0.19*	*0.31*			
14	SO-MI	mean	27.95	-5.78	-8.27	12	41^5	MM
		SEM	*2.85*	*0.27*	*0.34*			
	GA	mean	816.53	510.26	376.38			
		SEM	*40.68*	*26.82*	*22.18*			
	B&B	mean	2802.79	2802.79	2802.79			
		SEM	*180.28*	*180.28*	*180.28*			
	NOMAD	mean	23.10	19.88	13.23			
		SEM	*1.94*	*1.13*	*2.31*			
15	SO-MI	mean	-6.59	-10.29	-11.80	30	5^{10}	MM
		SEM	*0.37*	*0.36*	*0.47*			
	GA	mean	25.48	13.83	10.73			
		SEM	*1.46*	*1.09*	*1.07*			
	B&B	mean	62.03	62.03	62.03			
		SEM	*3.64*	*3.64*	*3.64*			
	NOMAD	mean	-6.39	-19.27	-19.52			
		SEM	*2.08*	*0.90*	*0.88*			
Score in %	SO-MI		9	9	8			
	GA		700	1550	827			
	B&B		2325	8282	5830			
	NOMAD		15	81	46			

2 convex	SO-MI	mean	0.00	0.00	0.00	8	21^4	UM
		SEM	*0.00*	*0.00*	*0.00*			
	GA	mean	274.17	153.30	82.79			
		SEM	*21.13*	*11.26*	*8.28*			
	B&B	mean	1471.05	1274.99	4.29			
		SEM	*132.65*	*145.11*	*1.10*			
	NOMAD	mean	117.70	7.36	0.14			
		SEM	*1.23*	*1.68*	*0.05*			

Table 4.3: Box-constrained problems. Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations.
($**$) denotes significance at $\alpha = 1\%$ for
$H_0 : \mu_{\text{SO-MI}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-MI}} < \mu_{\mathcal{A}}$;
($\diamond$) denotes significance at $\alpha = 5\%$ and ($\diamond\diamond$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\text{NOMAD}}$ and $H_1 : \mu_{\text{SO-MI}} > \mu_{\text{NOMAD}}$.

ID	Algorithm	Number of evaluations		
	$\mathcal{A}$	100	200	300
10	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($\diamond$)	($\diamond\diamond$)
11	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)
12	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)
13	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($\diamond\diamond$)		
14	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	($**$)
15	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($\diamond\diamond$)	($\diamond\diamond$)
2	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)

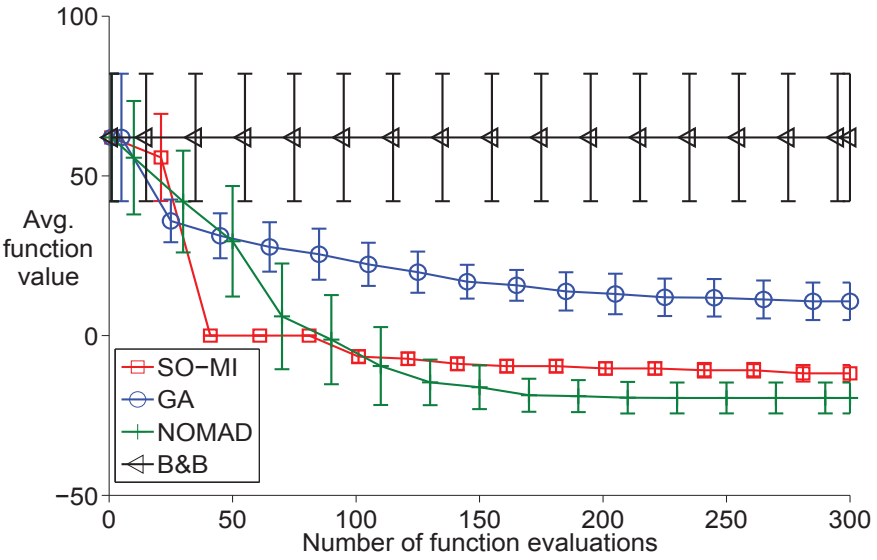


Figure 4.5: Box-constrained minimization problem, test problem 15. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

4.6.2 Constrained Problems

The numerical results for the constrained test problems are shown in Table 4.4. The results show that SO-MI performs best for six out of nine test problems after 100 evaluations, and for five out of nine test problems after 200 and 300 evaluations, respectively. NOMAD on the other hand had the best performance only for one of the nine test problems after 200 function evaluations, and for two of nine test problems after 300 function evaluations. These results indicate that SO-MI performs in general better than NOMAD also for the constrained problems, and that SO-MI is able to find improvements within fewer function evaluations as compared to NOMAD.

B&B found the best results for test problems three, six, and nine, where test problem six is in particular a convex problem, and test problems three and nine have a linear objective function. Test problems three and nine are essentially the same problem, but the nonlinear constraints have been omitted in test problem nine. Moreover, for test problems three and nine the globally optimal point of the continuous relaxation is the same as when integer constraints are imposed (the global optimum of the

continuous problem has all variable values zero), which explains the superior performance of B&B on these problems.

For test problem eight, the optimization of the relaxed subproblems in the tree nodes of B&B did not converge and the algorithm stalled. Thus, the reported result is the average of the function values of the initially given feasible points for the 30 trials. Moreover, B&B was not able to find improvements for the remaining five test problems (test problems 1, 4, 5, 7, and 16) within 300 evaluations, and has for these problems the worst performance among all compared algorithms.

GA was able to improve the initially given solution for all but one test problem, and performs in general better than B&B, but significantly worse than SO-MI and NOMAD. Although GA has been developed for solving multimodal optimization problems, the number of function evaluations needed for finding good solutions is very large due to the number of individuals in the single generations and the number of generations.

Also for this problem class a score has been computed that measures how much each algorithm deviates from the best feasible average result found after 100, 200, and 300 evaluations. Test problems 3, 6, and 9 have not been included in the calculation because problem 6 is convex, and problems 3 and 9 have linear objective functions and the continuous relaxations lead directly to an integer feasible solution. SO-MI has been developed for multimodal problems and therefore these problems are left out when computing the scores. The scores show that SO-MI performs also for this problem class better than any of the other algorithms. NOMAD performs better than GA and B&B, and its performance improves as the number of function evaluations increases.

The hypothesis testing for differences in means in Table 4.5 confirms that SO-MI attains significantly better results (at the level $\alpha = 1\%$) than all other algorithms for most problems. For reasons explained above, B&B performs significantly better on the special problems 3, 6, and 9.

Figure 4.6 illustrates the development of the function value for feasible points averaged over 30 trials versus the number of function evaluations for all algorithms for test problem five. The error bars illustrate the standard deviations for each algorithm. The figure shows that SO-MI is able to find significantly better feasible solutions than B&B and GA, and eventually also NOMAD. It can be seen that all B&B trials got

stuck in the same point, and the algorithm did not find any improvements within 300 function evaluations. NOMAD and GA are able to find slight improvements of the initially supplied feasible solutions, but are not able to find any more improvements after about 150 function evaluations.

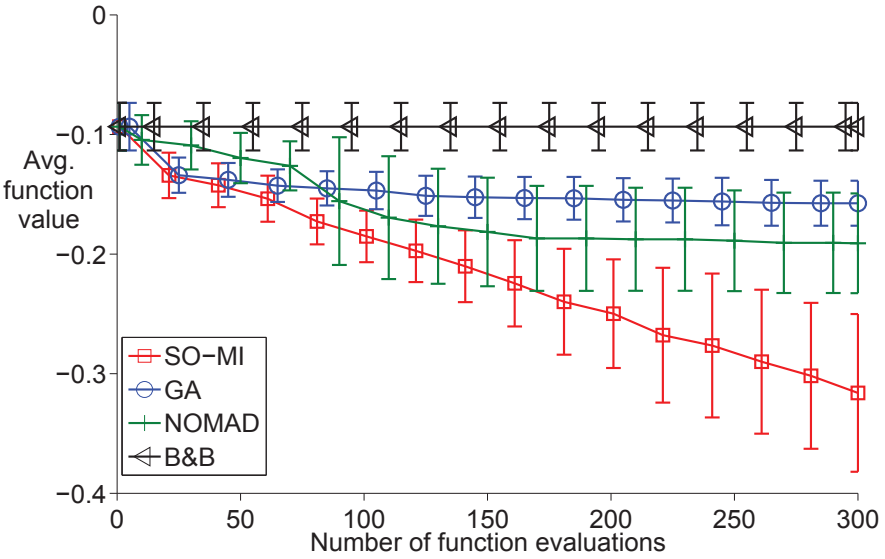


Figure 4.6: Constrained minimization problem, test problem 5. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 4.4: Problems with linear and/or nonlinear constraints. Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. B&B* means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, therefore also SEM is not available. MM - multimodal, UM - unimodal.

ID	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	dim.	$\lvert\Omega^D\rvert$	MM/UM
1	SO-MI	mean	-0.50	-0.66	-0.72	11	2^4	MM
		SEM	*0.03*	*0.02*	*0.02*			
	GA	mean	0.26	-0.31	-0.33			
		SEM	*0.03*	*0.03*	*0.03*			
	B&B	mean	-0.09	-0.09	-0.09			
		SEM	*0.02*	*0.02*	*0.02*			
	NOMAD	mean	-0.47	-0.53	-0.64			
		SEM	*0.02*	*0.02*	*0.02*			
4	SO-MI	mean	2.99	2.95	2.93	3	2	MM
		SEM	*0.02*	*0.02*	*0.01*			
	GA	mean	3.16	3.16	3.16			
		SEM	*0.02*	*0.02*	*0.02*			
	B&B	mean	3.18	3.18	3.18			
		SEM	*0.02*	*0.02*	*0.02*			
	NOMAD	mean	3.03	2.94	2.90			
		SEM	*0.02*	*0.01*	*0.01*			
5	SO-MI	mean	-0.18	-0.25	-0.32	25	11^6	MM
		SEM	*0.00*	*0.01*	*0.01*			
	GA	mean	-0.15	-0.15	-0.16			
		SEM	*0.00*	*0.00*	*0.00*			
	B&B	mean	-0.09	-0.09	-0.09			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	mean	-0.17	-0.19	-0.19			
		SEM	*0.01*	*0.01*	*0.01*			

7	SO-MI	mean	-4156.44	-4182.99	-4186.56	2	14	MM
		SEM	*14.39*	*9.37*	*8.12*			
	GA	mean	-3266.52	-3380.55	-3616.21			
		SEM	*88.78*	*99.30*	*99.71*			
	B&B	mean	-3194.01	-3194.01	-3194.01			
		SEM	*93.41*	*93.41*	*93.41*			
	NOMAD	mean	-3825.44	-3845.79	-3847.36			
		SEM	*48.91*	*47.20*	*47.62*			
8	SO-MI	mean	1057.39	847.65	761.448	7	21^3	MM
		SEM	*17.81*	*15.68*	*8.63*			
	GA	mean	1095045.15	874665.90	629728.43			
		SEM	*446436.77*	*409890.30*	*368763.38*			
	B&B*	mean	1436981.97	1436981.97	1436981.97			
		SEM	*NA*	*NA*	*NA*			
	NOMAD	mean	3830.17	1124.20	835.22			
		SEM	*916.38*	*66.73*	*18.79*			
16	SO-MI	mean	8.05	6.60	6.20	11	2^4	MM/flat
		SEM	*0.22*	*0.10*	*0.06*			
	GA	mean	10.75	10.60	10.10			
		SEM	*0.33*	*0.29*	*0.31*			
	B&B	mean	10.93	10.93	10.93			
		SEM	*0.35*	*0.35*	*0.35*			
	NOMAD	mean	9.09	8.65	7.76			
		SEM	*0.29*	*0.30*	*0.23*			
Score in %	SO-MI		0	0	1			
	GA		17265	15351	12579			
	B&B		22666	25226	28700			
	NOMAD		50	15	11			

3	SO-MI	mean	0.44	0.09	0.04	5	11^2	UM
		SEM	*0.06*	*0.02*	*0.01*			
	GA	mean	3.69	2.36	1.03			
		SEM	*0.34*	*0.34*	*0.24*			
	B&B	mean	0.00	0.00	0.00			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	mean	0.62	0.21	0.01			
		SEM	*0.11*	*0.07*	*0.01*			
6	SO-MI	mean	-29767.98	-29983.69	-30078.49	5	975	UM/flat
		SEM	*67.99*	*53.92*	*51.23*			
	GA	mean	-28993.04	-29285.90	-29461.46			
		SEM	*100.28*	*99.67*	*108.62*			
	B&B	mean	-30665.54	-30665.54	-30665.54			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	mean	-29407.09	-29998.12	-30215.53			
		SEM	*129.71*	*66.70*	*45.52*			
9 linear	SO-MI	mean	0.67	0.35	0.17	5	11^2	UM
		SEM	*0.14*	*0.15*	*0.07*			
	GA	mean	4.33	1.42	0.67			
		SEM	*0.35*	*0.26*	*0.22*			
	B&B	mean	0.00	0.00	0.00			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	mean	0.41	0.16	0.00			
		SEM	*0.09*	*0.06*	*0.00*			

Table 4.5: Box-constrained problems with additional linear and/or nonlinear constraints. Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations.
($*$) denotes significance at $\alpha = 5\%$ and ($**$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-MI}} < \mu_{\mathcal{A}}$;
($\diamond$) denotes significance at $\alpha = 5\%$ and ($\diamond\diamond$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\text{NOMAD}}$ and $H_1 : \mu_{\text{SO-MI}} > \mu_{\text{NOMAD}}$;
($\oplus$) denotes significance at $\alpha = 5\%$ and ($\oplus\oplus$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\text{B\&B}}$ and $H_1 : \mu_{\text{SO-MI}} > \mu_{\text{B\&B}}$.

ID	Algorithm	Number of evaluations		
	$\mathcal{A}$	100	200	300
1	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	($**$)
4	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD			($\diamond$)
5	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	($**$)
7	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)
8	GA	($**$)	($*$)	($*$)
	B&B	NA	NA	NA
	NOMAD	($**$)	($**$)	($**$)
16	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)

3	GA B&B NOMAD	$(**)$ $(\oplus\oplus)$	$(**)$ $(\oplus\oplus)$	$(**)$ $(\oplus\oplus)$
6	GA B&B NOMAD	$(**)$ $(\oplus\oplus)$ $(**)$	$(**)$ $(\oplus\oplus)$	$(**)$ $(\oplus\oplus)$ $(\diamond)$
9	GA B&B NOMAD	$(**)$ $(\oplus\oplus)$	$(**)$ $(\oplus)$	$(*)$ $(\oplus\oplus)$ $(\diamond\diamond)$

4.6.3 Structural Design Problems

The results for the structural design optimization applications are shown in Table 4.6. For both problems SO-MI outperforms all other algorithms and reaches significantly better results within the same number of function evaluations. B&B was not able to find any improvements for the three-dimensional truss dome problem. For the two-dimensional truss the optimization of the relaxed subproblems in the tree nodes of B&B did not converge within the allowed $100k$ function evaluations, and therefore the average of the function values of the 30 different initially provided feasible points is given in the table, and the standard error of the mean is therefore not available. B&B had the worst performance for both structural design problems.

GA was able to improve the initially supplied feasible solutions and outperformed NOMAD on the three-dimensional truss dome application (the 31-dimensional problem), but does worse than NOMAD on the smaller problem 17. The scores at the end of the table reflect again that SO-MI performs better than all other algorithms on this problem class. Averaging over both structural design problems shows that GA outperforms NOMAD.

For this problem class the standard errors of the means are for SO-MI at almost all stages of the algorithm lower than for the other algorithms (except for GA after 100 evaluations for test problem 18). Also the results of the statistical tests in Table 4.7 show that the means of the feasible objective function values attained by SO-MI are significantly lower than the results found by the other algorithms.

The objective function value averaged over all 30 trials versus the number of function evaluations for the three-dimensional truss dome application problem is illustrated for all four algorithms in Figure 4.7. The error bars illustrate the standard deviations of the algorithms. The figure shows that SO-MI finds significantly better solutions than all other algorithms and has the lowest standard deviations. Figure 4.8 illustrates the best design obtained for the 2D truss problem. Thicker lines illustrate truss members with larger cross sectional areas. The arrows indicate the height of nodes 2, 4, and 6. The maximal nodal displacement occurs at node 7 in vertical direction.

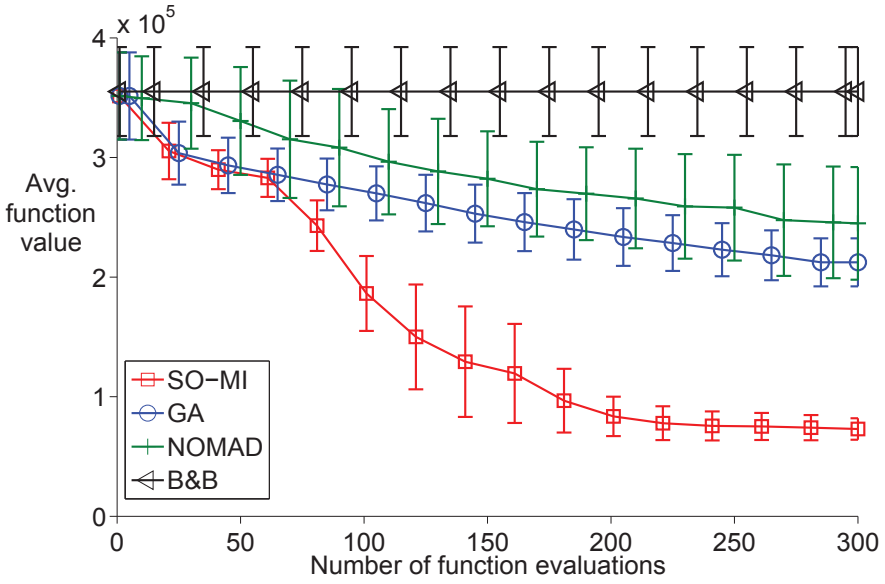


Figure 4.7: Truss dome design optimization, minimization problem, test problem 18. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 4.6: Structural design optimization problems. Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. B&B* means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, and therefore also SEM is not available. Both problems are minimization problems.

ID	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	dim.	$\lvert\Omega^D\rvert$
17 2D truss	SO-MI	mean	102624.50	98099.88	95902.51	14	51^{11}
		SEM	*807.45*	*707.95*	*614.30*		
	GA	mean	112975.58	112924.52	112816.52		
		SEM	*889.47*	*900.90*	*933.85*		
	B&B*	mean	113161.21	113161.21	113161.21		
		SEM	NA	NA	NA		
	NOMAD	mean	103354.72	101433.69	98484.86		
		SEM	*815.42*	*858.34*	*932.62*		
18 3D dome	SO-MI	mean	186281.11	83525.43	73010.86	31	10^{24}
		SEM	*5716.76*	*3007.10*	*1640.95*		
	GA	mean	277476.80	239829.09	212346.21		
		SEM	*3942.84*	*4613.54*	*3666.28*		
	B&B	mean	355160.38	355160.38	355160.38		
		SEM	*6786.15*	*6786.15*	*6786.15*		
	NOMAD	mean	303995.43	268712.00	244911.47		
		SEM	*9193.94*	*7153.09*	*8595.33*		
Score in %	SO-MI		0	0	0		
	GA		30	97	101		
	B&B		50	164	196		
	NOMAD		32	108	115		

Table 4.7: Structural design optimization problems. Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations. ($*$) denotes significance at $\alpha = 5\%$ and ($**$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-MI}} < \mu_{\mathcal{A}}$.

ID	Algorithm	Number of evaluations		
	$\mathcal{A}$	100	200	300
17	GA	(**)	(**)	(**)
	B&B	NA	NA	NA
	NOMAD		(**)	(*)
18	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)

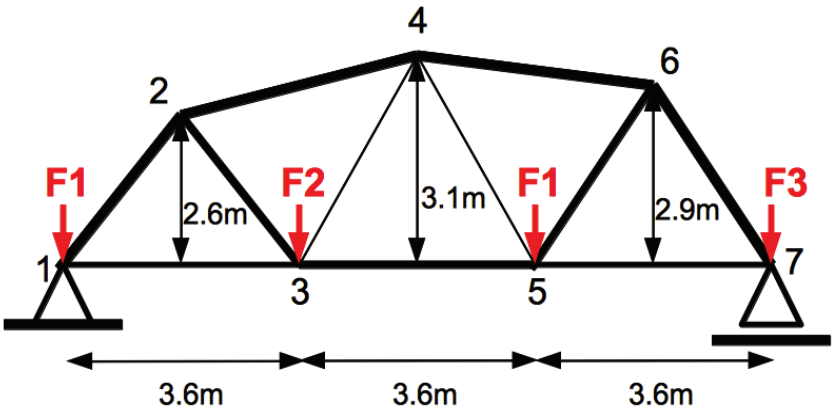


Figure 4.8: Best design of the 2D truss problem. Thicker lines symbolize truss members with larger cross sectional areas. The largest displacement is in vertical direction at node 7. See Appendix C for the problem description and data.

4.6.4 Reliability-Redundancy Problems

Table 4.8 summarizes the results of the four compared algorithms for the reliability-redundancy allocation application problems. The goal is here to *maximize* the reliability of the system, and therefore large numbers are better. Note that in this application area the digits after the decimal point are important and therefore four decimals are reported. The results show that SO-MI performs best for all three problems for over 200 function evaluations, indicating that SO-MI is successful in finding improvements more efficiently than all other algorithms.

NOMAD achieves only marginally better results for problem 20 after 300 function evaluations. B&B was not able to find any feasible improvements of the initially supplied feasible points, and for the series-parallel system (problem 21) the optimization of the relaxed problems in the tree nodes did not converge. The reported numbers for B&B for this problem are the average of the objective function values of the initially supplied feasible points. GA found slight improvements for all three test problems, but was in general not as good as SO-MI or NOMAD. This fact is also reflected by the scores at the end of the table. SO-MI performs best on this problem class, and the performance of NOMAD improves as the number of function evaluations increases.

The standard errors of the means are for all algorithms very low, and SO-MI and NOMAD have the lowest values. The statistical test results for differences of means are reported in Table 4.9, and the results show that the mean of SO-MI is for all problems at almost all stages of the algorithm (100, 200, and 300 evaluations) significantly higher (maximization problem) than for the other algorithms at levels $\alpha = 1\%$ and $\alpha = 5\%$, respectively.

Figure 4.9 shows the objective function value for feasible points averaged over 30 trials versus the number of function evaluations for all algorithms for the series-parallel system (test problem 21). Note that for the maximization problem the algorithm that increases the average feasible objective function value fastest is best. The error bars illustrate the standard deviations. The figure shows that NOMAD finds immediately improvements of the given feasible point because it uses only one point from which the mesh adaptive direct search starts. SO-MI on the other hand generates at first an initial experimental design and starts the "systematic" search for improvements after all points in the starting design have been evaluated. This is reflected by the initially constant average feasible objective function value of SO-MI

(the value is constant until evaluation $2(k+1) = 22$ because no other feasible points were in the initial experimental design). After the points in the initial experimental design have been evaluated, SO-MI immediately finds better feasible results than all other algorithms. Figure 4.10 shows the best reliability-redundancy allocation for the bridge system (test problem 19). As can be seen, redundancy has been added at various stages.

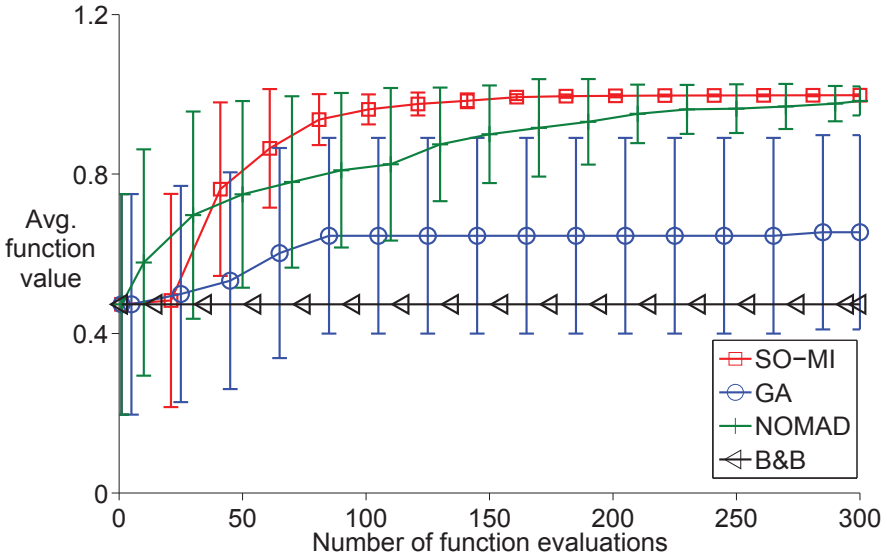


Figure 4.9: Reliability-redundancy allocation for series-parallel configuration, maximization problem, test problem 21. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 4.8: Reliability-redundancy allocation problems (maximization problems). Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. B&B* means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, and therefore also SEM is not available.

ID	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	dim.	$\lvert\Omega^D\rvert$
19 bridge system	SO-MI	mean	0.9843	0.9974	0.9982	10	5^{10}
		SEM	*0.00*	*0.00*	*0.00*		
	GA	mean	0.8211	0.8224	0.8369		
		SEM	*0.03*	*0.03*	*0.02*		
	B&B	mean	0.7573	0.7573	0.7573		
		SEM	*0.03*	*0.03*	*0.03*		
	NOMAD	mean	0.9640	0.9849	0.9950		
		SEM	*0.01*	*0.00*	*0.00*		
20 over-speed	SO-MI	mean	0.9939	0.9988	0.9991	8	4^{10}
		SEM	*0.00*	*0.00*	*0.00*		
	GA	mean	0.9588	0.9637	0.9646		
		SEM	*0.01*	*0.00*	*0.00*		
	B&B	mean	0.8060	0.8060	0.8060		
		SEM	*0.03*	*0.03*	*0.03*		
	NOMAD	mean	0.9884	0.9971	0.9992		
		SEM	*0.00*	*0.00*	*0.00*		
21 series parallel	SO-MI	mean	0.9618	0.9962	0.9977	10	5^{10}
		SEM	*0.01*	*0.00*	*0.00*		
	GA	mean	0.6452	0.6452	0.6539		
		SEM	*0.04*	*0.04*	*0.04*		
	B&B*	mean	0.4731	0.4731	0.4731		
		SEM	*NA*	*NA*	*NA*		
	NOMAD	mean	0.8132	0.9388	0.9833		
		SEM	*0.04*	*0.01*	*0.01*		
Score in %	SO-MI		0	0	0		
	GA		17	18	18		
	B&B*		31	32	32		
	NOMAD		6	2	0		

Table 4.9: Reliability-redundancy allocation problems. Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations. ($*$) denotes significance at $\alpha = 5\%$ and ($**$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-MI}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-MI}} > \mu_{\mathcal{A}}$ (maximization problem).

ID	Algorithm	Number of evaluations		
	$\mathcal{A}$	100	200	300
19	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($*$)	($**$)	($*$)
20	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	
21	GA	($**$)	($**$)	($**$)
	B&B	NA	NA	NA
	NOMAD	($**$)	($**$)	($*$)

4.7 Conclusions

In this chapter a surrogate model approach for finding (near) optimal solutions to black-box mixed-integer global optimization problems within a very restricted number of function evaluations has been introduced. The algorithm, SO-MI, iteratively evaluates the computationally expensive simulation model at four chosen points of the variable domain in parallel and updates a cubic radial basis function surrogate model. Four perturbation methods are used to diversify the selection of candidates for the next sample point in every iteration, and based on scoring criteria the best point of each group is chosen. In general, more than four points could be chosen to make use of more processors. The algorithm SO-MI converges to the global optimum almost surely.

SO-MI has been shown to perform very well in comparison to a branch and bound algorithm for nonlinear problems [78], a genetic algorithm, and the NOMAD algorithm for multimodal mixed-integer problems [10] on 16 test problems from the literature containing unconstrained, constrained, unimodal, and multimodal problems, two application problems arising from structural optimization, and three application problems from optimal

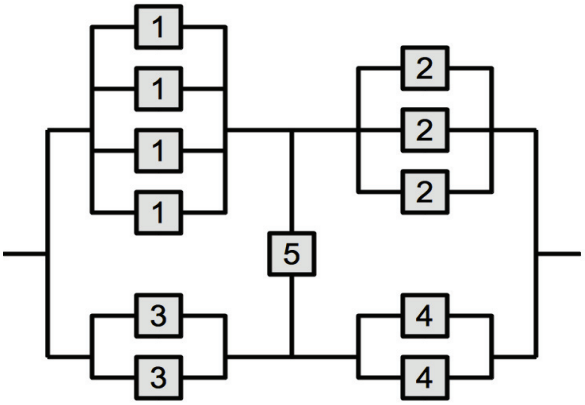


Figure 4.10: Best reliability-redundancy allocation for the bridge system. Redundancy has been added at components 1, 2, 3, and 4. See Appendix C for the problem description and data.

reliability design.

The numerical results show that SO-MI is able to find significantly better solutions than the other algorithms for 10 out of 16 literature test problems and all application problems when the number of allowed function evaluations is low (200 to 300 evaluations). On the remaining six problems, SO-MI performed only slightly worse than the best algorithm which is also reflected in a measure that is computed based on the deviation of the results obtained by each algorithm from the best solution. This measure shows that SO-MI performs in general better than all other algorithms for each problem group (unconstrained, constrained, structures problems, and reliability problems). Statistical tests on the differences of means for each problem also support this conclusion.

The genetic algorithm and branch and bound have for almost all problems the worst performance, whereas NOMAD is able to find competitive solutions for some problems. Note however that the results of these algorithms may be further improved by incorporating a surrogate model. The test problems included also a convex problem (problem 2) for which SO-MI performed best, and two special unimodal problems where the global optimum of the continuous relaxation is at an integer point (problems 3 and 9). As may be expected, branch and bound performed very well on problems 3 and 9, indicating that the algorithm may be preferred if the specific problem structure is known to be mathematically tractable.

However, for black-box problems, branch and bound requires in general too many function evaluations when computing lower bounds for the objective function value in the tree nodes. Also, for multimodal problems these lower bounds are not necessarily valid and pruning decisions may eventually be wrong.

Although developed for multimodal problems, the genetic algorithm does not perform well in the computational experiments. Genetic algorithms require in general many function evaluations (=population size $\times$ number of generations) to find good solutions. Thus, when only a very low number of evaluations can be allowed due to restrictions on the computational expense, either the number of individuals in each generation has to be reduced, which restricts the diversity in the population, or the number of generations has to be reduced, which reduces the evolutionary aspect of the algorithm.

The NOMAD algorithm is also a derivative-free method, and the C++ implementation incorporates a mixed-integer version and the variable neighborhood search that allows the algorithm to escape from local optima. There are local convergence results for NOMAD, but in the numerical experiments NOMAD is in general outperformed by SO-MI, especially on the structural optimization problems where the difficulty was in the black-box constraints rather than the objective function. However, compared to branch and bound and the genetic algorithm, NOMAD performed very well.

In conclusion, the introduced algorithm SO-MI extends the research area of using surrogate models for solving mixed-integer optimization problems. The computational results indicate that SO-MI is a promising algorithm and performs significantly better than commonly used algorithms for mixed-integer problems if dealing with multimodal black-box functions. In addition it can be shown that the SO-MI algorithm converges to the global optimum almost surely for multimodal mixed-integer problems with black-box objective functions. None of the other algorithms have such a proof for this type of problem. It is expected that the performance of SO-MI for black-box constrained problems can be further improved by replacing the penalty approach for example with an approach that uses a response surface for each constraint to eliminate infeasible-predicted candidate points from consideration [118]. A comparison of various ways for handling constraints exceeds however the scope of this chapter and is left for future research. Another extension for future research is considering the possibility of relaxing the integer constraints during the optimization if the objective function values can be computed when the integer variables assume continuous values.

Chapter 5

SO-I: A Surrogate Model Algorithm for Expensive Nonlinear Integer Programming Problems Including Global Optimization Applications

Abstract

This chapter presents the surrogate model based algorithm SO-I for solving purely integer optimization problems that have computationally expensive black-box objective functions and that may have computationally expensive constraints. The algorithm was developed for solving global optimization problems, meaning that the relaxed optimization problems have many local optima. However, the method is also shown to perform well on many local optimization problems, and problems with linear objective functions. The performance of SO-I, a genetic algorithm, a branch and bound algorithm for nonlinear optimization problems, and NOMAD has been compared on 17 test problems from the literature, and on eight realizations of two application problems. One application problem relates to hydropower generation, and the other one to throughput maximization. The average objective function values over several trials for each problem and hypothesis tests for differences in means show that SO-I outperforms the other algorithms for almost all

test problems. Only NOMAD performed significantly better than SO-I on four out of the 25 examined problems. Moreover, branch and bound failed to iterate on all application problems because of difficulties arising when evaluating the black-box objective function for the continuous relaxation or at infeasible points.

Abbreviations and Nomenclature

B&B	Branch and bound algorithm
GA	Genetic algorithm
NOMAD	Nonsmooth Optimization by Mesh Adaptive Direct Search
RBF	Radial basis function
SEM	Standard error of the means
SO-I	Surrogate Optimization - Integer
$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integer numbers
$\mathbf{u}$	Discrete decision variables, see equation (5.1d)
$f(\cdot)$	Objective function, see equation (5.1a)
$c_j(\cdot)$	jth constraint function, $j = 1, \ldots, m$, see equation (5.1b)
m	Number of constraints
k	Problem dimension
j	Index for the constraints
i	Index for the variables
u_i^l, u_i^u	Lower and upper bounds for the ith variable, see equation (5.1c)
Ω_b	Box-constrained variable domain
Ω	Feasible variable domain
$\mathcal{S}$	Set of already evaluated points
n_0	Number of points in initial experimental design
$q(\cdot)$	Auxiliary function for minimizing constraint violation in phase 1, see equation (5.2)
$f_p(\cdot)$	Objective function value augmented with penalty term, see equation (5.3)
$f_{\max}$	Objective function value of the worst feasible point found so far, see equation (5.3)
p	Penalty factor, see equation (5.3)
$v(\cdot)$	Squared constraint violation function, see equation (5.4)
$\boldsymbol{\chi}_\jmath$	$\jmath$th candidate point for next sample site, $\jmath = 1, \ldots, t$
n	Number of already sampled points
$s_b(\cdot)$	Radial basis function interpolant
$V_D(\cdot)$	Score for distance criterion, see equation (3.6)
$V_R(\cdot)$	Score for response surface criterion, see equation (3.7)
$V(\cdot)$	Weighted score, see equation (3.8)
ω_R, ω_D	Weights for response surface and distance criteria, respectively
H_0, H_1	Null hypothesis and alternative hypothesis, respectively

5.1 Introduction

This chapter presents an adaption of the SO-MI algorithm introduced in Chapter 4 to problems where all variables are restricted to be integers. Although SO-MI as it is implemented could be used for solving integer optimization problems, adapting the candidate point creation to only integer points is more efficient. Devising an algorithm for computationally expensive black-box optimization problems that have integrality constraints for *all* variables may come handy in several application problems. For example, similarly to the reliability-redundancy allocation problems described in Chapter 4, there are system reliability optimization problems that are only based on adding component redundancy [80, Chapters 3-5] (so-called redundancy allocation problems). Moreover, applications arising from throughput maximization are often integer problems [108]. This type of applications is encountered, for example, in production planning and facility layout [51, 134], but also in the design of networks-on-chip routers [69]. Another purely integer optimization problem arises in hydropower energy generation where several turbines of different efficiencies are installed in a hydropower plant, and the question is how many turbines of the same kind should be in use to maximize the total generated power given a certain amount of water that can be released from the reservoir [24, 85]. Furthermore, design optimization problems similar to the ones discussed in Chapter 4 can be of purely discrete nature. For example, if the coordinates of all nodes of the structure are fixed and only the cross-sectional areas or the wall thicknesses of the members have to be determined such that the total structural weight is minimized with respect to nodal displacement and stress constraints, the optimization problem has only integer variables.

Another important application is encountered in the management of the agricultural land use in a watershed. Depending on the land use, the phosphorus runoff may be beyond a certain threshold and may decrease the water quality significantly. One solution for decreasing the phosphorus runoff is to convert (or retire) parts of the land. This practice may however invoke high conversion costs, and thus the objective is to find an optimal strategy such that the land conversion costs are minimal while the phosphorus runoff is below a given limit. A highly nonlinear and computationally expensive simulation model has to be used to compute the phosphorus runoff. This application problem will be examined in detail in Chapter 6.

Similarly as in the mixed-integer case, the most commonly used algorithms in the literature for solving discrete optimization problems comprise branch and

bound methods (introduced by [82]) and evolutionary algorithms [12, 13, 94]. As discussed in Chapter 4, these algorithms may, however, not be suitable for solving computationally expensive black-box problems because too many function evaluations are needed to obtain a good solution.

Algorithms for solving purely discrete optimization problems with computationally expensive objective functions are rather scarce. Although the algorithms briefly discussed in Section 4.1 [33, 116] may be able to solve problems with only discrete variables, this option has not been studied in the literature. Moreover, altering the strategy of selecting the sample points in each iteration of these algorithms may improve their performance and efficiency as compared to using the mixed-integer version on purely integer problems.

Furthermore, the NOMAD algorithm [2, 4, 11] is able to solve purely integer problems, but the efficiency of NOMAD for problems of this type has not been examined in the literature.

The remainder of this chapter is organized as follows. In Section 5.2 the general mathematical description of discrete optimization problems is given. Section 5.3 introduces the surrogate model based algorithm SO-I (**S**urrogate **O**ptimization - **I**nteger) for discrete global optimization problems that may have computationally expensive constraints. In contrast to SO-MI, SO-I uses a first optimization phase for finding feasible points, i.e. an initial user-supplied feasible point is not necessary. SO-I has been compared to a genetic algorithm, NOMAD, and branch and bound on 17 test problems from the literature and two types of application problems (throughput maximization, and hydropower generation) that are described in Section 5.4. The numerical results are presented in Section 5.5. Because the focus is on computationally expensive functions, the computational effort is measured in the number of objective function evaluations rather than the CPU time needed by the algorithms. When dealing with computationally expensive objective functions that may take several hours for computing one function value, the algorithms' own computation time becomes insignificant. Section 5.6 concludes this chapter.

5.2 Constrained Integer Optimization Problems

The optimization problem type considered in this chapter is in general of the following form

$$\begin{aligned} & \min f(\mathbf{u}) && \text{(5.1a)} \\ \text{s.t.} \quad & c_j(\mathbf{u}) \leq 0,\ \forall j = 1, \ldots, m && \text{(5.1b)} \\ & -\infty < u_i^l \leq u_i \leq u_i^u < \infty,\ \forall i = 1, \ldots, k && \text{(5.1c)} \\ & u_i \in \mathbb{Z},\ \forall i = 1, \ldots, k && \text{(5.1d)} \end{aligned}$$

where $f(\mathbf{u}) \in \mathbb{R}$ denotes the costly black-box objective function, $c_j(\mathbf{u}) \in \mathbb{R}, j = 1, \ldots, m$, are the m costly black-box constraints, and u_i^l and u_i^u, where $u_i^l < u_i^u$, denote the lower and upper bounds on the variable u_i, respectively (box constraints). Denote Ω_b the intersection of the box-constrained variable domain with the integer lattices $\mathbb{Z}^k$, and by Ω the feasible variable domain. Throughout this chapter the variables $\mathbf{u}$ are assumed to be discrete. The set of already sampled points is denoted by $\mathcal{S}$, and it is assumed that all already sampled points are distinct.

5.3 SO-I: Surrogate Model Algorithm for Discrete Global Optimization Problems

The algorithm for solving discrete optimization problems with costly objective and constraint functions starts by building an initial experimental design. A symmetric Latin hypercube design [156] with $n_0 = 2(k+1)$ distinct points, where k denotes the problem dimension, is generated, and the values are rounded to the closest integers. The algorithm uses a cubic radial basis function interpolant as defined in equation (1.9), and similarly as for SO-MI, it must be ensured that the rank of matrix $\mathbf{P}$ in equation (1.11) is $k+1$. Thus, the initial experimental design is regenerated until $\text{rank}(\mathbf{P}) = k+1$.

The optimization process following the initial experimental design consists of two phases. The purpose of phase 1 is to find a feasible solution. The auxiliary function

$$q(\mathbf{u}) = \sum_{j=1}^{m} \max\left\{0, c_j(\mathbf{u})\right\}, \tag{5.2}$$

is minimized with respect to the box and integrality constraints defined in equations (5.1c) and (5.1d) until a first feasible point has been found ($q(\mathbf{u}) = 0$ for a feasible point). If the optimization problem has only box constraints or the initial experimental design contains at least one feasible point, optimization phase 1 is skipped.

In optimization phase 2, the penalized objective function

$$f_p(\mathbf{u}) = \begin{cases} f_{\max} + pv(\mathbf{u}) & \text{if } \mathbf{u} \text{ is infeasible} \\ f(\mathbf{u}) & \text{if } \mathbf{u} \text{ is feasible} \end{cases}, \tag{5.3}$$

is minimized subject to the box and integrality constraints in equations (5.1c) and (5.1d). Here, p denotes the penalty factor, and $f_{\max}$ is the feasible point with the largest function value found so far (the worst feasible point), and

$$v(\mathbf{u}) = \sum_{j=1}^{m} \max\left\{0, c_j(\mathbf{u})\right\}^2, \tag{5.4}$$

where it is assumed that $\mathbf{u} \in \Omega_b$, i.e. only points within the box-constrained domain are considered. Thus, if $v(\mathbf{u}) = 0$, the point $\mathbf{u}$ satisfies every constraint $c_j(\mathbf{u}), j = 1, \ldots, m$, in equation (5.1b) and the box constraints in equation (5.1c).

The penalty must be adjusted so that the function values at the infeasible points are higher than the best feasible objective function value. Otherwise the surrogate model would predict better objective function values in the infeasible region, the search might be drawn into that region, and many expensive function evaluations might be done without improving the solution. Therefore, to safeguard that the infeasible points have worse function values than the feasible points, their function value is set to that of the worst feasible point found so far plus a penalty term, which works similarly to a barrier term. To overcome numerical difficulties that might arise due to adding the penalty, function values that are higher than the median of all values of $f_p(\mathbf{u})$ are replaced by the median value. This also prevents the surface from oscillating wildly.

Both optimization phases work iteratively and in each iteration one new point for doing the next expensive function evaluation is added to the set of already sampled points. In order to minimize $q(\mathbf{u})$ in (5.2) and subsequently $f_p(\mathbf{u})$ in (5.3) a new algorithm for dealing with discrete variables has been developed. When minimizing $q(\mathbf{u})$ the goal is to find a point where the function value is non-positive. This point will then be a feasible point. The

goal in optimization phase 2 is to find a feasible point where the objective function value $f_p(\mathbf{u})$ is smaller than the value of the best feasible point found so far.

It is assumed that the objective and constraint functions are computationally expensive to evaluate and that the problem structure cannot be exploited. Therefore, using a response surface is a valid approach because no information other than the output of the simulation model for a given input variable vector is necessary. In this chapter a cubic radial basis function model is used, but every other surrogate model may in general be used.

A candidate point approach similar to the one described in Section 4.3.2 is applied to determine the next sample site. Two groups of candidate points are used, namely randomly generated integer points in Ω_b, and candidates generated by perturbing the best point found so far. In optimization phase 1 the best point found so far is the one with minimal $q(\mathbf{u})$, i.e. the point with minimal constraint violation. In optimization phase 2 the best point found so far is the *feasible* point with minimal objective function value. All variables of this point are perturbed with probability 0.5, and depending on how much the chosen variable should be perturbed, either the value 1, 2 or 3 is added or subtracted (at random). If the resulting value exceeds the variable's upper or lower bound, the value is set to the value of the variable bound that is exceeded, so that $\boldsymbol{\chi}_\jmath \in \Omega_b, \forall \jmath = 1, \ldots, t$, where $\boldsymbol{\chi}_\jmath$ denotes the $\jmath$th candidate point.

Given the sample data obtained so far, the parameters of the cubic RBF model are computed by solving the system (1.10). In optimization phase 1 the values of $q(\mathbf{u})$ in equation (5.2) are used, while in optimization phase 2 the values of $f_p(\mathbf{u})$ in equation (5.3) are used in the right hand side of (1.10) to compute the model parameters, where $\mathbf{u} \in \mathcal{S}$. The response surface is then used to predict the values of $q(\boldsymbol{\chi}_\jmath), \forall \jmath = 1, \ldots, t$, in optimization phase 1, and $f_p(\boldsymbol{\chi}_\jmath), \forall \jmath = 1, \ldots, t$, in optimization phase 2 at every candidate point. Note that the response surface is cheap to evaluate, and this step is therefore computationally inexpensive. The candidate points' predicted function values are scaled to the interval $[0, 1]$ so that low predicted values obtain a lower score, and high predicted values obtain a higher score V_R (response surface criterion, see equation (3.7) where s_{mix} must replaced by the RBF interpolant s_b defined in equation (1.9)). Next, the distance of all candidate points to the set of already sampled points $\mathcal{S}$ is calculated. These distances are scaled to the interval $[0, 1]$ so that high distances obtain a lower score, and low distances obtain a higher score V_D (distance

criterion, see equation (3.6)). Based on these two criteria the weighted score $V(\boldsymbol{\chi}_j), j = 1, \ldots, t$, defined in equation (3.8) is computed for each candidate point, and the candidate point with the lowest score is chosen as the next sample site.

In optimization phase 1, the weights in equation (3.8) are fixed to $\omega_R = 0.9$ for the response surface criterion, and $\omega_D = 0.1$ for the distance criterion. This emphasizes the local search in an attempt to find a feasible point quickly by giving an advantage to candidate points with low predicted function values.

In optimization phase 2, three different approaches have been examined for adjusting the weights in equation (3.8):

- *Version 1:* $\omega_R = 0.9$, $\omega_D = 0.1$ constant
- *Version 2:* In the nth iteration, $\omega_D = 1 - \log(n)/\log(N_{\max})$, $\omega_R = 1 - \omega_D$, where $N_{\max}$ denotes the maximum number of allowed iterations
- *Version 3:* The algorithm repeatedly cycles through a fixed pattern $(0, 0.1, 0.2, \ldots, 0.9, 1.0)$, i.e. the weights change in every iteration (e.g. in iteration 1 $\omega_D = 1$, $\omega_R = 0$; in iteration 2 $\omega_D = 0.9$, $\omega_R = 0.1$, etc.)

The adjustment in Version 1 may be regarded as a more local search since preference is always given to candidate points that have a low predicted objective function value, and the distance criterion has only a very low influence. Version 2 may be interpreted as a transition from a global search (initially candidate points that are far away from already sampled points are preferred) to a local search. The variable domain is explored during the first iterations in order to find promising regions where the global optimum might be. As the sampling continues, the search becomes more local and candidate points with low predicted objective function values obtain better scores, and therefore the promising regions of the variable domain are examined more thoroughly. Version 3 repeatedly transitions from a global to a local search in an attempt to escape from local optima if no further improvements in the vicinity of the best feasible point found so far could be obtained.

Furthermore, experiments were made where the minimum of an auxiliary function is used as criterion for finding the next sample point (Version 4). In particular, the point that minimizes the weighted score in equation (3.8) with cycling weights as in Version 3 has been used (see also the solid green graph for the weighted score function in Figure 3.1). Numerical

experiments showed that Versions 1 and 2 deliver in general better results than Versions 3 and 4, and that there are no differences between the performance of the algorithms when using Version 1 or 2. For reasons of space considerations the results of these numerical experiments are not presented here. In the following, Version 1 will be used in the comparison with the other algorithms, i.e. the weight for the response surface and the distance criterion are fixed to the values 0.9 and 0.1, respectively. As shown in Chapter 3, using the minimum of the response surface works well if the response surface is multimodal, but there is a potential risk of the algorithm getting stuck in a local optimum. Here the weights are adjusted such that the response surface criterion has more influence than the distance criterion, which enables the algorithm to prefer candidates with low predicted objective function values, but there is also the chance to escape from local optima.

After the candidate point with the best score has been determined, the costly objective and constraint functions are evaluated at that point, and the response surface parameters are updated with the new data. This process (candidate point generation, score calculation, response surface updating) iterates. Note that for building the response surface the variables are assumed to be continuous to obtain a smooth surface, but the generated candidate and sample points only have discrete variable values. The specific steps of SO-I with Version 1 as sampling strategy are given in Algorithm 6.

Algorithm 6 ***SO-I: Surrogate Optimization - Integer***

1. *Construct an initial experimental design:*
   (a) *Build a symmetric Latin hypercube design where all variables are integers and bounded only by the box constraints, and such that* $\text{rank}(\mathbf{P}) = k + 1$ *in equation (1.11).*
   (b) *Compute the costly objective and constraint function values.*
2. *Optimization phase 1: If there is no feasible point in the initial experimental design, minimize* $q(\mathbf{u})$ *in equation (5.2). Iterate until a feasible point has been found or the stopping criterion has been met:*
   (a) *Find the point with lowest auxiliary function value* $q(\mathbf{u}), \mathbf{u} \in \mathcal{S}$*, and denote this point by* $(\mathbf{u}^q_{\min}, q_{\min})$*.*
   (b) *Compute the response surface parameters based on the points in* $\mathcal{S}$ *and their corresponding auxiliary function values* $q(\mathbf{u})$*.*

(c) *Generate candidate points by perturbing* $\mathbf{u}^q_{\min}$*, and by randomly selecting points from the variable domain* Ω_b*. Eliminate all candidates that are already contained in* $\mathcal{S}$ *from further consideration. Compute the candidate point scores using equations (3.6), (3.7), and (3.8) with fixed weights* $\omega_D = 0.1, \omega_R = 0.9$*, and do the expensive objective and constraint function evaluation at the candidate point with the best score.*

3. *Optimization phase 2: Minimize* $f_p(\mathbf{u})$ *in equation (5.3). Iterate until the stopping criterion has been met:*

   (a) *Find the feasible point with lowest objective function value and denote the pair by* $(\mathbf{u}^f_{\min}, f_{\min})$*. Find the feasible point with highest objective function value and denote the pair by* $(\mathbf{u}^f_{\max}, f_{\max})$*.*

   (b) *Set the function values to* $f_p(\mathbf{u}) = f_{\max} + 100v(\mathbf{u})$*,* $\forall \mathbf{u} \in \mathcal{S}$ *that are infeasible, and* $f_p(\mathbf{u}) = f(\mathbf{u})$ *for all feasible points.*

   (c) *Compute the response surface parameters using the points* $\mathbf{u} \in \mathcal{S}$ *and their corresponding function values* $f_p(\mathbf{u})$*.*

   (d) *Generate a large set of candidate points by perturbing* $\mathbf{u}^f_{\min}$ *and by randomly selecting points from the variable domain* Ω_b*. Eliminate all candidates that are already contained in* $\mathcal{S}$ *from further consideration. Compute the candidate point scores using equations (3.6), (3.7), and (3.8) with weights* $\omega_D = 0.1$ *and* $\omega_R = 0.9$*. Do the expensive objective and constraint function evaluation at the candidate point with the best score.*

4. *Return the best point found.*

Note that although the values of the objective function are not used during optimization phase 1, the values are computed (often the objective and constraint function values are the output of the same computationally expensive simulation, which is assumed also here) and later on used in optimization phase 2.

The convergence of the algorithm using the candidate point sampling approach is trivial and for reasons of completeness stated in Theorem 2.

Theorem 2 *Let f be the real-valued function defined on $\Omega_b \subset \mathbb{Z}^k$, and denote the cardinality $|\Omega_b| = \kappa$. Let $f(\mathbf{u}^*) = \min_{\mathbf{u}\in\Omega} f(\mathbf{u}) > -\infty$ be the feasible global minimum of f, i.e. $f(\mathbf{u}) > f(\mathbf{u}^*)\ \forall \mathbf{u} \neq \mathbf{u}^*, \mathbf{u} \in \Omega$. Assume the candidate points for the next sample site are generated as described in Algorithm 6 (Steps 2(c) and 3(d), respectively). Denote $\mathbf{u}_1$ the first feasible point found. Define the sequence of feasible random vectors $\{\mathbf{u}_n^*\}_{n\geq 1}$ as $\mathbf{u}_1^* = \mathbf{u}_1$, $\mathbf{u}_{n+1}^* = \mathbf{u}_{n+1}$ if $f(\mathbf{u}_{n+1}) < f(\mathbf{u}_n^*)$ and $c_j(\mathbf{u}_{n+1}) \leq 0\ \forall j$, and $\mathbf{u}_{n+1}^* = \mathbf{u}_n^*$ otherwise. Then, $\mathbf{u}_n^* \to \mathbf{u}^*$ as $n \to \kappa - 1$.*

Proof 2 *The convergence of the algorithm follows from a simple counting argument. The number of points in the box-constrained variable domain Ω_b is finite, and no point will be sampled more than once. Moreover, every point in Ω_b may become a candidate point because some of the candidates are generated by uniformly selecting integer points from Ω_b. Thus, as the algorithm proceeds, the probability that the next chosen point will be the global optimum goes in the limit to 1.*

5.4 Test Problems

5.4.1 Test Setup

Algorithms for solving discrete global black-box optimization problems have not widely been studied in the literature. Therefore, the SO-I algorithm introduced in Section 5.3 is in the following compared to a branch and bound algorithm for nonlinear optimization problems, a genetic algorithm, and NOMAD. Branch and bound and genetic algorithms are often used for solving discrete optimization problems. Note, however, that it is assumed that nothing about the mathematical problem structure is known, and thus reformulation strategies cannot be applied to enhance the performance of branch and bound. The performance of NOMAD for purely integer problems has not been studied in the literature yet.

The branch and bound method used is the same as described in Section 4.4. For the genetic algorithm, Matlab's implementation (function `ga`) has been used, which is able to solve discrete problems (Matlab 2011b and newer versions), and works for purely integer problems significantly better than for mixed-integer problems. Furthermore, NOMAD version 3.5.0 has been used. Note, that NOMAD is included in the *OPTI Toolbox* (http://www.i2c2.aut.ac.nz/Wiki/OPTI/), which is a Matlab toolbox for optimization and simplifies the use of NOMAD as compared to its C++ implementation. NOMAD is used with the variable neighborhood search

technique (setting VNS 0.75) and the progressive barrier approach (setting PB) is used for treating the constraints.

The maximum number of allowed function evaluations for each algorithm is 400. The genetic algorithm has a population of 20, and stops when 400 function evaluations have been done. A time limit has not been defined for the algorithms because the performance of the algorithms is measured with respect to the solution quality after an equal number of function evaluations. Assuming that one objective function evaluation requires a time consuming simulation, the algorithms' own computation times become insignificant.

It is expected that branch and bound will not deliver good results for multimodal problems because obtaining valid lower bounds for the objective function value is in general not possible if the problem structure cannot be exploited. However, the algorithm has been included to show how the surrogate model based algorithms compare on unimodal problems. Moreover, it can be expected that the performance of branch and bound, the genetic algorithm, and NOMAD may be improved if a surrogate model is incorporated. For the branch and bound algorithm, for example, a surrogate model could be used when optimizing the relaxed subproblems in the tree nodes (although guarantees for the validity of the lower bounds can still not be given). For the genetic algorithm surrogate models could be applied for creating a new population. The surrogate model could be used to predict the feasibility and fitness of each individual in the new generation, and thus individuals that are predicted to have a low fitness value, or that are predicted infeasible could be excluded from the next generation and replaced by a better individual. The used NOMAD version has a feature that allows the algorithm to use surrogate models, but the user has to implement the surrogate model him/herself. Moreover, implementations of the described algorithm extensions are not available. Thus, these extensions cannot be included in the following comparison.

5.4.2 Generic Test Problems

The performance of the algorithms has been compared on 17 test problems from the literature, and eight realizations of two application problems arising in throughput maximization and hydropower generation maximization. Note that algorithms for purely integer black-box optimization problems have not been studied in depth in the literature, and thus finding a representative and widely used test bench of problems is impossible.

Thus, some of the 17 test problems are alterations of the problems used by Koziel and Michalewicz [77] (continuous global optimization test problems). Furthermore, several box-constrained continuous global optimization test problems have been examined as well as a box-constrained convex problem, and a purely linear optimization problem, and integrality constraints have been added to the problem formulations. Three test problems from the collection of Mixed Integer Nonlinear Programming models [23] have also been included in the test bench with integrality constraints for all variables.

The problem dimensions range between 2 and 30, and the cardinality of the box-constrained variable domain Ω_b ranges between 5324 and 101^{13}. Most test problems have been derived from continuous global optimization test problems by imposing integrality constraints on every variable. Some test problems have been used in different variations. On the one hand the number of variables has been changed in order to examine the behavior of the algorithms as the problem dimension increases. On the other hand the variable domain has been changed in order to examine the influence of larger domains on the solution quality. Although the 17 test problems have analytical descriptions (given in Appendix D) the problems are treated as black-boxes, i.e. the mathematical problem structure cannot be exploited by any of the algorithms. The test problems are used to examine the algorithms' ability to solve problems with different characteristics such as multimodality, linear and nonlinear constraints and objective functions, or binary variables when no information about the problem structure can be exploited. Thus, the test bench represents a variety of problem characteristics that may be encountered when solving real-world black-box problems.

5.4.3 Throughput Maximization Application

In addition to the 17 literature test problems two realizations of an application problem about throughput maximization [108] have been used to compare the algorithms. The problem of throughput maximization given restrictions on the total buffer size B and service rate R is encountered in different application areas such as production planning and facility layout [51, 134]. The space in the production facility is limited and practical restrictions enforce limits on the total service rate at each station.

Throughput maximization is also an important topic in data networks in computer science where the goal is to maximize the throughput subject to given restrictions on the buffer size. Especially for networks-on-chip

router design, the on-chip network should be implemented with very little area overhead in order to minimize implementation costs. The input buffers of on-chip routers take a significant portion of the silicon area of networks-on-chips, and thus their size should be as small as possible [69].

Here, the throughput maximization of a flow line with h stations is considered. The buffer storage in front of station 1 is infinite, whereas the buffer storages $b_2, b_3, \ldots, b_h$ in front of stations $2, 3, \ldots, h$ are finite. Every station i has a single server with service rate r_i, $i = 1, \ldots, h$. The service time required by every job is known. If the buffer at station i is full, then station $i-1$ is blocked because no job finished at station $i-1$ can be transferred to the buffer at station i. The total buffer space and the total service rate is limited, i.e.

$$\sum_{i=2}^{h} b_i \leq B \quad \text{(buffer space restriction)}, \tag{5.5a}$$

$$\sum_{i=1}^{h} r_i \leq R \quad \text{(service rate restriction)}, \tag{5.5b}$$

where B and R are known constants. The goal is then to find a buffer allocation and service rates such that the throughput (average output of the flow line per unit time) is maximized.

Two problem realizations have been considered. First, a small scale problem with $h = 3$ stations has been used. For this problem $B = R = 20$. There are five variables $(b_2, b_3, r_1, r_2, r_3)$ and their lower and upper bounds have been set to 1 and 20, respectively. Secondly, a flow line with $h = 12$ stations has been considered where $R = B = 80$. For this problem the lower and upper bounds on the variable values have also been set to 1 and 20, respectively, and there are 23 variables $(b_2, \ldots, b_{12}, r_1, \ldots, r_{12})$. Altogether, 2050 jobs have to be processed for each problem realization. To calculate the objective function value, the processing of all jobs has to be simulated. The system is initially empty, and then 2000 jobs are released. The time T for releasing the 50 last jobs is recorded, and the throughput is calculated as $50/T$.

5.4.4 Hydropower Generation Maximization Application

Furthermore, the algorithms were compared on six realizations of a hydropower generation application problem. The goal is to maximize the power output over one day for hydropower plants with multiple types of generating units. Electricity produced by hydropower plants is an important source of renewable energy around the world. Large hydropower facilities, like the Three Gorges Project in China [24, 85], are designed with different generator types (denoted by u_i, $i = 1, \ldots, k$) because some generators are more efficient (in terms of power generated/volume of discharged water) with high water heads and others are more efficient with lower heads. The water head is the height of the water in the reservoir relative to its height after discharge and thus determines the potential energy.

Hydropower planning is done for different time periods. For longer time periods (e.g., one or more weeks), stochastic analysis is more important since the weather in the future affects the inflow. For planning over a shorter period (like one day), stochastic factors are less important because Q_j, the total amount of water to be released from reservoir j, over one day and the water head is known at the beginning of the day.

In this application problem a short term analysis of how to distribute Q_j units of water over one time period, (e.g., a day), among all different generator units is done. This problem is inspired by current research in the Three Gorges Project-Gezhouba system on how to allocate water over a year long period to determine what the Q_j for each day should be. For computational reasons the allocation of water among different generating units cannot be considered directly in an annual analysis. The problem examined here is a subproblem of the annual hydroelectric power generation problem to determine the water allocation among generating units of different types. The current Three Gorges Project analysis solves this subproblem associated with the daily allocation of water among the generator units by exhaustive search on the integer values of u_i, which is much less efficient than the SO-I method proposed here. Because the Three Gorges Dam is such a large power source in the power grid, the focus of this study is to determine the maximum power output possible, assuming other capacities of elements of the power grid will be adjusted by long term planning after knowing the maximum power output of the Three Gorges system. Thus, there are no maximum hourly demand constraints incorporated in the optimization problem. Of course, if the problem is changed to examine optimization under

additional demand constraints, it is expected that SO-I will perform similarly or even better since the constraints further restrict the parameter space.

The general problem description follows. Given are five types of hydropower generating units $u_1, \ldots, u_5$ that have minimum and maximum capacities of the water $q_i, i = 1, \ldots, 5$, that runs through the respective generator type. All identical generator units of type i will receive the same amount of water q_i. The goal is to determine how many identical units of each generator type should be used in order to maximize the total power generated while not exceeding the maximum amount of water Q_j that may be released from reservoir j. For each variable set $u_1, \ldots, u_5$ the amount of water q_i, $i = 1, \ldots, 5$, that goes through the generating unit of type i must be adjusted such that the generated power is maximized while not exceeding Q_j.

Considered were two groups of hydropower test problems related to water distribution among generator types. The first problem group has only one reservoir and one hydropower plant with five different generator types. The second group has two reservoirs and two hydropower plants, one upstream with two different generator types that are more efficient with higher water heads, and one further downstream with three different generator types that are more efficient with lower water heads. For both groups three different limits on the maximal available water Q_j were considered, representing the different water heads at different times of the year.

Table 5.1 summarizes the characteristics of all test problems. Note that all variables are assumed to be discrete. The column "ID" contains the problem identification number. The column "Notes" gives references about the problem's origin or other characteristics that are considered "special". For example, problem 16 is convex and problem 17 is a purely linear integer programming problem. The last column indicates if the relaxed problem (all variables continuous) has several local, global and/or stationary points (denoted by MM), or if the relaxed problem has only one local optimum and no other stationary points (denoted by UM). Furthermore, it is indicated how many linear constraints (LC) and nonlinear constraints (NLC) each problem has, and if the objective function is linear (lin.obj.) or a black-box (BB objective). Note, that although these test problem characteristics are known, they are not exploited by the compared algorithms. The test problems are used to examine the suitability of the algorithms for finding (near) optimal solutions to a wide range of problems with different

Table 5.1: Summary of test problems
(a) http://www.aridolan.com/ga/gaa/MultiVarMin.html; ^{c} denotes constrained problems;
see Appendix D for further information.

ID	k	Domain	Notes	Relaxed problem characteristics
1^c	2	$[13, 100] \times [0, 100]$	[77]	MM, 2 NLC
2	5	$[-100, 100]^5$	(a)	MM
3^c	5	$[0, 10]^3 \times [0, 1]^2$	[23]	lin.obj., 2 NLC, 3 LC
4^c	5	$[78, 102] \times [33, 45] \times [27, 45]^3$	[77]	UM, 6 NLC
5^c	7	$[-10, 10]^7$	[77]	UM, 4 NLC
6^c	13	$[0, 1]^{10} \times [0, 100]^3$	[77]	MM, 9 LC
7	10	$[3, 9]^{10}$	[23]	UM
8^c	16	$[0, 1]^{16}$	[23]	MM, 7 NLC
9	12	$[-1, 3]^{12}$	Rastrigin [142]	MM
10	30	$[-1, 3]^{30}$	Rastrigin [142]	MM
11^c	25	$[0, 10]^{25}$	[77]	MM, 1 NLC, 1 LC
12	20	$[0, 1]^{20}$	Schoen [128]	MM
13	10	$[3, 99]^{10}$	[23]	MM
14^c	13	$[0, 100]^{13}$	[77]	MM, 9 LC
15	12	$[-10, 30]^{12}$	Rastrigin [142]	MM
16	8	$[-10, 10]^8$	convex	UM
17^c	5	$[0, 10]^3 \times [0, 1]^2$	linear	lin.obj., 3 LC
18^c	5	$[1, 20]^5$	max. throughput	BB objective, 2 LC
19^c	23	$[1, 20]^{23}$	max. throughput	BB objective, 2 LC
$20a^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 1 NLC
$20b^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 1 NLC
$20c^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 1 NLC
$21a^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 2 NLC
$21b^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 2 NLC
$21c^c$	5	$[0, 10]^5$	max. hydropower	BB objective, 2 NLC

characteristics without exploiting the problem structure.

5.5 Numerical Results

In the following the abbreviation GA stands for genetic algorithm, B&B denotes the branch and bound algorithm, and SO-I is the surrogate model based algorithm. For reasons of better presentation the test problems have been divided into unimodal, unconstrained multimodal, constrained multimodal, binary, linear, throughput maximization, and hydropower generation problems. Thirty trials were made with each algorithm for every problem, i.e. 30 different initial symmetric Latin hypercube designs were used for the SO-I algorithm, 30 different initial starting points were used for B&B and NOMAD, and 30 different initial populations were used with GA. Note that the first point in the experimental design (SO-I), in the initial population (GA), and the initial starting point (B&B, NOMAD) were the same for each algorithm for the same trial of a given problem in order to obtain a fair comparison.

The numerical results are presented in Tables 5.2-5.16. The average ("mean") of the best *feasible* solution found, and the standard errors of the means ("SEM") after 100, 200, 300, and 400 function evaluations are shown in columns "100 eval.", "200 eval.", "300 eval.", and "400 eval.", respectively. These numbers are computed based on the successful trials (feasible points have been found) for each algorithm. Thus, if an algorithm failed to find a feasible solution within 400 evaluations for two out of 30 trials, for example, the average was computed from the remaining 28 successful trials. The number of trials of each algorithm for which no feasible solution was found within 400 evaluations is reported in column "#NF". The column "dim" shows the dimension of each test problem, and the entries in the column "$|\Omega_b|$" are the cardinality of the box-constrained domain. The column "ID" contains the problem identification number as defined in Table 5.1, and the superscript ^{c} at that number indicates that the problem has additional constraints besides the variables' upper and lower bounds. The best solution in each column for every problem is marked by boxes, and the SEM value is given in italic font. Table entries marked by a dash denote that the respective algorithm was not able to find a feasible solution within the given number of function evaluations for all 30 trials. Additionally, the results of hypothesis tests for differences in means for all problem classes are given.

5.5.1 Unimodal Problems

The numerical results for the unimodal test problems are given in Table 5.2. The results show that SO-I reaches for every problem the best result among all algorithms. B&B reaches the same results for test problem 13 after 200 function evaluations and for problem 7. B&B does not find any feasible solution within 400 function evaluations for test problems 4 and 5. NOMAD is able to find a solution with the same average objective function value as SO-I for test problem 13 after more than 300 function evaluations, and for problem 16 after more than 200 evaluations. GA found the best solution only for test problem 7 after more than 300 function evaluations.

Comparing the results of GA and NOMAD shows that it is not possible to determine which algorithm performs better. There are test problems for which GA initially outperforms NOMAD, and others where NOMAD outperforms GA. The standard errors of the means are except for problem 4 lowest for SO-I. GA and NOMAD were able to find feasible solutions for both constrained problems (problems 4 and 5) in all trials, whereas SO-I was not able to find a feasible solution for eight instances of test problem 4. The relative errors between the global optimum (or the best solution found over all trials and all algorithms)[1] and the average function value found by SO-I after 400 function evaluations are zero for the unconstrained problems. For test problem 5 the relative error is about 10% and for problem 4 the relative error is about 1%.

Problems 7 and 13 have a similar structure, but the variable domain of problem 13 is larger than that of problem 7. SO-I is able to find for both problems the best solution within 100 function evaluations. Also NOMAD was able to find the optimal solutions of both problems, but it needed more than 300 evaluations. In contrast, GA is able to find the optimal solution for problem 7 after more than 300 evaluations, but the result for problem 13 is about 3% worse than the optimal solution. Also B&B finds the best solution for test problem 7 within 100 evaluations, and for test problem 13 after more than 100 evaluations. Thus, none of the compared algorithms seems to be significantly influenced by the larger variable domain.

Since SO-I reaches the best solution for every problem, only the hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}$ and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$, where $\mathcal{A} \in \{\text{NOMAD, B\&B, GA}\}$ has to be tested. The results of the hypothesis testing in Table 5.3 show that SO-I reaches a significantly lower mean value than B&B for three out of

[1] See Appendix D for the optimal values.

five problems, i.e. in these cases the null hypothesis H_0 can be rejected for $\mathcal{A} =$ B&B. Compared to NOMAD and GA the means of SO-I are especially within 200 evaluations significantly better.

Figure 5.1 shows the objective function value averaged over thirty trials for test problem 13 for each algorithm. The error bars illustrate the standard deviations. Test problem 13 is an unconstrained minimization problem. The figure shows that SO-I is able to reduce the average objective function value fastest and finds the optimal solution within the fewest number of function evaluations. B&B finds the optimal solution within slightly more than 100 evaluations, whereas NOMAD needs more than 350 evaluations to reach comparable results. GA did not find the optimal solution within 400 evaluations.

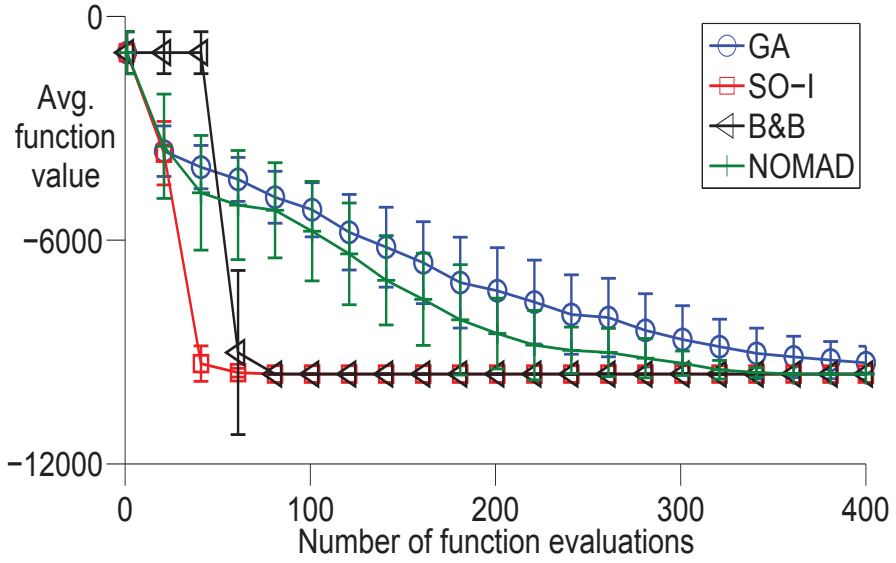


Figure 5.1: Unconstrained unimodal minimization problem, test problem 13. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 5.2: Unimodal problems. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
4^c	GA	0	mean	-29425.68	-29779.80	-29956.08	-30073.77	5	$25 \cdot 13 \cdot 19^3$
			SEM	*49.56*	*46.08*	*47.34*	*43.30*		
	SO-I	8	mean	-29729.50	-30045.82	-30188.51	-30283.84		
			SEM	*115.19*	*74.61*	*55.24*	*37.23*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	-29384.48	-29816.84	-30048.45	-30192.67		
			SEM	*97.09*	*64.94*	*57.81*	*35.29*		
5^c	GA	0	mean	-	2922.30	1121.20	896.53	7	21^7
			SEM	-	*1358.99*	*83.94*	*29.21*		
	SO-I	0	mean	236468.10	1278.57	842.90	771.40		
			SEM	*180228.60*	*247.93*	*34.76*	*14.97*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	742037.97	18065.77	2189.17	1770.50		
			SEM	*403498.14*	*8768.64*	*517.71*	*462.58*		
7	GA	0	mean	-22.83	-36.51	-42.78	-43.13	10	7^{10}
			SEM	*0.95*	*0.83*	*0.35*	*0.00*		
	SO-I	0	mean	-43.13	-43.13	-43.13	-43.13		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	-43.13	-43.13	-43.13	-43.13		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	NOMAD	0	mean	-27.71	-36.82	-40.48	-43.13		
			SEM	*0.72*	*0.85*	*0.69*	*0.00*		

13	GA	0	mean	-5182.64	-7348.78	-8654.65	-9289.87	10	97^{10}
			SEM	*132.04*	*211.49*	*166.52*	*81.24*		
	SO-I	0	mean	-9591.72	-9591.72	-9591.72	-9591.72		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	-8654.53	-9591.72	-9591.72	-9591.72		
			SEM	*937.19*	*0.00*	*0.00*	*0.00*		
	NOMAD	0	mean	-5759.21	-8504.33	-9301.37	-9591.72		
			SEM	*243.53*	*172.92*	*60.00*	*0.00*		
16	GA	0	mean	101.68	32.82	10.24	2.72	8	21^{8}
			SEM	*8.96*	*3.82*	*2.36*	*0.95*		
	SO-I	0	mean	0.15	0.00	0.00	0.00		
			SEM	*0.05*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	1538.67	1538.67	260.34	36.06		
			SEM	*101.17*	*101.17*	*99.67*	*10.93*		
	NOMAD	0	mean	22.38	3.79	0.00	0.00		
			SEM	*2.13*	*1.67*	*0.00*	*0.00*		

Table 5.3: Unimodal problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations.
^{c} denotes constrained problems.
$(*)$ denotes significance at $\alpha = 5\%$ and $(**)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$.

ID	Algorithm	Number of evaluations			
	$\mathcal{A}$	100	200	300	400
	GA	$(*)$	$(**)$	$(**)$	$(**)$
4^c	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD				
	GA	$(**)$		$(**)$	$(**)$
5^c	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD				$(**)$
	GA	$(**)$	$(**)$		
7	B&B				
	NOMAD	$(**)$	$(**)$		
	GA	$(**)$	$(**)$	$(**)$	$(**)$
13	B&B				
	NOMAD	$(**)$	$(**)$	$(**)$	$(**)$
	GA	$(**)$	$(**)$	$(**)$	$(**)$
16	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD	$(**)$			

5.5.2 Unconstrained Multimodal Problems

For the unconstrained multimodal problems, NOMAD and SO-I perform best (see Table 5.4). NOMAD finds the best solution for three out of four problems, whereas SO-I finds the best solution for two of the four problems. B&B is outperformed by GA for all four problems. For test problems 10 and 15, B&B was not able to find any improvement of the initially given point in any trial. Thus, the reported numbers are the mean and SEM of the starting points.

Problems 9 and 10 differ only with respect to the number of variables (12 and 30 dimensions, respectively). SO-I and NOMAD are both able to find the optimal solution for the smaller dimensional problem 9. For problem 10, NOMAD and SO-I deviate from the optimal solution by 3% and 9%, respectively, indicating that the solution quality for both algorithms slightly decreases as the dimension increases. The solution found by GA for the 12-dimensional problem 9 is about 9% worse than the optimal solution after 400 evaluations, whereas for problem 10 the difference is almost 43%. B&B was able to find improvements of the starting point for problem 9, but for problem 10 no improvements could be found within 400 evaluations. Thus, all algorithms are affected by the larger dimension. NOMAD is least affected, and the performance of B&B decreases most.

The results of the hypothesis tests for differences in means are given in Table 5.5. Since NOMAD reaches for this problem class better results than SO-I for two test problems (2 and 10), also the hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$, and $H_1 : \mu_{\text{SO-I}} > \mu_{\text{NOMAD}}$ has to be tested. The hypothesis tests show that the results reached by NOMAD are significantly better than those of SO-I at the 5% level for problem 2 and at the 1% level for problem 10, i.e. the null hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ must be rejected. On the other hand, the means of SO-I are significantly lower than those of NOMAD at the 1% level for test problem 15. For test problems 9, 10, and 15 the results of SO-I are significantly better than those of GA and B&B.

The objective function values averaged over the 30 trials and error bars for the standard deviations for test problem 9 are illustrated in Figure 5.2 for all four algorithms. The figure shows that SO-I and NOMAD perform about equally well with NOMAD having a slightly larger standard deviation. GA is able to iteratively improve the average objective function value over the 400 function evaluations, whereas B&B finds improvements only after more

than 200 evaluations.

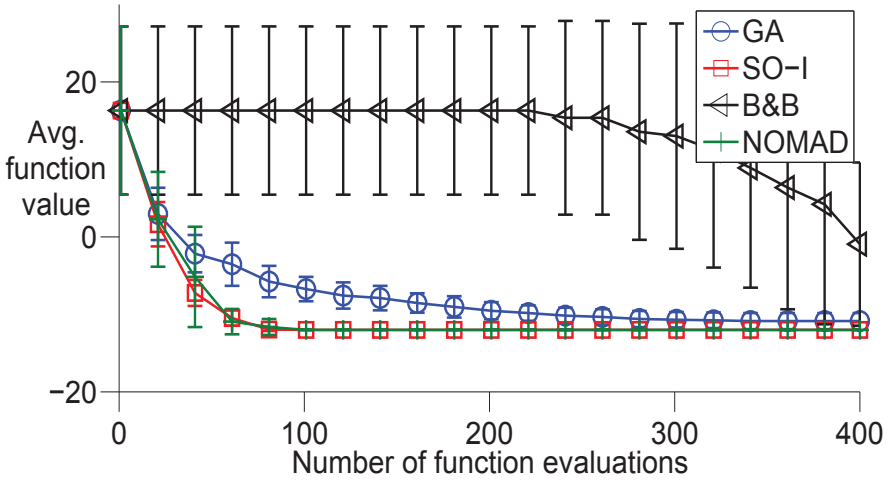


Figure 5.2: Unconstrained multimodal minimization problem, test problem 9. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 5.4: Unconstrained multimodal problems. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
2	GA	0	mean	-281.49	-377.59	-407.98	-427.91	5	201^5
			SEM	*13.24*	*9.69*	*8.01*	*7.38*		
	SO-I	0	mean	-334.70	-390.87	-426.06	-437.16		
			SEM	*13.79*	*14.94*	*13.02*	*12.09*		
	B&B	0	mean	607.78	-52.89	-311.68	-311.68		
			SEM	*103.62*	*94.70*	*5.85*	*5.85*		
	NOMAD	0	mean	-364.72	-439.81	-471.87	-485.73		
			SEM	*11.96*	*14.05*	*9.82*	*9.18*		
9	GA	0	mean	-6.73	-9.50	-10.70	-10.87	12	5^{12}
			SEM	*0.29*	*0.20*	*0.18*	*0.19*		
	SO-I	0	mean	-12.00	-12.00	-12.00	-12.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	16.30	16.30	13.00	-0.93		
			SEM	*1.98*	*1.98*	*2.66*	*1.92*		
	NOMAD	0	mean	-12.00	-12.00	-12.00	-12.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
10	GA	0	mean	-0.07	-8.63	-13.63	-17.17	30	5^{30}
			SEM	*1.01*	*0.83*	*0.68*	*0.66*		
	SO-I	0	mean	-16.00	-26.10	-26.93	-27.23		
			SEM	*0.31*	*0.21*	*0.15*	*0.16*		
	B&B	0	mean	47.53	47.53	47.53	47.53		
			SEM	*2.92*	*2.92*	*2.92*	*2.92*		
	NOMAD	0	mean	-19.40	-28.63	-28.97	-29.17		
			SEM	*1.69*	*0.47*	*0.42*	*0.41*		
15	GA	0	mean	466.63	188.80	77.73	33.83	12	41^{12}
			SEM	*30.33*	*17.21*	*7.92*	*4.52*		
	SO-I	0	mean	-11.97	-12.00	-12.00	-12.00		
			SEM	*0.03*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	2650.17	2650.17	2650.17	2650.17		
			SEM	*182.57*	*182.57*	*182.57*	*182.57*		
	NOMAD	0	mean	19.57	14.77	-1.70	-10.67		
			SEM	*2.22*	*1.49*	*2.80*	*1.33*		

Table 5.5: Unconstrained multimodal problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations. ($**$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$; ($\diamond$) denotes significance at $\alpha = 5\%$ and ($\diamond\diamond$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ and $H_1 : \mu_{\text{SO-I}} > \mu_{\text{NOMAD}}$.

ID	Algorithm	Number of evaluations			
	$\mathcal{A}$	100	200	300	400
	GA	($**$)			
2	B&B	($**$)	($**$)	($**$)	($**$)
	NOMAD			($\diamond$)	($\diamond$)
	GA	($**$)	($**$)	($**$)	($**$)
9	B&B	($**$)	($**$)	($**$)	($**$)
	NOMAD				
	GA	($**$)	($**$)	($**$)	($**$)
10	B&B	($**$)	($**$)	($**$)	($**$)
	NOMAD	($\diamond$)	($\diamond\diamond$)	($\diamond\diamond$)	($\diamond$)
	GA	($**$)	($**$)	($**$)	($**$)
15	B&B	($**$)	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)	

5.5.3 Constrained Multimodal Problems

The results for the constrained multimodal problems are given in Table 5.6. The results show that SO-I performs constantly best for two of the four examined problems. NOMAD outperforms SO-I for problem 11 for less than 200 function evaluations. GA found the optimal feasible solution for test problem 1 after more than 100 function evaluations, and outperformed NOMAD and SO-I for problem 11. B&B did not find any feasible solution within 400 evaluations for problems 1 and 11, and had for the other two problems difficulties finding a feasible solution. For test problem 1 also NOMAD was not able to generate a feasible solution for any of the 30 trials. GA also had difficulties finding feasible solutions within 100 function evaluations for problems 1, 6, and 14.

The results of the hypothesis tests are given in Table 5.7. For test problem 11 NOMAD and GA reached better results than SO-I for 100, 200, and 400 evaluations, and thus also the hypotheses $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ ($H_1 : \mu_{\text{SO-I}} > \mu_{\text{NOMAD}}$), and $H_0 : \mu_{\text{SO-I}} = \mu_{\text{GA}}$ ($H_1 : \mu_{\text{SO-I}} > \mu_{\text{GA}}$) have been tested. For this test problem the null hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ can be rejected at the 5% significance level for up to 100 function evaluations, and the hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\text{GA}}$ can be rejected at the 1% significance level for up to 100 evaluations. For problem 14, B&B reaches a significantly better result than SO-I after 300 and more function evaluations. For the remaining problems SO-I reaches significantly better results than B&B, and it reaches significantly better results than NOMAD for test problems 1, 6, and 14.

Figure 5.3 shows the average objective function value for feasible points versus the number of function evaluations for test problem 6. The error bars illustrate the standard deviations. The offset of the graphs indicates how many function evaluations each algorithm needed at most to find a first feasible solution in the 30 trials. As can be seen, for this test problem SO-I was most successful and needed fewer than 50 evaluations to find a first feasible point in all 30 trials. After the feasible point had been found, the average objective function value could be improved significantly within the next 50 evaluations. B&B, GA, and NOMAD also found feasible points in all 30 trials for this test problem, but needed significantly more evaluations. The average objective function value could also not be improved as much as for SO-I.

Table 5.6: Constrained multimodal problems. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
1^c	GA	2	mean	-	-3971.00	-3971.00	-3971.00	2	8888
			SEM	-	*0.00*	*0.00*	*0.00*		
	SO-I	0	mean	-3971.00	-3971.00	-3971.00	-3971.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	30	mean	-	-	-	-		
			SEM	-	-	-	-		
6^c	GA	0	mean	-	-	-6.07	-6.07	13	$2^{10} \cdot 101^3$
			SEM	-	-	*0.59*	*0.59*		
	SO-I	0	mean	-13.90	-14.50	-14.73	-14.83		
			SEM	*0.19*	*0.15*	*0.12*	*0.10*		
	B&B	0	mean	-	-12.00	-12.00	-12.00		
			SEM	-	*0.00*	*0.00*	*0.00*		
	NOMAD	0	mean	-	-	-5.97	-5.97		
			SEM	-	-	*0.03*	*0.03*		
11^c	GA	0	mean	-0.14	-0.17	-0.19	-0.22	25	11^{25}
			SEM	*0.00*	*0.00*	*0.00*	*0.01*		
	SO-I	0	mean	-0.13	-0.16	-0.20	-0.21		
			SEM	*0.00*	*0.01*	*0.01*	*0.01*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	-0.14	-0.16	-0.17	-0.17		
			SEM	*0.01*	*0.01*	*0.01*	*0.01*		
14^c	GA	1	mean	-	-	-	-40105.07	13	101^{13}
			SEM	-	-	-	*3175.53*		
	SO-I	0	mean	-22552.77	-31359.50	-35517.40	-40687.10		
			SEM	*3369.06*	*3325.75*	*3281.80*	*3145.90*		
	B&B	0	mean	-	-	-50186.73	-50186.73		
			SEM	-	-	*4.47*	*4.47*		
	NOMAD	0	mean	-	-	-	-48363.03		
			SEM	-	-	-	*1197.02*		

Table 5.7: Constrained multimodal problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations.
^{c} denotes constrained problems.
$(**)$ denotes significance at $\alpha = 1\%$ for
$H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$;
$(\oplus\oplus)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\text{B\&B}}$ and
$H_1 : \mu_{\text{SO-I}} > \mu_{\text{B\&B}}$;
$(\square\square)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\text{GA}}$ and
$H_1 : \mu_{\text{SO-I}} > \mu_{\text{GA}}$.

ID	Algorithm	Number of evaluations			
	$\mathcal{A}$	100	200	300	400
	GA	$(**)$			
1^c	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD	$(**)$	$(**)$	$(**)$	$(**)$
	GA	$(**)$	$(**)$	$(**)$	$(**)$
6^c	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD	$(**)$	$(**)$	$(**)$	$(**)$
	GA	$(\square\square)$			
11^c	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD	$(\diamond)$			
	GA	$(**)$	$(**)$	$(**)$	
14^c	B&B	$(**)$	$(**)$	$(\oplus\oplus)$	$(\oplus\oplus)$
	NOMAD	$(**)$	$(**)$	$(**)$	

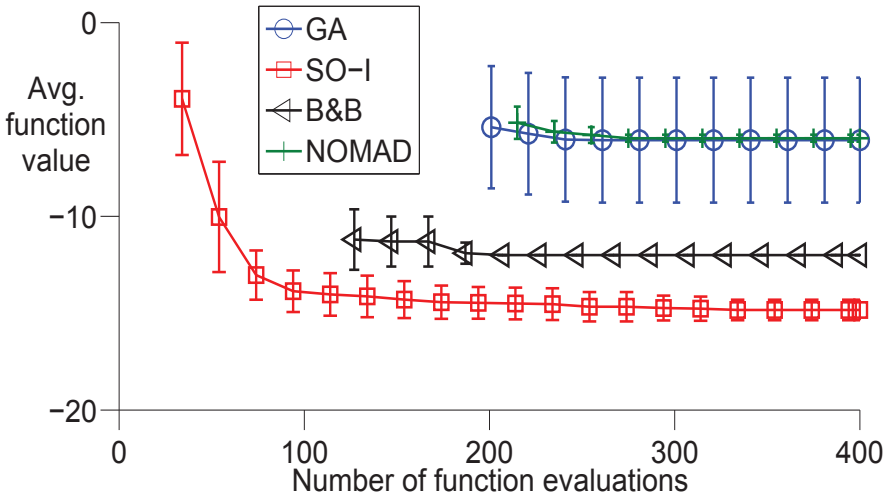


Figure 5.3: Constrained multimodal minimization problem, test problem 6. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

5.5.4 Binary Problems

The results for the two test problems with binary variables are shown in Table 5.8. One constrained and one unconstrained problem has been considered. NOMAD and SO-I perform better than GA. B&B did not find any improvement of the starting point for test problem 12 within 400 evaluations, and the mean and SEM values have been computed based on the objective function values of the initial points. B&B found the optimal solution for test problem 8, but needed more than 300 function evaluations to find a first feasible point. Moreover, the solution reported for B&B is for only *two out of 30 trials*, since the algorithm did not find any feasible solution within 400 evaluations for the other 28 trials. Also GA, SO-I and NOMAD had difficulties finding a feasible solution (see column #NF). For binary problems it was not expected that SO-I would work well at all because only two variable values are possible for each integer dimension, and the fitted surface was hence expected to be very inaccurate. The results for these two problems suggest, however, that a surrogate model based approach is as well useful when binary problems are dealt with.

The results of the hypothesis testing for problem 12 in Table 5.9 indicate that SO-I performs significantly better than all other algorithms. For test problem 8 (where NOMAD and B&B found better solutions than

SO-I) the null hypotheses $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ must be rejected at the 5% significance level and the hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\text{B\&B}}$ could not be rejected.

The development of the average objective function values over the 400 allowed evaluations for test problem 12 is illustrated in Figure 5.4 together with the standard deviations. The figure shows that SO-I, NOMAD, and GA are able to find significant improvements within 100 evaluations, whereas B&B did not find any improvements of the starting points within 400 evaluations. The relative error between the global minimum (or the best known solution) and the average objective function value found by SO-I after 400 function evaluations is about 8% for test problem 8 and zero for problem 12.

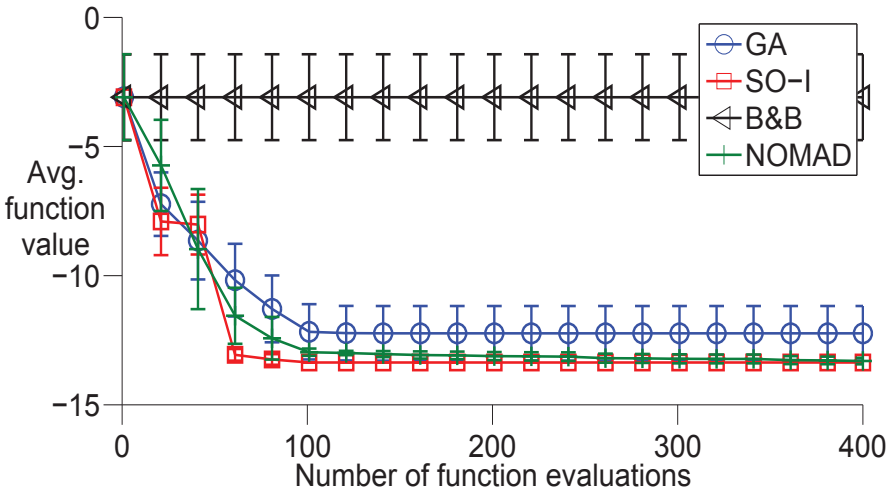


Figure 5.4: Unconstrained binary minimization problem, test problem 12. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 5.8: Binary problems. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
8^c	GA	8	mean	17.73	17.73	17.73	17.73	16	2^{16}
			SEM	*0.86*	*0.86*	*0.86*	*0.86*		
	SO-I	15	mean	15.13	14.60	14.07	14.07		
			SEM	*0.95*	*0.86*	*0.73*	*0.73*		
	B&B	28	mean	-	-	-	13.00		
			SEM	-	-	-	*0.00*		
	NOMAD	22	mean	13.00	13.00	13.00	13.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
12	GA	0	mean	-12.17	-12.23	-12.23	-12.23	20	2^{20}
			SEM	*0.20*	*0.19*	*0.19*	*0.19*		
	SO-I	0	mean	-13.36	-13.36	-13.36	-13.36		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	-3.09	-3.09	-3.09	-3.09		
			SEM	*0.30*	*0.30*	*0.30*	*0.30*		
	NOMAD	0	mean	-12.94	-13.11	-13.22	-13.31		
			SEM	*0.03*	*0.03*	*0.03*	*0.02*		

Table 5.9: Binary problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations.
^{c} denotes constrained problems.
$(*)$ denotes significance at $\alpha = 5\%$ and $(**)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$;
$(\diamond)$ denotes significance at $\alpha = 5\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ and $H_1 : \mu_{\text{SO-I}} > \mu_{\text{NOMAD}}$.

ID	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
8^c	GA	$(*)$	$(**)$	$(**)$	$(**)$
	B&B	$(**)$	$(**)$	$(**)$	
	NOMAD	$(\diamond)$	$(\diamond)$		
12	GA	$(**)$	$(**)$	$(**)$	$(**)$
	B&B	$(**)$	$(**)$	$(**)$	$(**)$
	NOMAD	$(**)$	$(**)$	$(**)$	

5.5.5 Linear Problems

The results for the linear problems are given in Table 5.10. SO-I found the optimal feasible solution for both test problems. B&B and NOMAD found the optimal feasible solution for problem 3. GA needed more than 100 function evaluations to find a first feasible point for both problems, and B&B had initial difficulties finding a feasible point only for test problem 17. NOMAD did not find any feasible solution within 400 evaluations for 8 trials of problem 17.

For the two problems with linear objective functions the hypothesis tests for differences in means in Table 5.11 show that SO-I performs significantly better than GA on both test problems. The results also show that SO-I performs significantly better than NOMAD and B&B for problem 17.

Figure 5.5 shows the objective function values of feasible points averaged over 30 trials for all algorithms for test problem 17 and their corresponding standard deviations. The offset of the graphs indicates the number of evaluations needed by each algorithm for finding a feasible solution. As can be seen, NOMAD and SO-I are most efficient with this respect. SO-I is however

able to improve the objective function values within fewer evaluations than NOMAD.

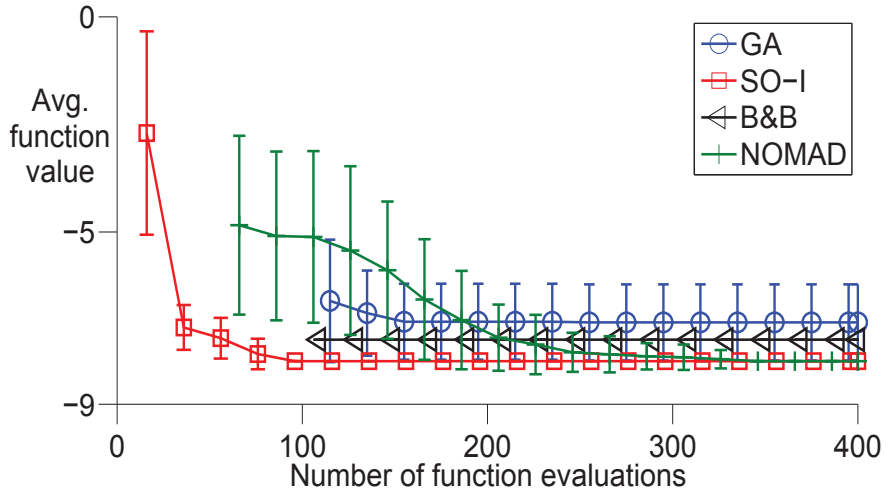


Figure 5.5: Constrained minimization problem with linear objective function, test problem 17. Objective function value of feasible points averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 5.10: Problems with linear objective function. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
3^c	GA	0	mean	-	0.72	0.72	0.72	5	5324
			SEM	-	*0.14*	*0.14*	*0.14*		
	SO-I	0	mean	0.00	0.00	0.00	0.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	0.00	0.00	0.00	0.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	NOMAD	0	mean	0.00	0.00	0.00	0.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
17^c	GA	0	mean	-	-7.08	-7.10	-7.10	5	5324
			SEM	-	*0.16*	*0.16*	*0.16*		
	SO-I	0	mean	-8.00	-8.00	-8.00	-8.00		
			SEM	*0.00*	*0.00*	*0.00*	*0.00*		
	B&B	0	mean	-	-7.50	-7.50	-7.50		
			SEM	-	*0.00*	*0.00*	*0.00*		
	NOMAD	8	mean	-5.09	-7.25	-7.91	-8.00		
			SEM	*0.42*	*0.18*	*0.06*	*0.00*		

Table 5.11: Linear problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations.
^{c} denotes constrained problems.
(**) denotes significance at $\alpha = 1\%$ for
$H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$.

ID	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
3^c	GA	(**)	(**)	(**)	(**)
	B&B				
	NOMAD				
17^c	GA	(**)	(**)	(**)	(**)
	B&B	(**)	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)	

5.5.6 Throughput Maximization Problems

The numerical results for the two throughput maximization problems are reported in Table 5.12. Note that both problems are maximization problems, and therefore large mean values are better. For both problems SO-I found the best average objective function values, but failed to find a feasible solution for three trials of problem 18. NOMAD also failed to find a feasible solution for three trials of this problem. Although GA was able to find feasible solutions for all trials of both problems, its solution quality is in general worse than that of SO-I. NOMAD had initial difficulties finding a feasible solution for problem 19 within 200 evaluations, and after a feasible solution had been found, only very slight improvements could be achieved within the next 200 evaluations.

Problem 19 is in general the same as problem 18, but has a higher dimension. This increasing dimension seems to have a negative effect on the performance of GA and NOMAD, because for the higher dimensional problem both algorithms have initial difficulties finding a first feasible point. On the other hand, the performance of SO-I improved with increasing problem dimension. SO-I was able to find feasible solutions within fewer than 100 evaluations for all 30 trials of problem 19.

B&B could not be applied to the throughput maximization problems because the evaluation of the objective function fails when the variable values are not integer. Thus, when optimizing the continuous relaxation in the tree nodes for obtaining lower bounds for the objective function value, the simulation model fails, and therefore B&B fails. This application problem shows one of the major drawbacks of applying B&B to discrete black-box optimization problems, i.e. if the black-box evaluation fails when the input variable vector contains continuous values, B&B cannot find any solution.

Since SO-I reaches the best average objective function value for both throughput maximization problems, only the hypothesis $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}$ ($H_1 : \mu_{\text{SO-I}} > \mu_{\mathcal{A}}$), $\mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, has to be tested. The results of these tests are shown in Table 5.13. Since B&B could not be used for solving these problems, the entries are set to NA (the results of all other algorithms can be considered "significantly better" whenever feasible points have been found). SO-I finds for both throughput maximization problems significantly better solutions than NOMAD. Also GA is outperformed by SO-I at the $\alpha = 1\%$ significance level for all algorithm stages of problem 18, and until 100 evaluations of problem 19.

Figure 5.6 shows the average objective function values of the algorithms for problem 18. The error bars indicate the standard deviations. Note that B&B could not be illustrated because it could not be used to solve the problem. The figure shows that SO-I is able to find feasible solutions fastest. The solution quality of SO-I is better than that of GA and NOMAD. NOMAD is also able to find feasible points, but its solution quality over the 400 function evaluations is slightly worse than that of GA.

Table 5.12: Throughput maximization problems. Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are maximization problems.
^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
18^c	GA	0	mean	6.04	7.16	7.80	8.24	5	5^{20}
			SEM	*0.18*	*0.15*	*0.16*	*0.17*		
	SO-I	3	mean	7.43	8.60	9.11	9.18		
			SEM	*0.23*	*0.17*	*0.13*	*0.11*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	3	mean	4.39	5.93	6.99	7.66		
			SEM	*0.45*	*0.36*	*0.26*	*0.20*		
19^c	GA	0	mean	-	2.39	2.82	3.10	23	23^{20}
			SEM	-	*0.16*	*0.16*	*0.17*		
	SO-I	0	mean	1.88	2.50	2.89	3.15		
			SEM	*0.11*	*0.17*	*0.19*	*0.20*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	-	-	0.87	0.89		
			SEM	-	-	*0.06*	*0.06*		

Table 5.13: Throughput maximization problems. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations. ^{c} denotes constrained problems.
(*) denotes significance at $\alpha = 5\%$ and (**) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} > \mu_{\mathcal{A}}$.

ID	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
18^c	GA	(**)	(**)	(**)	(**)
	B&B	NA	NA	NA	NA
	NOMAD		(*)	(*)	(**)
19^c	GA	(**)			
	B&B	NA	NA	NA	NA
	NOMAD	(**)	(**)	(**)	(**)

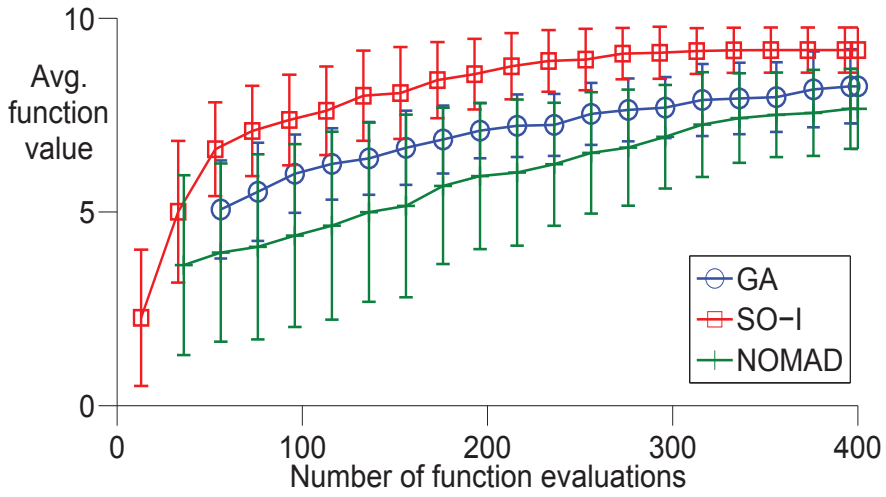


Figure 5.6: Throughput maximization problem, test problem 18. Objective function value for feasible points averaged over 30 trials (GA), and 27 trials (SO-I and NOMAD) vs. number of function evaluations; error bars illustrate standard deviations.

5.5.7 Hydropower Maximization Problems

The results for the hydropower maximization problems are given in Tables 5.14 and 5.16. Note that these are maximization problems, and

therefore large values are better. The problem instances in Table 5.14, and those in Table 5.16, differ mainly with respect to the amount of water Q_j that can be released from reservoir j, i.e. the bound on the constraint(s). Problems 20a - 20c have only one constraint (one hydropower plant), whereas problems 21a-21c have 2 constraints (two hydropower plants). Different water amounts have been used to imitate seasons with different amounts of available water in the reservoir(s). The amount of available water Q_j is lowest for problems 20a and 21a, and highest for problems 20c and 21c. For the problems with one hydropower plant (Table 5.14), SO-I performed best for two out of three problems, and NOMAD performed best for one problem. GA had initial difficulties finding a feasible solution for problems 20a and 20b. However, GA (and NOMAD) found feasible solutions for all instances of all three problems within 400 evaluations, whereas SO-I failed to do so for two instances of problem 20c. The solution quality of GA is on the other hand worse than for SO-I and NOMAD also for the problems where all algorithms were able to find feasible points for all trials.

For the hydropower problems with two reservoirs (and therefore two constraints, Table 5.16), SO-I found the best feasible solutions for all three problems. GA performs on these problems in general better than NOMAD, indicating that NOMAD has difficulties dealing with the additional constraint. Although SO-I could not find a feasible point within 400 evaluations for two instances of problem 21b and seven instances of problem 21c, the solution quality is in general better than that of GA and NOMAD, and the standard errors of the means are lower.

B&B could not be used for solving the hydropower application problems because if the integer variable values in an iteration did not allow any feasible solution with respect to the total allowed outflow Q_j, then the black-box calculation (for finding the values of the amount of water q_i that goes through each generator type i) failed. As a result, no objective function value could be computed, and thus B&B failed. With SO-I, this problem was avoided because infeasible points were given an objective function value set to the worst feasible objective function value found so far plus a penalty term derived from the constraint function values, and thus the objective function values of infeasible points were irrelevant for the remaining computations. This penalty approach could not be used for the branch and bound algorithm because the pruning decisions depend on the function value of the relaxed subproblems, and adding a penalty term as for SO-I can adversely influence these pruning decisions.

For the hydropower generation problems with one constraint the results of SO-I are significantly better than those of NOMAD and GA (and B&B) for problems 20a and 20b as shown in Table 5.15. For test problem 20c, NOMAD performs significantly better than SO-I for up to 300 evaluations. These results indicate that the performance of NOMAD improves as the upper bound on the constraint and therefore the feasible variable domain increases. For the problems with two constraints (21a, 21b, 21c), SO-I performs significantly better than all other algorithms at almost all stages (Table 5.17), indicating that NOMAD and GA have difficulties dealing with an increasing number of constraints.

Figure 5.7 shows the average objective function values of feasible points for SO-I, NOMAD, and GA for test problem 21a and their standard deviations. Note that for this problem a graph for B&B cannot be illustrated because the algorithm could not solve the problem. The figure shows that SO-I and NOMAD find feasible solutions within fewer function evaluations than GA. Although GA needed more than 150 evaluations to find a feasible point, its solution quality is at that point better than that of NOMAD. After 300 evaluations the performance of GA and NOMAD is approximately equal. SO-I has continuously a better average objective function value than NOMAD and GA.

Table 5.14: Hydropower generation maximization problems with one hydropower plant (one constraint). Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are maximization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
20a^{c}	GA	0	mean	-	718.42	735.34	735.34	5	5^{20}
			SEM	-	*9.92*	*6.75*	*6.75*		
	SO-I	0	mean	753.73	758.08	758.25	758.25		
			SEM	*1.16*	*0.17*	*0.00*	*0.00*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	482.00	732.78	744.00	744.00		
			SEM	*55.80*	*8.11*	*0.96*	*0.96*		
20b^{c}	GA	0	mean	-	1944.08	1995.50	2008.83	5	5^{20}
			SEM	-	*11.78*	*7.06*	*4.68*		
	SO-I	0	mean	1986.67	2008.21	2020.00	2020.67		
			SEM	*6.98*	*2.94*	*1.31*	*1.33*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	1823.08	1976.65	2003.46	2003.83		
			SEM	*29.59*	*12.11*	*5.07*	*5.01*		
20c^{c}	GA	0	mean	3828.25	4043.97	4087.92	4108.84	5	5^{20}
			SEM	*88.49*	*9.84*	*5.17*	*4.49*		
	SO-I	2	mean	4031.34	4090.80	4103.30	4114.60		
			SEM	*7.42*	*3.94*	*3.16*	*2.68*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	4081.83	4116.75	4125.08	4125.75		
			SEM	*10.73*	*11.67*	*12.01*	*12.06*		

Table 5.15: Hydropower generation maximization problems with one hydropower plant. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations. ^{c} denotes constrained problems. $(*)$ denotes significance at $\alpha = 5\%$ and $(**)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} > \mu_{\mathcal{A}}$; $(\diamond\diamond)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\text{NOMAD}}$ and $H_1 : \mu_{\text{SO-I}} < \mu_{\text{NOMAD}}$.

ID	Algorithm	Number of evaluations			
	$\mathcal{A}$	100	200	300	400
	GA	$(**)$	$(**)$	$(**)$	$(**)$
20a^c	B&B	NA	NA	NA	NA
	NOMAD	$(**)$	$(**)$	$(**)$	$(**)$
	GA	$(**)$	$(**)$	$(**)$	$(*)$
20b^c	B&B	NA	NA	NA	NA
	NOMAD	$(**)$		$(*)$	$(**)$
	GA	$(*)$	$(**)$	$(**)$	
20c^c	B&B	NA	NA	NA	NA
	NOMAD	$(\diamond\diamond)$	$(\diamond\diamond)$	$(\diamond\diamond)$	

Table 5.16: Hydropower generation maximization problems with two hydropower plants (two constraints). Mean objective function values (mean) over 30 trials with 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes. Standard errors of means (SEM) are given for each algorithm in italic. All problems are maximization problems. ^{c} denotes constrained problems.

ID	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.	dim.	$\lvert\Omega^D\rvert$
21a^{c}	GA	0	mean	-	1443.94	1557.61	1560.36	5	5^{20}
			SEM	-	*44.41*	*26.03*	*25.96*		
	SO-I	0	mean	1621.85	1664.81	1675.79	1677.17		
			SEM	*17.56*	*2.81*	*2.21*	*2.10*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	1002.78	1279.68	1529.53	1626.18		
			SEM	*54.98*	*37.62*	*31.95*	*19.64*		
21b^{c}	GA	0	mean	3320.73	3783.83	3914.70	4016.17	5	5^{20}
			SEM	*114.94*	*33.88*	*27.97*	*23.14*		
	SO-I	2	mean	3865.54	4004.11	4075.46	4094.54		
			SEM	*30.82*	*19.92*	*15.07*	*12.53*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	3358.83	3747.57	3834.47	3899.40		
			SEM	*72.81*	*37.72*	*32.86*	*25.57*		
21c^{c}	GA	0	mean	7739.33	8002.33	8125.33	8220.67	5	5^{20}
			SEM	*35.70*	*36.28*	*31.92*	*22.22*		
	SO-I	7	mean	8145.65	8240.43	8274.78	8295.65		
			SEM	*28.66*	*19.50*	*18.88*	*16.48*		
	B&B	30	mean	-	-	-	-		
			SEM	-	-	-	-		
	NOMAD	0	mean	7637.83	7894.67	8076.33	8122.33		
			SEM	*47.85*	*42.09*	*35.15*	*32.05*		

Table 5.17: Hydropower generation maximization problems with two hydropower plants. Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations. ^{c} denotes constrained problems. $(*)$ denotes significance at $\alpha = 5\%$ and $(**)$ denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, B\&B, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} > \mu_{\mathcal{A}}$.

ID	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
21a^c	GA	$(**)$	$(**)$	$(**)$	$(**)$
	B&B	NA	NA	NA	NA
	NOMAD	$(**)$	$(**)$	$(*)$	
21b^c	GA	$(**)$	$(**)$	$(**)$	$(**)$
	B&B	NA	NA	NA	NA
	NOMAD	$(**)$	$(**)$	$(**)$	$(**)$
21c^c	GA	$(**)$	$(**)$	$(**)$	$(**)$
	B&B	NA	NA	NA	NA
	NOMAD	$(**)$	$(**)$	$(*)$	

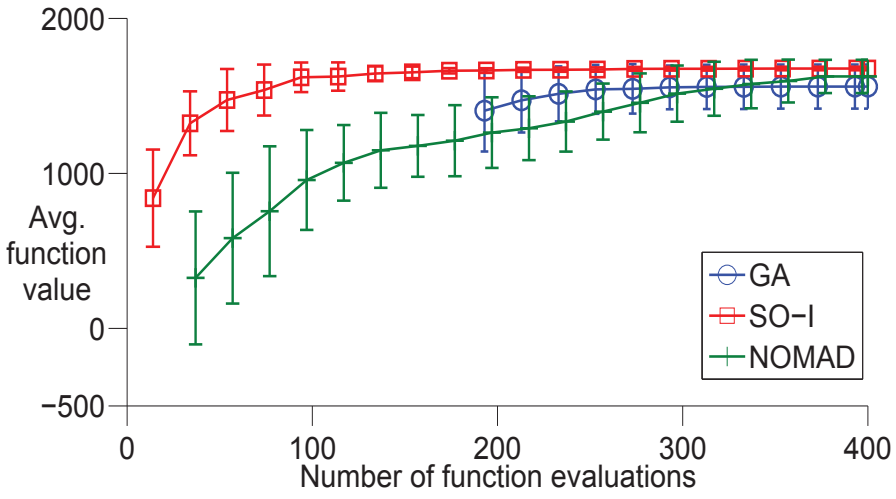


Figure 5.7: Hydropower generation maximization problem with two constraints, test problem 21a. Objective function value averaged over 30 trials vs. number of function evaluations; error bars illustrate standard deviations.

5.5.8 General Discussion

The results of the numerical experiments show that SO-I outperforms GA, NOMAD, and B&B for almost all generic and application problem instances. It was expected that B&B would outperform SO-I at least on the unimodal test problems. However, B&B required for most of these problems many more function evaluations than SO-I for achieving the same solution quality. This fact can be related to the computation of the lower bounds for the objective function value in the tree nodes of the branch and bound algorithm. Computing the lower bounds requires minimizing the objective function with some variables relaxed to take on continuous values. Even though the subproblems may be unimodal, the number of function evaluations for determining the minimum of the relaxed subproblem may become high, and thus, the total number of function evaluations increases quickly.

The application problems show that B&B should not be used when the structure of the underlying problem is unknown. None of the application problems could be solved with the branch and bound algorithm, either because the continuous relaxation of the variables causes an error in the simulation code (throughput maximization), or because infeasible variable values do not allow the algorithm to compute an objective function value (hydropower generation maximization). These examples show that branch and bound may in general encounter difficulties when dealing with black-box functions.

SO-I outperforms GA and B&B for almost all problems, i.e. also on the constrained problems, indicating that the chosen penalty approach for treating the constraints is effective. NOMAD performed better than SO-I only on four of the 25 problems, and performed for several problems better than B&B and GA. One drawback of SO-I is that it failed finding a feasible solution for a few cases more than GA. Table 5.18 summarizes for how many out of 30 trials for each constrained test problem every algorithm failed to find a feasible solution within 400 function evaluations. A dash denotes that all 30 trials were successful. The last row of the table shows the total number of failed trials summed up over all 17 constrained test problems (total number of failures out of $17 \cdot 30 = 510$ trials). As can be seen, GA failed to find feasible solutions for the fewest problems, followed by NOMAD, and SO-I. SO-I had a lower trial failure count than NOMAD (37 vs. 63 trials), but NOMAD had a lower problem failure count than SO-I (4 vs. 6 problems). B&B had a significantly worse performance

than the other three algorithms with respect to finding feasible solutions within a very restricted number of function evaluations. Comparing the results of SO-I and GA for the eight unconstrained problems shows however that SO-I is able to find better results for all unconstrained problems, indicating that the solution quality of SO-I is in general better than that of GA.

Table 5.18: Number of trials per problem where no feasible solution could be found within 400 function evaluations (constrained problems only). A dash denotes that all 30 trials were successful, and feasible solutions were found. ^{c} denotes constrained problems.

ID	GA	SO-I	B&B	NOMAD
1^c	2	-	30	30
3^c	-	-	-	-
4^c	-	8	30	-
5^c	-	-	30	-
6^c	-	-	-	-
8^c	8	15	28	22
11^c	-	-	30	-
14^c	1	-	-	-
17^c	-	-	-	8
18^c	-	3	30	3
19^c	-	-	30	-
$20a^c$	-	-	30	-
$20b^c$	-	-	30	-
$20c^c$	-	2	30	-
$21a^c$	-	-	30	-
$21b^c$	-	2	30	-
$21c^c$	-	7	30	-
Total	11	37	388	63

Finally, Table 5.19 summarizes for how many trials of every test problem each algorithm found the global minimum (or the best known solution, see Appendix D). The numbers show that SO-I was able to find the global optimum for the most trials for 20 out of the 25 examined test problems. NOMAD found the global optimum more often than SO-I for problems 10 and 11, B&B found the global optimum for test problem 14 more often, and GA found the optimal solution more often than SO-I for problems 4 and 21b.

Table 5.19: Number of trials per problem where the global minimum (or best known solution) was found within 400 function evaluations by each algorithm. A dash denotes that the global optimum was not found in any of the 30 trials. ^{c} denotes constrained problems. Best result for each problem is marked by boxes.

ID	GA	SO-I	B&B	NOMAD
1^c	28	30	-	-
2	-	1	-	-
3^c	15	30	30	30
4^c	1	-	-	1
5^c	1	11	-	-
6^c	-	27	-	-
7	30	30	30	30
8^c	9	13	2	8
9	9	30	-	30
10	-	-	-	26
11	-	-	-	1
12	4	30	-	5
13	12	30	30	30
14^c	-	1	23	-
15	-	30	-	1
16	10	30	1	30
17^c	10	30	-	22
18^c	-	3	-	-
19^c	-	1	-	-
$20a^c$	6	30	-	3
$20b^c$	12	23	-	11
$20c^c$	9	11	-	1
$21a^c$	15	25	-	15
$21b^c$	9	7	-	-
$21c^c$	8	15	-	3

5.6 Conclusions

In this chapter an extension of the SO-MI algorithm introduced in Chapter 4 to handling purely discrete computationally expensive black-box

optimization problems that may have computationally expensive constraints has been introduced. The performance of the new algorithm SO-I was numerically compared to a genetic algorithm, a branch and bound algorithm for nonlinear optimization problems, and NOMAD. While genetic algorithms and branch and bound have been widely used in the literature to solve discrete optimization problems, a study that examines the performance of NOMAD and surrogate model based algorithms such as SO-I has not been conducted.

The algorithms were compared on 17 generic test problems. Since there is no test bench for discrete global optimization problems, problems with different characteristics have been examined. Thus, problems where the continuous relaxation is multimodal or unimodal, problems with linear and nonlinear objective and constraint functions as well as binary problems have been included in the comparison. Several problems were derived from continuous global optimization test problems by imposing integrality constraints on the variables. Also problems from the collection of Mixed Integer Nonlinear Programming models [23] were used. Furthermore, the algorithms were applied to eight realizations of application problems arising from throughput and hydropower generation maximization.

The computational results show that the surrogate model based algorithm SO-I outperforms all other algorithms on almost all problems. There are only four problems for which NOMAD found significantly better solutions than SO-I, and the genetic algorithm and branch and bound were not able to find results that were significantly better for any of the totally 25 examined problems. Moreover, branch and bound was not able to solve the eight application problem instances because the objective function value could either not be calculated when some variables were continuous in the relaxed subproblems, or because infeasible variable values did not allow the simulation model to generate an objective function value. These examples show that difficulties may in general be encountered when applying the branch and bound algorithm to black-box optimization problems.

The numerical results indicate that SO-I is applicable to a much wider class of problems than the branch and bound algorithm because SO-I does not require the objective or constraint functions to be of a specific structure (e.g. linear or unimodal). Thus, using surrogate models is a computationally efficient and promising approach for solving integer optimization problems in general.

It is important to remind the reader that because the focus is on computationally expensive functions, the computational effort is measured in the number of objective function evaluations. For expensive functions that need, for example, several minutes or more per function evaluation, the objective function evaluation requires at least an order of magnitude more CPU time than the remaining calculations of the optimization algorithm steps. Hence in this situation, the number of objective function evaluations is the dominant issue. However, for unimodal and linear problems with very inexpensively computed objective function values, the SO-I algorithm would not necessarily perform as well comparatively as branch and bound because the reduction in evaluations might be outweighed by the extra computation cost for the surrogate model construction.

As more and more engineering and management problems are based on computationally expensive black-box objective and constraint functions, it is necessary to use as few function and constraint evaluations as possible for obtaining accurate solutions within an acceptable time frame. The SO-I algorithm shows great promise to achieve this goal, and thus substantial savings in computation times are possible. One drawback of SO-I is that it cannot be ensured that optimization phase 1 (finding a first feasible point) is successful. Thus, an improved strategy of finding feasible points and handling constraints in general could be developed in the future. An extension of SO-I to handle constraints directly instead of adding a penalty term to the objective function value is considered within the scope of a watershed management problem in Chapter 6.

Chapter 6

SO-Ic: Watershed Management Optimization Using A Discrete Surrogate Model Algorithm with Explicit Constraint Handling

Abstract

In this chapter a computationally expensive global optimization problem about the management of the agricultural land use in the Cannonsville reservoir watershed in upstate New York is considered. The problem has only integer variables, and an extension of SO-I is used for tackling the problem. The goal of the optimization is to retire parts of the land in a watershed at minimal cost such that constraints on the total allowable phosphorus runoff are satisfied. Computing the objective and constraint function values requires a computationally expensive simulation, and thus only a very limited number of function evaluations is possible from a computation time point of view. The SO-I algorithm is extended for treating the phosphorus constraint directly rather than with a penalty approach. A surrogate model for the constraint is used to discard infeasible-predicted candidate points from further consideration, and therefore it is more likely that the points selected for doing the expensive simulation are feasible. The algorithm, SO-Ic, is compared to a genetic algorithm, NOMAD, and the discrete dynamically dimensioned search (discrete-DDS) algorithm for three problem instances with different

limits for the total phosphorus runoff. The numerical experiments show that SO-Ic outperforms all other algorithms, and finds significantly better solutions for all problem instances.

Abbreviations and Nomenclature

BMP	Best Management Practice
CBA	Cost-Benefit Analysis
GA	Genetic algorithm
Discrete-DDS	Discrete Dynamically Dimensioned Search
HRU	Hydrologic Response Unit
NOMAD	Nonsmooth Optimization by Mesh Adaptive Direct Search
NPV	Net Present Value
SEM	Standard error of the means
SO-Ic	Surrogate Optimization - Integer constraints
SO-I	Surrogate Optimization - Integer
SWAT	Soil and Water Assessment Tool
TP	Total phosphorus
VSA	Variable Source Area
$\mathbf{u}$	Discrete decision variables, $\mathbf{u} = (u_{1,1}, u_{1,2}, \ldots, u_{1,K}, u_{2,1}, u_{2,2}, \ldots, u_{2,K})$, first index denotes land use (1=corn, 2=pasture), second index denotes subbasin
$\mathbf{u}^T$	Transpose of $\mathbf{u}$
K	Number of subbasins
k	Subbasin index
$f(\cdot)$	Objective function, see equation (6.5a)
$c(\cdot)$	Constraint function, see equation (6.5b)
$H_{\max}$	Maximal amount of allowed phosphorus runoff, see equation (6.5b)
i	Index for the land use, $i = 1$ (corn), $i = 2$ (pasture)
$s_f(\cdot)$	Response surface for the objective function
$s_p(\cdot)$	Response surface for the phosphorus constraint function
$\mathcal{S}$	Set of already evaluated points
$\boldsymbol{\chi}$	Candidate point for next sample site
n_0	Number of points in initial experimental design
$V(\cdot)$	Weighted score, see equation (3.8)

6.1 Introduction

In this Chapter the SO-I algorithm developed in Chapter 5 will be extended and applied to solving a computationally expensive optimization problem that arises in the management of the agricultural land use of a watershed in the Cannonsville reservoir in upstate New York. The goal is to convert (retire) parts of the land in a watershed at minimum cost in order to keep the total phosphorus (TP) runoff below a certain limit. The problem has only integer variables, and for computing the objective and constraint function values a computationally expensive simulation model that takes more than two minutes CPU time has to be run. SO-I is further developed with respect to the constraint handling. A response surface is used to approximate the constraint on the TP runoff in the water [48], and the prediction of the response surface for the TP constraint is used to exclude candidate points from the search [118]. The surrogate model algorithm for integer problems and with direct constraint handling is in the following referred to as SO-Ic (**S**urrogate **O**ptimization - **I**nteger **c**onstraints).

The Cannonsville reservoir and the problem of managing the agricultural land use in the watershed is described in detail in Sections 6.2 and 6.3[1], and the mathematical problem formulation is given. Section 6.4 describes the direct constraint handling of SO-Ic using a response surface to approximate the constraint. The watershed management problem has only one constraint, but the approach can be generalized to more constraints [48]. However, note that if, for example, a kriging model is used as response surface for objective and constraint functions and if the number of constraints is large, the approach of using a separate surrogate model for each constraint may become computationally expensive, because separate numerical optimizations have to be carried out for fitting the models. In those cases a penalty approach may be more efficient. The results of SO-Ic for the watershed management problem are compared to the discrete dynamically dimensioned search (discrete-DDS) algorithm [140][2], the genetic algorithm for discrete problems (Matlab versions 2011b and newer), and NOMAD. The branch and bound algorithm is not applicable for this application problem because the black-box simulation fails when variables take on continuous values as is the case when optimizing the relaxed subproblems in the tree nodes. The results of the numerical experiments are discussed in Section 6.5. Section 6.6

[1]Many thanks to Joshua Woodbury who kindly provided the verbal problem formulation, the description of the cost-benefit analysis, and the simulation model.

[2]Implementation by Joshua Woodbury.

concludes this chapter.

6.2 The Cannonsville Reservoir Watershed

The Cannonsville Reservoir located in Delaware County in upstate New York is one of the largest of New York City's drinking water reservoirs. The watershed that drains into the reservoir is approximately 1200 km^2, consisting of mostly forested and agricultural lands, of which approximately 0.5% is urban. The dominant form of agriculture is dairy, which has been shown to contribute high levels of phosphorus from runoff, particularly from corn lands fertilized with manure. Because of this, the reservoir has often experienced water quality problems due to heavy phosphorus loading. The phosphorus loads have at times exceeded the total maximum daily load (TMDL) set for the reservoir (20 μg/L [107]). Under a Memorandum of Agreement between the county, New York City, and government agencies, repeated violations of the TMDL could lead to restrictions on economic development in Delaware County. The phosphorus has caused concerns as to whether or not New York City can continually use this water without the need for a water filtration plant.

In order to compare the algorithm SO-Ic presented in this chapter to the various derivative-free methods mentioned in Section 6.1 (GA, NOMAD, discrete-DDS), a test basin is used. This test basin is a 10 subbasin subwatershed of the Cannonsville watershed which is considered to be representative of the entire watershed. The test basin is used in order to reduce the computational time required to solve the optimization problem. Corn land and pasture land in this test basin are the two most significant producers of TP runoff [139]. For this reason, corn land and pasture land are the two land use types eligible for land conversion in this study.

6.3 Watershed Land Use Management Optimization Problem

The optimization problem is to determine the optimal locations of land conversion (or land retirement) in order to reduce the amount of TP runoff within the watershed. Land conversion, sometimes referred to as land retirement, is a best management practice (BMP) employed in watersheds that drain to bodies of water that are particularly sensitive to agricultural

runoff. This particular BMP involves contracts between government agencies and local landowners to remove certain land from crop production. As compensation for retiring the land from crop production, the government agencies agree to provide some percentage of setup costs to convert the land as well as yearly rental payments and sometimes yearly maintenance payments. These payments are provided to the landowner in order to offset the loss from removing the land from crop production. The land conversion contracts are specified to last a certain number of years, often 10, 15, or 20 years. The focus of this study is on contracts with lengths of 10 years.

Since the cost of these projects can be quite high (millions of US$), the goal is to minimize the total conversion costs while keeping the TP runoff below a given threshold. These are, however, two conflicting requirements. When increasing the amount of retired land, the TP runoff decreases, but the costs increase. On the other hand, decreasing the amount of retired land leads to cost reduction, but an increased TP runoff. Thus, finding an optimal solution can become a complicated task in relatively large, agricultural watersheds such as the ones found in the Catskill region of upstate New York.

The amount of TP runoff is computed with a modified version of the Soil and Water Assessment Tool (SWAT) model 2005 [9]. The SWAT version used in this study is referred to as SWAT-VSA, which incorporates variable source area (VSA) hydrology by modifying the SWAT input files while maintaining the Curve Number method used in the standard SWAT model [44]. The SWAT-VSA model also incorporates several source code modifications shown to improve the overall fit to measured data in the Cannonsville basin. These modifications are described in [139].

The main difference between the SWAT 2005 model setup and the SWAT-VSA setup is the way in which hydrologic response units (HRUs) are defined. The HRUs in the SWAT 2005 model are defined by the coincidence of land use and soil type [103], while the HRUs in the SWAT-VSA model are defined by the coincidence of land use and wetness class. The wetness classes used in this model are determined using a topographic index, as described in [44]. These wetness classes are ranked from 1 (the driest and least likely to saturate) to 10 (the wettest and most likely to saturate). This method, combined with alterations to the available water content, allows for more accurate descriptions of VSAs, which are major sources of runoff in humid, well-vegetated regions, especially those with permeable soils underlain by a shallow restricting layer, such as upstate New York (see,

for example, [41, 42, 102]).

Since the SWAT model generates runoff and nutrient transport at the HRU scale, the HRUs are converted in the optimization. This study focuses in particular on the corn and pasture HRUs, but the method can easily be extended to other types of agricultural HRUs. Rather than using each agricultural HRU in the watershed as a decision variable (i.e. converted or not converted), the method presented here combines agricultural HRUs in each subbasin with the same land use into a single decision variable. This drastically reduces the number of decision variables, which in turn drastically decreases the computational time for each optimization.

In each subbasin, the agricultural HRUs with the same land use are combined into a single decision variable by taking advantage of the setup of the SWAT-VSA model. Since the HRUs are created by land use and wetness class, which assumes that given a certain land use, the wettest HRU will produce the most runoff and nutrient transport (see Figure 6.1), it can be assumed that in an optimization scheme, the HRU with the highest wetness class will be converted before an HRU with a lower wetness class. This facilitates the formulation of decision variables such that for a particular land use, wetness class 10 will be converted before wetness class 9, which will be converted before wetness class 8 and so on until all wetness classes are converted.

With this assumption, each decision variable, u_{ik}, for the ith land use ($i = 1$ (corn), $i = 2$ (pasture)) in the kth subbasin ($k = 1, 2, , \ldots, K$), can take on any discrete value from 1 to 11, where a value of one implies that all wetness classes are converted, a value of two implies that wetness classes 2 through 10 are converted, a value of three implies that wetness classes 3 through 10 are converted, and so on, until 10, where only wetness class 10 is converted. A value of 11 implies that no wetness classes are converted for that particular land use.

The costs associated with the land conversion are determined by using a cost-benefit analysis (CBA) [152]. CBA is a systematic method to compare the costs and the benefits of a project or an investment in order to determine whether the project is sound, i.e. whether or not the benefits outweigh the costs. The CBA used in this study is the net present value (NPV) approach. The NPV is defined as the present values of the benefits minus the costs. It requires the quantification of the costs and benefits of the project as well as a discount rate ξ that discounts the future costs and benefits to today's

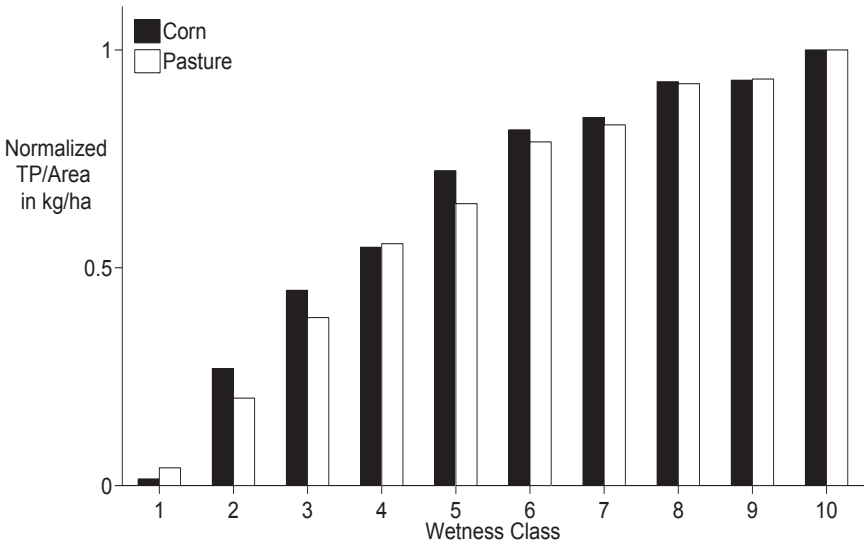


Figure 6.1: Normalized total phosphorus runoff (TP) per hectare from corn (black bars) and pasture (white bars) wetness classes in the watershed. The amount of TP runoff per hectare increases from wetness class 1 (the driest and least likely to saturate) to wetness class 10 (the wettest and most likely to saturate).

values. If the NPV is negative, then the project is too costly and should not be implemented, but if the NPV is greater than or equal to zero, the project may be undertaken.

The costs incurred by the land conversion BMP are as follows. The landowners' costs are setup costs (SUC_0), yearly maintenance costs (MC_τ), and the costs to replace the feed lost due to land conversion (CF_τ). The initial setup costs include mowing, site preparation, seeding, and fencing. The yearly maintenance cost represents the yearly upkeep of the land conversion BMPs to preserve the effectiveness. Much of this cost comes from man hours, which include the operation of machinery and general activities such as mowing and cleaning. The maintenance costs also include fuel and equipment costs. Thirdly, the cost to replace the loss of feed due to the land conversion have to be offset. In the Cannonsville watershed farmers grow corn primarily to feed their cows. If some of this corn cannot be grown, farmers must purchase feed from other sources.

The benefits in this analysis are the payments received by the landowners from the government agencies for the land conversion project. There is an initial payment per acre (INC_0) given by the government agency, and there is often a cost sharing (SUP_0) provided for the landowners. This is referred to as the setup payment, which is some fraction of the initial setup costs. Since the land is taken out of production for some number of years, the government agency provides a yearly rental payment (RP_τ) to the landowners to help offset the loss of revenue from taking the land out of production. The government agency also provides payment for some percentage of the yearly maintenance costs (MCP_τ). There is also a benefit from the cost of producing corn (CPA_τ) since the landowner no longer needs to produce corn.

The NPV approach used here is quite simple, and lends itself to many different projects, both environmental and non-environmental. The NPV approach is a way to use the time value of money to investigate the long-term viability of a project or action. The method uses the initial cost of the project along with the expected yearly cash in- and outflows along with a discount rate to discount future cash flow values to today's value. The NPV method has been used to analyze watershed management practices in many different agricultural watersheds [131, 143, 146, 150, 153, 160]. The assumption is that once the lifetime of the project is agreed upon, it must be completed.

The costs to the government agencies per kg of phosphorus reduction are determined as follows.

$$\text{NPV}_{\text{LT}} = \text{Benefits}_{\text{LT}} - \text{Costs}_{\text{LT}}, \tag{6.1}$$

where

$$\text{Benefits}_{\text{LT}} = \text{INC}_0 + \text{SUP}_0 + \sum_{\tau=1}^{\text{LT}} \frac{\text{RP}_\tau}{(1+\xi)^\tau} + \sum_{\tau=1}^{\text{LT}} \frac{\text{MCP}_\tau}{(1+\xi)^\tau} + \sum_{\tau=1}^{\text{LT}} \frac{\text{CPA}_\tau}{(1+\xi)^\tau}, \tag{6.2}$$

and

$$\text{Costs}_{\text{LT}} = \text{SUC}_0 + \sum_{\tau=1}^{\text{LT}} \frac{\text{MC}_\tau}{(1+\xi)^\tau} + \sum_{\tau=1}^{\text{LT}} \frac{\text{CF}_\tau}{(1+\xi)^\tau}, \tag{6.3}$$

where NPV_{LT} is the net present value and LT is the lifetime of the project, which is 10 years. The objective is to obtain an NPV greater than or equal to zero. One of the most important variables in the above equation is the discount rate ξ. This value can be highly variable. When the discount rate is held constant, an average value of 8% is used. This value falls within the

range of 6-10% used in other environmental projects, and is recommended by the Environmental Protection Agency [16, 91].

In order to determine the cost to the government agencies per kg of phosphorus reduction, the total cost of the project must be determined. This is done by taking the benefits to the landowners from the NPV equation, which represent the costs to the government agencies, and discounting these values to present values. The equation for the total cost, $\mathrm{TC}_{\mathrm{LT}}$, of the project to the government agencies is thus defined by

$$\mathrm{TC}_{\mathrm{LT}} = \mathrm{INC}_0 + \mathrm{SUP}_0 + \sum_{\tau=1}^{\mathrm{LT}} \frac{\mathrm{RP}_\tau}{(1+\xi)^\tau} + \sum_{\tau=1}^{\mathrm{LT}} \frac{\mathrm{MCP}_\tau}{(1+\xi)^\tau}. \tag{6.4}$$

This study uses a SWAT-VSA model with 10 subbasins, and corn and pasture being the two land types eligible for land conversion. This results in a total of 20 decision variables since each of the 10 subbasins contains both corn and pasture. The model is highly nonlinear and complex, and a computationally expensive simulation has to be run for each vector of decision variables to determine the total conversion costs (6.4) and the corresponding TP runoff. The conversion costs and TP runoff are the output of the same computationally expensive simulation, i.e. whenever the objective function is evaluated, also the amount of TP runoff is obtained. The amount of TP runoff from the watershed is taken from the watershed outlet.

Due to the computational expense related to the objective and constraint function evaluation, the goal is to find an accurate solution of the problem within as few function evaluations as possible. The SWAT-VSA model is a black-box, and therefore derivatives are not available. Moreover, computing derivatives numerically would require too many function evaluations, and computation times would in turn increase drastically. Thus, derivative-free methods are pursued in this chapter.

Mathematically, the problem can be described as follows. Denote the decision variable vector by $\mathbf{u} = (u_{1,1}, u_{1,2}, \ldots, u_{1,K}, u_{2,1}, u_{2,2}, \ldots, u_{2,K})^T$, where the first subscript denotes the land use (1=corn, 2=pasture), and the second subscript denotes the kth subbasin of the watershed. In the considered instance there are totally $K = 10$ subbasins in the watershed, but the problem can be easily extended to more subbasins. The constrained optimization problem can then be formulated as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{u}) && \text{(6.5a)} \\ \text{s.t.} \quad & c(\mathbf{u}) - H_{\max} \leq 0 && \text{(6.5b)} \\ & u_{ik} \in \{1, 2, \ldots, 11\}, i = 1, 2, k = 1, 2, \ldots, K, && \text{(6.5c)} \end{aligned}$$

where $f(\mathbf{u})$ denotes the total costs (TC_{LT} defined in equation (6.4)), and $H_{\max}$ denotes the maximal amount of allowed TP runoff, and the problem dimension is $2K$.

6.4 SO-Ic: Discrete Surrogate Model Algorithm with Direct Constraint Handling

The SO-I algorithm described in Chapter 5 has been extended to handling constraints directly rather than with a penalty approach. The basic structure of SO-I is maintained, i.e. the algorithm starts with an initial experimental design with $2(2K + 1)$ points, where $2K = 20$ ($K = 10$ subbasins) is the problem dimension. The computationally expensive simulation model is evaluated at all points in the starting design, and thus objective and constraint function values are obtained.

Based on the data, two response surfaces are used, namely one for approximating the objective function $f(\mathbf{u})$ in equation (6.5a) and another one for approximating the constraint function in equation (6.5b). The prediction of the response surface for the objective function at a point $\mathbf{u}$ will in the following be denoted by $s_f(\mathbf{u})$, and the prediction of the response surface for the TP constraint will be denoted by $s_p(\mathbf{u})$. A cubic radial basis function interpolant has been used for approximating both functions. When doing so, it is assumed that all variables are continuous in order to obtain smooth surfaces. However, when evaluating the costly simulation model, only integer points are selected, and thus every sampled point will satisfy the integrality constraints in equation (6.5c). Since constraint and objective function values are the output of the same simulation, there is always an equal number of objective and constraint function values. Thus, the matrix in equation (1.10) needs to be LU-factorized only once for computing the parameters of s_f and s_p, respectively.

Although using a surrogate model to approximate black-box constraints may be more accurate than using a penalty approach, using a surrogate model for each constraint may in general become, depending on the used

surrogate model, computationally inefficient with an increasing number of constraints and function evaluations [118]. This is, however, not a problem for the considered watershed management optimization problem because only one constraint is present.

If there is no feasible point in the initial experimental design, optimization phase 1, as described in Section 5.3, is applied, where the constraint violation function defined in equation (5.2) is minimized. In the case of this specific application, optimization phase 1 is equivalent to minimizing the constraint function using the surrogate model s_p. Once a feasible point has been found, optimization phase 2 starts. Candidate points are generated as described in Section 5.3. The response surface for the phosphorus constraint is used to discard all candidate points that are predicted infeasible, i.e. all candidate points $\boldsymbol{\chi}$ for which $s_p(\boldsymbol{\chi})$ is strictly positive. If no point is left, a new set of candidates is generated until at least one feasible-predicted point is obtained that is not contained in the set $\mathcal{S}$ of already evaluated points.

As for SO-I, the scoring criteria in equations (3.6) and (3.7) are used to determine the best candidate point at which the next computationally expensive simulation model will be evaluated. The weights for the scoring criteria have been adjusted in a cycling manner in order to repeatedly transition from a global to a local search. Numerical experiments with both constant and cycling weights for the criteria showed that the cycling weights lead to a better performance in this algorithm. The reason may be related to the constraint handling. Since no penalties are used, the response surface for the objective function does not have the large function values at the boundary of the feasible region. Infeasible-predicted points are discarded and hence the probability to sample in the infeasible region is decreased. Therefore, it is more likely to select sample points within the feasible region without continuously assigning a large weight to the response surface criterion.

After the objective and constraint function values have been obtained, the parameters of the response surfaces s_f and s_p are updated, and the algorithm iterates through generating candidate points, computing scoring criteria for feasible-predicted candidates, and parameter updating until a maximal number of allowed function evaluations has been reached. The specific steps of SO-Ic are summarized in Algorithm 7.

Algorithm 7 ***SO-Ic: Surrogate Optimization - Integer constraints***

1. *Initial experimental design. Generate an initial experimental design with* $n_0 = 2(2K+1)$ *points, where* $2K$ *denotes the problem dimension. Round the variable values to the closest integers, and in case the rank of matrix* $\mathbf{P}$ *in equation (1.11) is less than* $2K+1$*, create a new initial experimental design. Repeat until* $\text{rank}(\mathbf{P}) = 2K+1$*. Do the computationally expensive simulations at the* n_0 *selected points to obtain the objective and constraint function values. Denote the set of sampled points by* $\mathcal{S}$*.*

2. *Optimization phase 1. If there is no feasible point in the initial experimental design, minimize the constraint function in equation (6.5b) using the response surface approach until a feasible point has been found as follows.*

   (a) *Compute the parameters of the response surface* s_p *using the sample points in* $\mathcal{S}$ *and their corresponding constraint function values.*

   (b) *Generate candidates for the next computationally expensive simulation as in algorithm SO-I. Eliminate candidates that are already contained in* $\mathcal{S}$*.*

   (c) *Select the candidate with the best value for* V *in equation (3.8) as next sample site, where* s_{mix} *in equation (3.7) is replaced by* s_p*. Do the computationally expensive simulation to obtain the objective and constraint function values at the selected point.*

   (d) *Update the parameters of the response surface* s_p*.*

   (e) *Iterate through Steps 2b-2d until the first sample point with* $c(\mathbf{u}) - H_{\max} \leq 0$ *has been found or the maximum number of allowed function evaluations has been reached.*

3. *Optimization phase 2. Iterate through Steps 3a-3e until the maximum number of allowed function evaluations has been reached.*

   (a) *Compute the parameters of the response surfaces* s_f *and* s_p *using the sample points in* $\mathcal{S}$ *and their corresponding objective and constraint function values, respectively.*

   (b) *Generate candidates for the next computationally expensive simulation as in algorithm SO-I. Eliminate candidates that are already contained in* $\mathcal{S}$*.*

(c) *Use the response surface s_p to predict the constraint function values for all candidates, and discard candidates that are predicted infeasible.*

(d) *Choose the candidate point with the best value for V in equation (3.8) as next sample site, where s_{mix} in equation (3.7) is replaced by s_f.*

(e) *Add the selected point to the set $\mathcal{S}$, and do the computationally expensive simulation at the selected point to obtain constraint and objective function values.*

4. Return the best solution found.

6.5 Numerical Experiments

6.5.1 Experimental Setup

The performance of SO-Ic has been compared in the numerical experiments to three other algorithms, namely Matlab's genetic algorithm (GA), NOMAD [2, 3, 4], and the discrete dynamically dimensioned search (discrete-DDS) algorithm [140]. All four algorithms were implemented and tested in Matlab 2011b. NOMAD has been used through the interface of the OPTI Toolbox[3] because the black-box simulation model was only given as a Matlab file. NOMAD has been used with the variable neighborhood search option `VNS 0.75` as suggested by the developers in the code manual. The constraint has been treated with the progressive barrier approach (setting `PB`). The genetic algorithm used 20 individuals and stopped latest when the maximum number of allowed function evaluations of 400 was reached.

The TP constraint has been treated within the discrete-DDS framework by using a penalty approach. Since discrete-DDS does not require any penalty function parameters [140], the constraint handling can easily be done following the method outlined in [35]. As in [35], the objective function used in this study is defined in such a way that any infeasible solution (i.e. any solution that yields a TP value larger than the allowable $H_{\max}$ value) always has an objective function value that is worse than any feasible solution. The second requirement of the constraint handling method outlined in [35] is that the magnitude of the violation of an infeasible solution must be quantified

[3] http://www.i2c2.aut.ac.nz/Wiki/OPTI/

in order to compare two infeasible solutions. Thus, if denoting the penalized objective function by $f_p^{\text{discrete-DDS}}$, the value is adjusted to be

$$f_p^{\text{discrete-DDS}}(\mathbf{u}) = \begin{cases} f(\mathbf{u}) + f(\mathbf{u})\,(c(\mathbf{u}) - H_{\max}) & \text{if } c(\mathbf{u}) > H_{\max} \\ f(\mathbf{u}) & \text{otherwise} \end{cases}. \quad (6.6)$$

Three different limits for the TP content have been considered in order to examine the influence on the total costs. The objective was to minimize the total costs such that the TP runoff was reduced to 40%, 60%, and 80% of the base case. The TP content at the outlet of the watershed in the base case, i.e. without any land conversion, is 68753 kg. Thus, the three cases considered in this study are

- $H_{\max} = 55000[\,\text{kg}]$ (20% reduction)
- $H_{\max} = 41250[\,\text{kg}]$ (40% reduction)
- $H_{\max} = 27500[\,\text{kg}]$ (60% reduction).

Therefore, three optimization problems have been solved.

In order to average out the random component of the initial experimental design/the initial starting point, 20 trials[4] have been made with each algorithm, and for each trial every algorithm was given the same initial point to obtain a conclusive comparison. The same starting points have been used for each of the three TP reduction goals given above. In the case of SO-Ic and GA, the point was added to the initial experimental design and the initial generation, respectively. For discrete-DDS and NOMAD the point was used as an initial guess of the solution. Note that the initially supplied points were not necessarily feasible with respect to the TP constraint in equation (6.5b), and that a point that is feasible for a TP reduction of 20% is not necessarily feasible when a 60% reduction is required.

The 20 trials contained 16 random starting points, and the following four "special" starting points to examine their influence on the solution quality of the algorithms:

- all land is initially converted:

$$\mathbf{u}_1 = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)^T \quad (6.7)$$

[4]Due to the computation time required for each algorithm run, the number of trials had to be restricted to 20.

- no land is initially converted:

$$\mathbf{u}_2 = (11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)^T \tag{6.8}$$

- only all corn land is initially converted:

$$\mathbf{u}_3 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)^T \tag{6.9}$$

- only all pasture land is initially converted:

$$\mathbf{u}_4 = (11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T. \tag{6.10}$$

6.5.2 Numerical Results

The results of the numerical experiments for all algorithms are summarized in Tables 6.1-6.5. The first column of Table 6.1 shows the TP reduction goals ("TP-reduction"). The column "#NF" indicates for how many of the 20 trials an algorithm was not able to find a feasible solution within 400 function evaluations. When computing the means and standard error of the means (SEM), only the successful trials (trials in which feasible solutions were found) are used. The columns "100 eval.", "200 eval.", "300 eval.", and "400 eval." show the objective function values for *feasible* points averaged over the number of successful trials for each algorithm after 100, 200, 300, and 400 evaluations, respectively, and in italic font the corresponding SEM values are reported. The provided numbers are all for feasible points only, and the best results are marked by boxes. A dash as in the case for NOMAD for the goal of 60% TP reduction denotes that the algorithm was not able to find a feasible solution within the given number of function evaluations.

The results in Table 6.1 show that SO-Ic finds for each TP reduction goal lower mean objective function values than all other algorithms. Also the SEM values for SO-Ic are the lowest, indicating that the performance of SO-Ic is least influenced by the choice of the starting points. Column #NF shows that SO-Ic and GA were able to find feasible solutions for all 20 trials of each problem instance. NOMAD failed to find feasible points for one trial of the 20% and 40% reduction goals, and for four trials of the 60% reduction goal. Discrete-DDS only failed finding a feasible solution for one trial with the 60% reduction goal. Although NOMAD performed comparatively well for several of the integer test problems in Chapter 5, its performance on this specific application is the worst among all algorithms

when comparing the values after 200 and more evaluations. This indicates that NOMAD may be stuck in local optima for too long, and is thus not able to find feasible objective function value improvements efficiently. Moreover, for the 60% reduction goal where the feasible variable domain was smallest, NOMAD had considerable difficulties finding feasible solutions.

Table 6.2 contains information about hypothesis tests for differences of means between SO-Ic and the other three algorithms. The table shows that, except for discrete-DDS for the 40% TP reduction goal, the mean values of SO-Ic are lower than those of all other algorithms at the $\alpha = 1\%$ and $\alpha = 5\%$ significance level. For NOMAD in the 60% case, the mean of SO-Ic is significantly better because SO-Ic was able to find feasible solutions within fewer than 100 evaluations, whereas NOMAD needed more than 300 evaluations to find feasible points.

Figures 6.2-6.4 show the average objective function value for feasible points for all three cases of desired TP reduction. The error bars illustrate the standard deviations. The offset of the graphs indicates the number of function evaluations needed by each algorithm at most for finding a first feasible point. As can be seen, when the goal is to reduce the TP value for only 20% (Figure 6.2), all algorithms find feasible solutions within very few function evaluations. However, as the constraint becomes tighter and the TP reduction goal increases, all algorithms need more evaluations for finding a feasible point, which can be seen especially in the case of NOMAD in Figures 6.3 and 6.4.

Tables 6.3-6.5 show the results of the four algorithms for each individual trial when using the special starting points defined in equations (6.7)-(6.10). The results show that SO-Ic finds the best solutions for all four special starting points when the goals are 20% and 60% TP reduction. When the goal is to reduce the TP content in the water for 40%, NOMAD finds better solutions than all other algorithms if initially either all corn (equation (6.9)) or all pasture (equation (6.10)) is converted. If initially either all land (equation (6.7)) or no land at all (equation (6.8)) is converted, SO-Ic performs better than the other algorithms only for the first 100 evaluations, but is then outperformed by discrete-DDS. However, NOMAD and discrete-DDS

Table 6.1: Mean objective function values (mean) over 20 trials after 100, 200, 300, and 400 function evaluations for different TP reduction goals. Best result of all algorithms is marked by boxes. Standard errors of means (*SEM*) are given for each algorithm in italic. A dash denotes that no feasible point has been found within the given number of function evaluations for all 20 trials. The numbers are for feasible points only.

TP-reduction	Algorithm	#NF	Statistic	100 eval.	200 eval.	300 eval.	400 eval.
20%	SO-Ic	0	mean	347721	324640	313399	307860
			SEM	*6251*	*3233*	*2095*	*2376*
	GA	0	mean	3729065	2184150	1181006	565003
			SEM	*208443*	*145291*	*110449*	*54181*
	discrete-DDS	0	mean	1652401	560383	377660	333284
			SEM	*131696*	*68028*	*29976*	*11835*
	NOMAD	1	mean	3206996	2860810	2695970	2070492
			SEM	*176736*	*158215*	*143530*	*174394*
40%	SO-Ic	0	mean	920939	897387	877884	868013
			SEM	*3053*	*2752*	*2462*	*1405*
	GA	0	mean	4178503	2485771	1500878	1007773
			SEM	*195680*	*192413*	*153017*	*101269*
	discrete-DDS	0	mean	2441009	1216343	1075360	1049251
			SEM	*157964*	*124094*	*116897*	*109633*
	NOMAD	1	mean	3928959	3517309	3350436	3040368
			SEM	*274077*	*291237*	*306324*	*300723*
60%	SO-Ic	0	mean	3865907	3533223	3454931	3436388
			SEM	*47126*	*31133*	*7867*	*7822*
	GA	0	mean	8082981	6757972	5903389	5438066
			SEM	*125269*	*96735*	*82225*	*116165*
	discrete-DDS	1	mean	7538805	5951460	5462802	5404192
			SEM	*216020*	*168696*	*132447*	*140458*
	NOMAD	4	mean	-	-	-	6157491
			SEM	-	-	-	*230750*

Table 6.2: Hypothesis testing for differences in means (μ) after 100, 200, 300, and 400 function evaluations.
($*$) denotes significance at $\alpha = 5\%$ and ($**$) denotes significance at $\alpha = 1\%$ for $H_0 : \mu_{\text{SO-I}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{\text{GA, discrete-DDS, NOMAD}\}$, and $H_1 : \mu_{\text{SO-I}} < \mu_{\mathcal{A}}$.

TP-reduction	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
20%	GA	($**$)	($**$)	($**$)	($**$)
	discrete-DDS	($**$)	($**$)	($*$)	($*$)
	NOMAD	($**$)	($**$)	($**$)	($**$)
40%	GA	($**$)	($**$)	($**$)	($*$)
	discrete-DDS	($**$)	($**$)		
	NOMAD	($**$)	($**$)	($**$)	($**$)
60%	GA	($**$)	($**$)	($**$)	($**$)
	discrete-DDS	($**$)	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)	($**$)

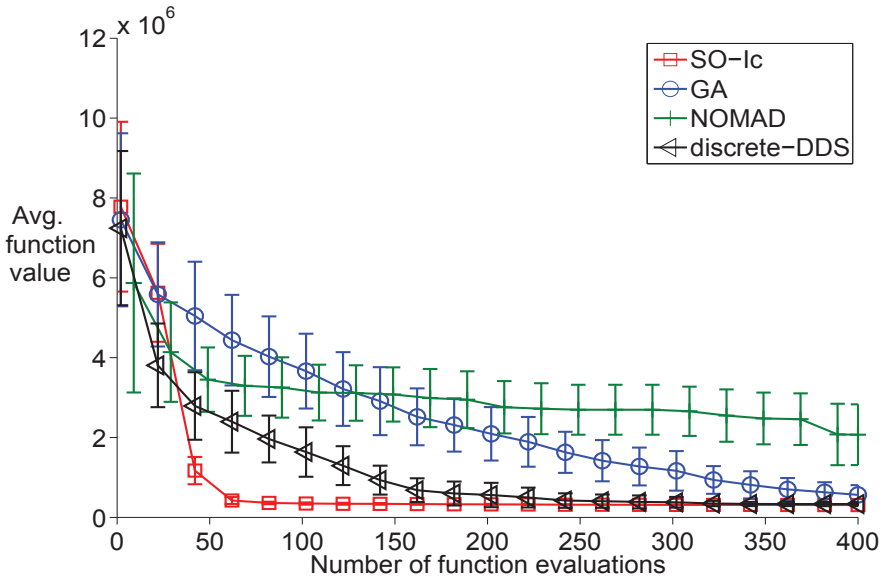


Figure 6.2: 20% TP reduction goal. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

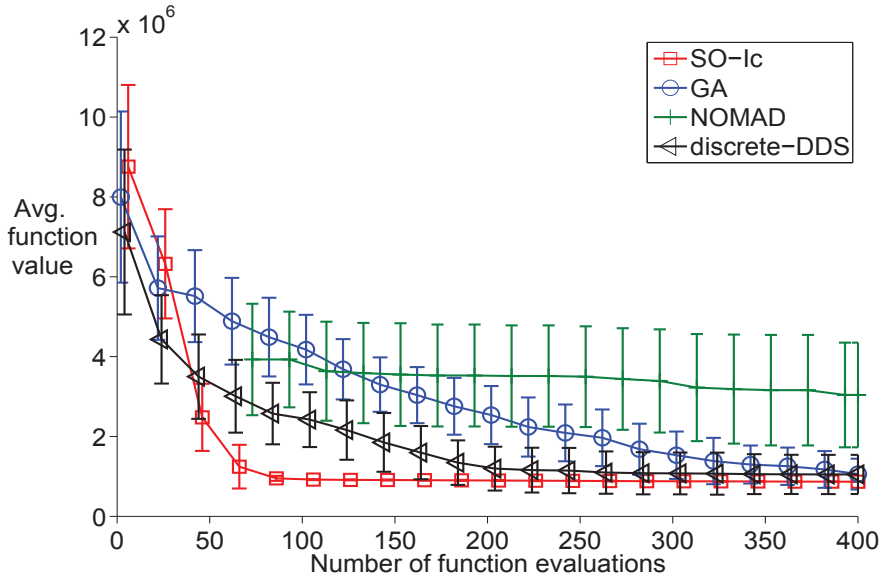


Figure 6.3: 40% TP reduction goal. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

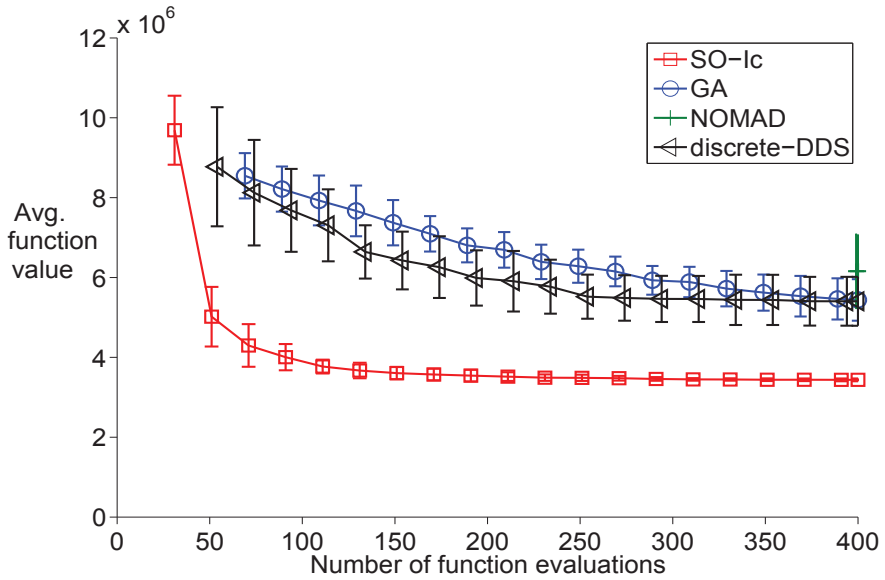


Figure 6.4: 60% TP reduction goal. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

failed to find a feasible solution when initially no land is converted. If no land is converted, the TP content in the water is at the base case level, and thus infeasible for all TP reduction goals. While NOMAD fails for all three TP reduction goals to find a feasible solution starting from this special point, discrete-DDS only fails when the goal is to reduce the TP for 60%.

Out of the four special cases, NOMAD finds its best result for all TP reduction goals always for the staring point $\mathbf{u}_3$, i.e. when only all corn land is initially converted. GA on the other hand reaches its best results always for starting point $\mathbf{u}_1$, i.e. when all land is initially converted. For SO-Ic and discrete-DDS different special points lead to the best results, and thus the performance of these algorithms seems to be less dependent on the starting point. The standard deviation of the solutions found by the algorithms over the four trials with special starting points also showed that SO-Ic was least influenced by the selected starting point and had for all three TP reduction goals the lowest standard deviations. Discrete-DDS had the second lowest standard deviations, and GA and NOMAD had the highest values. Thus, it can be concluded that SO-Ic is the most robust algorithm.

As may be expected, the higher the TP reduction goal is, the higher the conversion costs are. For SO-Ic the costs increase for a factor of about three when increasing the TP reduction from 20% to 40%, whereas increasing the TP reduction from 20% to 60% increases the costs for a factor of more than 10. For NOMAD the differences were not as large and a TP reduction of 60% incurred only about three times the costs as when the goal was a 20% reduction. Thus, NOMAD does not seem to be able to exploit the fact of the larger feasible variable domain when only a 20% TP reduction is required, and fails to search more globally for other regions where significant objective function value improvements could be possible.

Table 6.3: 20% TP reduction goal. Objective function value for special starting points after 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes.

Special point	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
$\mathbf{u}_1$	SO-Ic	399730	352501	316065	302228
	GA	3684024	2084127	1044427	510003
	discrete-DDS	1298502	694858	526301	306023
	NOMAD	3590038	3264167	3264167	2737815
$\mathbf{u}_2$	SO-Ic	335814	325412	323157	316553
	GA	4253431	2978701	2176577	863408
	discrete-DDS	1411758	366221	352882	328211
	NOMAD	-	-	-	-
$\mathbf{u}_3$	SO-Ic	341083	323906	320708	295381
	GA	1001770	1001770	1001770	1001770
	discrete-DDS	628997	329446	316031	310588
	NOMAD	500183	500183	500183	379082
$\mathbf{u}_4$	SO-Ic	337155	323965	317161	317161
	GA	5315847	3180148	2248653	748825
	discrete-DDS	2054291	350379	328438	328438
	NOMAD	3339389	2865536	2865536	2865536

Table 6.4: 40% TP reduction goal. Objective function value for special starting points after 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes.

Special point	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
$\mathbf{u}_1$	SO-Ic	914087	899438	872083	860200
	GA	4270298	2861688	1345235	901597
	discrete-DDS	2360111	708985	678755	666777
	NOMAD	5031686	5031686	4886143	4886143
$\mathbf{u}_2$	SO-Ic	911707	881573	877131	863953
	GA	4177358	2886836	1485337	1093569
	discrete-DDS	2532821	741110	717372	637772
	NOMAD	-	-	-	-
$\mathbf{u}_3$	SO-Ic	935508	891428	873887	872603
	GA	1001770	1001770	1001770	1001770
	discrete-DDS	868606	725432	725432	725432
	NOMAD	731754	712725	707632	700103
$\mathbf{u}_4$	SO-Ic	899283	890115	875105	861557
	GA	4668567	2919728	1735963	1559272
	discrete-DDS	2759582	2429047	2315654	1976526
	NOMAD	832326	832326	832326	832326

Furthermore, Figures 6.5-6.8 show the average objective function values for feasible points for each single algorithm with the three TP reduction goals. As can be seen, the differences of the average objective function values between the 20% and the 40% reduction goals are much lower for all algorithms than the differences between the 40% and the 60% reduction goals. This indicates, that increasing the TP reduction goal from 20% to 40% can be done at a much lower additional cost than increasing the reduction goal to 60%.

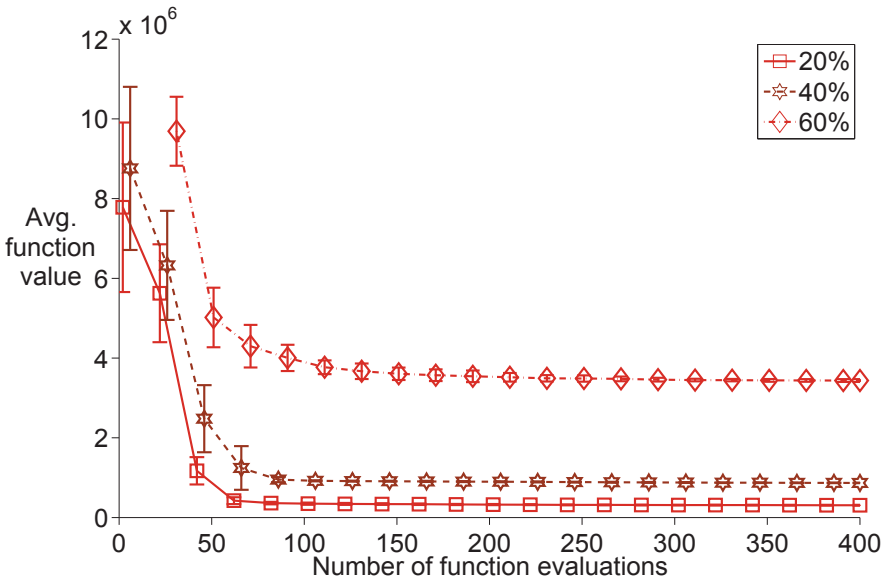


Figure 6.5: SO-Ic performance for 20%, 40%, and 60% TP reduction. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

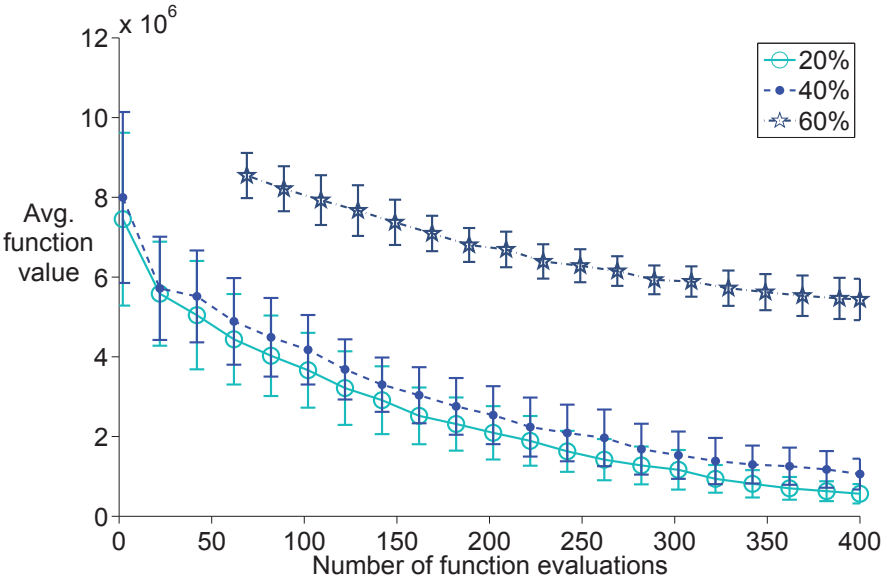


Figure 6.6: GA performance for 20%, 40%, and 60% TP reduction. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

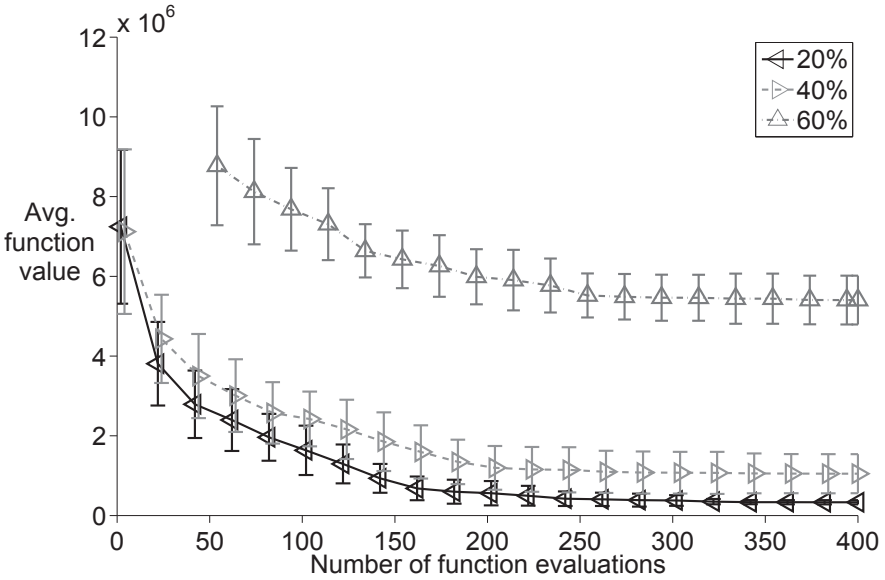


Figure 6.7: Discrete-DDS performance for 20%, 40%, and 60% TP reduction. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

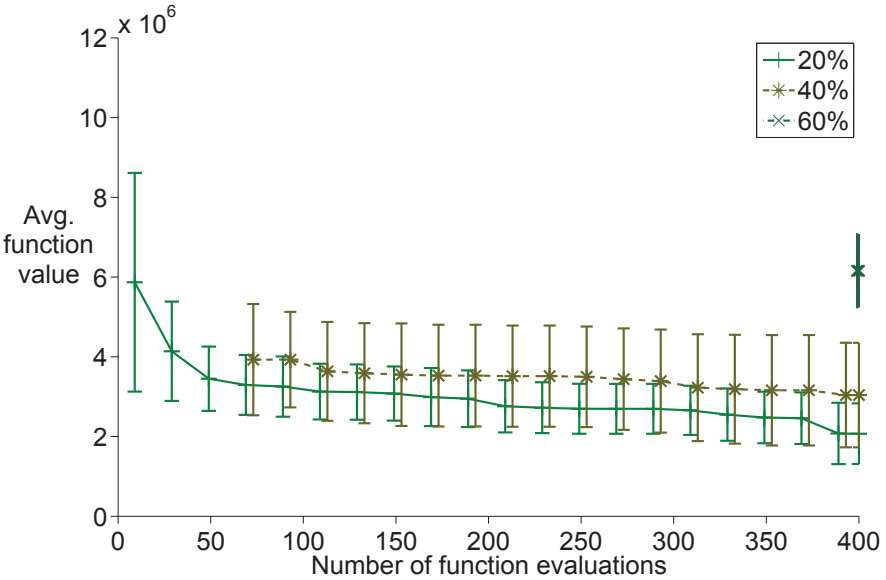


Figure 6.8: NOMAD performance for 20%, 40%, and 60% TP reduction. Objective function value averaged over 20 trials vs. number of function evaluations; error bars illustrate standard deviations.

Table 6.5: 60% TP reduction goal. Objective function value for special starting points after 100, 200, 300, and 400 function evaluations. Best result of all algorithms is marked by boxes.

Special point	Algorithm $\mathcal{A}$	Number of evaluations 100	200	300	400
$\mathbf{u}_1$	SO-Ic	3986540	3481243	3477478	3469863
	GA	7637850	6131424	5197862	4640246
	discrete-DDS	8606150	5674884	5479539	5477404
	NOMAD	6264847	6141209	6141209	6141209
$\mathbf{u}_2$	SO-Ic	3863481	3460186	3456523	3456231
	GA	7458287	7027093	5745856	5368368
	discrete-DDS	-	-	-	-
	NOMAD	-	-	-	-
$\mathbf{u}_3$	SO-Ic	3764394	3407824	3406378	3400207
	GA	7893233	6487336	5666424	4811001
	discrete-DDS	7919924	7909680	5937146	5932442
	NOMAD	4889745	4831762	4483758	4483758
$\mathbf{u}_4$	SO-Ic	3593804	3461652	3409704	3398922
	GA	8252913	6464985	6071684	5495053
	discrete-DDS	6918792	6187244	5946740	5388905
	NOMAD	6242599	5751215	5751215	5751215

6.6 Conclusions

In this chapter the surrogate model algorithm SO-I for computationally expensive global optimization problems with integrality constraints for all variables has been further developed for directly treating constraints. The algorithm SO-Ic uses a surrogate model instead of a penalty approach for handling the constraints. The algorithm has been specifically developed and tested on a computationally expensive watershed optimization problem where parts of the land of the Cannonsville reservoir watershed in upstate New York have to be retired in order to keep the TP runoff below a given threshold. The goal is to retire parts of the land such that the incurred costs are minimal. The watershed management application has only one constraint, and thus two surrogate models are used in SO-Ic, namely one for

the objective and one for the constraint function.

The basic structure of SO-Ic is the same as for SO-I. An initial experimental design is generated, and in optimization phase one a first feasible point is determined by minimizing the constraint violation function (in this specific application the actual constraint is minimized). In the second optimization phase candidate points are generated by randomly selecting points from the variable domain, and by perturbing the best feasible point found so far. The surrogate model for the constraint is then used to predict the constraint function values of the candidate points, and points predicted infeasible (positive predicted constraint function values) are discarded. Thus, it is more likely that the selected point for the next function evaluation will be feasible.

The performance of SO-Ic has been compared to NOMAD, a genetic algorithm (GA), and the discrete dynamically dimensioned search (discrete-DDS) algorithm for three problem instances with different upper bounds for the allowable amount of TP runoff in the water. The numerical results show that SO-Ic outperforms all other algorithms for all three problem instances. This fact is supported by hypothesis tests for differences in means between the algorithms. NOMAD had for a few trials of all three problem instances difficulties finding feasible solutions, and its performance was in general worse than that of GA and discrete-DDS.

A comparison of the solution quality of the algorithms for four special starting points showed furthermore that SO-Ic is the most robust algorithm and its solution quality is least influenced by the starting point. GA and NOMAD had the highest standard deviations, and their solution quality seems thus more dependent on the starting point. Moreover, the results for the three upper bounds on the allowable TP runoff showed that the cost increase is much less when increasing the TP reduction goal from 20% to 40%, than when increasing the reduction goal from 40% to 60%, which indicates that the total costs do not behave linearly with the TP reduction goal.

SO-Ic can in general easily be extended for problems with more than one constraint [118]. However, as the number of constraints increases also the computation time of the algorithm may, depending on the surrogate model, increase, and thus the approach of using a surrogate model for each individual constraint may become inefficient. A study of different approaches for handling constraints exceeds the scope of this thesis, and is left for future research.

Chapter 7

Concluding Remarks

Abstract

In this thesis surrogate model algorithms for computationally expensive global optimization problems with continuous, mixed-integer, and integer variables have been developed. At this point it shall be briefly summarized in how far the questions posed in Chapter 1 could be answered. Moreover, questions for future research that arose during this study are outlined.

Abbreviations

EGO	Efficient global optimization
Discrete-DDS	Discrete Dynamically Dimensioned Search
NOMAD	Nonsmooth Optimization by Mesh Adaptive Direct Search
RBF	Radial basis function
SO-I	Surrogate Optimization - Integer
SO-Ic	Surrogate Optimization - Integer constraints
SO-M	Surrogate Optimization - Mixture
SO-M-c	Surrogate Optimization - Mixture - candidate sampling
SO-M-s	Surrogate Optimization - Mixture -surface minimum
SO-MI	Surrogate Optimization - Mixed Integer

7.1 Which surrogate model should be used for a given problem?

Various studies in the literature have shown that one surrogate model does not perform best for all kinds of problems, and when dealing with black-box problems it is impossible to know a priori which surrogate model will perform best. The question of which surrogate model is best suited for a given problem has been addressed by developing mixture surrogate model algorithms. The algorithms SO-M, SO-M-c, and SO-M-s for continuous deterministic global optimization problems described in Chapters 2 and 3 use Dempster-Shafer theory to determine the influence of individual models on the mixture. One major issue of importance that has been encountered during this study is the strategy of iteratively selecting new sample sites for doing the computationally expensive simulation. The three algorithms use different sampling strategies, and their influence on the solution quality has been examined in numerical studies.

It was found that mixtures that contain a cubic RBF model perform in general very well, whereas algorithms that use only a polynomial regression model should be avoided. Mixture surrogate models should be the choice whenever it is a priori unknown which surrogate model will perform best because they automatically prevent selecting the worst single model. Moreover, mixture models perform often even better than the single models contributing to the mixture. With respect to choosing the "most successful" sampling strategy, it was found that the problem dimension plays an important role. SO-M-s performed on average very well on problems of at most six dimensions, whereas SO-M-c proves successful for higher-dimensional problems.

7.2 Can surrogate model algorithms be used for black-box problems with integrality constraints?

Yes, they can! And their performance is very promising. It was shown that the algorithms SO-MI and SO-I developed in Chapters 4 and 5 are able to find accurate solutions to mixed-integer and purely integer black-box global optimization problems, respectively. The few surrogate model algorithms that have so far been developed in the literature for these types of problems either use mixed-integer subsolvers (which may become a computational burden as the problem dimension increases), or they are made for binary

problems only.

SO-I and SO-MI extend this research area and are applicable to higher dimensional problems where integer variables may have a wide range of values rather than only binary values. The two algorithms use a random sampling strategy, and are thus very efficient with respect to computation times when determining the sample points in each iteration. Constraints are treated with a penalty approach, and numerical experiments showed that both algorithms perform significantly better than NOMAD, genetic algorithms, and branch and bound, where the latter two algorithms are commonly used options when dealing with integer and mixed-integer problems.

It was not expected that branch and bound would perform very well on black-box problems where the algebraic problem description cannot be exploited for using reformulation or convexification strategies. However, freely available codes that can be used to solve black-box integer and mixed-integer optimization problems are very scarce. The numerical studies confirmed that branch and bound is not suitable at all for black-box problems. Firstly, if the objective function is multimodal, the lower bounds computed by branch and bound are not necessarily valid, and pruning decisions may thus be wrong. Secondly, and more importantly, many black-box problems do not allow integer variables to take on continuous values as is the case when optimizing relaxed subproblems in the tree nodes. In these cases objective function values cannot be computed, and branch and bound fails finding any solution.

Although successful for optimization problems with cheap objective function values, genetic algorithms did not prove suitable for computationally expensive problems where only a limited number of function evaluations can be allowed because they need in general many function evaluations (number of generation $\times$ number of individuals) to find good solutions.

NOMAD, on the other hand, proved more successful than branch and bound and the genetic algorithm. NOMAD is applicable for mixed-integer and integer black-box optimization problems, but its performance for these problem types has so far not widely been studied in the literature.

It has to be stressed that the algorithms were compared with respect to their performance after an equal number of function evaluations, and that their computation time can in general be neglected when evaluating the objective and constraint functions takes from several minutes to several hours or even

days. If the objective and constraint functions are computationally cheap, other algorithms may be more efficient since in these cases the driving computational expense comes from the optimization algorithm rather than the function evaluations.

7.3 How can the agricultural land use of an upstate New York watershed be managed to reduce the phosphorus runoff at minimal cost?

The SO-Ic algorithm in Chapter 6 tries to answer this question by using a separate surrogate model for the objective function and the total phosphorus constraint [48, 118]. The examined real-world watershed management problem has integrality constraints imposed on all variables. The problem had only a single constraint whose value is obtained by the same computationally expensive simulation as the objective function value. Thus, building a response surface for the constraint is computationally cheap. Numerical experiments for the watershed management problem showed that the constraint handling in SO-Ic is very successful, and the algorithm performs significantly better than the genetic algorithm, NOMAD, and discrete-DDS.

The constraint handling with the response surface in SO-Ic is very efficient for this special problem because there was only one constraint. However, if the number of constraints is large, then using a surrogate model for each constraint may become computationally more demanding [118] (depending on the type of surrogate model used), and in these cases penalty methods may be more efficient.

7.4 Future research directions and open questions

During this study various new questions arose that could be addressed in the future. Depending on the number of allowed function evaluations, or the maximum allowed CPU time, one major question for all developed algorithms is when to switch from the local to the global search, and vice

versa. Of course, once a promising region of the variable domain has been found, it should be imperative to thoroughly explore this region (local search). After how many function evaluations can the region however be considered throughly explored? Should a predefined number of consecutive unsuccessful function evaluations be used, or a limit on the percentage of improvement between consecutive iterations? These questions have partially been addressed in the algorithm SO-M, where so-called densely sampled regions have been defined for some variables to guide the local and global search phases. Other criteria, or maybe distance measures may be developed in order to better address this topic.

During the study of mixture surrogate models one major question was which surrogate models should be included in the mixture. Other choices of radial basis functions (linear, thin-plate spline, multiquadric, etc.) have not been examined in the study. Similarly, only the kriging model with Gaussian correlation function has been included. In future studies it may be examined if, for example, mixtures containing different radial basis function types, or kriging models with different correlation functions (Gaussian, exponential, cubic, etc.) would be able to improve the solution quality. Of course, an increasing number of models in the mixture will increase the necessity of several processors to speed up the computations by parallelization, and the question of how many models can and should be used while still obtaining a computationally efficient algorithm that is able to find good solutions becomes important.

Moreover, for future comparisons, derivative-free trust-region methods such as DFO [28, 29], UOBYQA [111], NEWUOA [112], or BOBYQA [113] could be compared to the developed algorithms. Although these algorithms are local optimizers, they may be competitive also on global problems especially when the computational budget is very restricted. Also the influence of using a multistart gradient-based method or a derivative-free method with a convergence guarantee for optimizing the auxiliary problems in Gutmann's RBF method, EGO, or SO-M-s, could be further examined with respect to its influence on the results.

Regarding integer and mixed-integer black-box global optimization problems one major difficulty was to find suitable algorithms and test problems for comparing the performance of SO-MI and SO-I. When the work of the corresponding chapters was submitted for journal publication, the anonymous referees of the papers objected that a comparison to genetic algorithms and branch and bound may not be fair because these algorithms are not really

designed to work under such strict limitations on the number of function evaluations, but they did not suggest any alternatives. Except for NOMAD, there are no freely-available codes that are able to deal with integer and mixed-integer black-box problems with computationally expensive objective and constraint functions. The TOMLAB Matlab toolbox is able to deal with problems that have *cheap* constraints, but there are many parameters that have to be adjusted by the user, and thus fair comparisons are not possible either. Moreover, the software is commercial, and tests on high-dimensional unconstrained mixed-integer black-box problems showed that the algorithm becomes itself a computational burden due to the optimization subproblems that have to be solved for determining the sample points in each iteration.

As briefly outlined before, algorithms such as EGO [74], Gutmann's RBF method [57], or Regis' AQUARS method [124] may straightforwardly be adapted for integer and mixed-integer problems by replacing the optimization subroutine for solving the auxiliary problems by a MINLP solver. This may be efficient for low-dimensional problems, but may (just as in TOMLAB) become a computational burden with increasing problem dimensions. On the other hand, instead of using a MINLP subsolver, stochastic methods such as the candidate point approach may be similarly effective and save computation time. These algorithm extensions could be considered in the future.

A second issue criticized by the anonymous referees of the chapter papers are the test problems. Due to the lack of algorithms for integer and mixed-integer black-box problems, there is also no "widely used" suit of test problems (as was, for example, the Dixon and Szegö [39] test bench for continuous global optimization problems) for evaluating the efficiency of the algorithms. The goal when selecting test problems for SO-I and SO-MI was to have a wide range of dimensions and a variety of characteristics and application problems in order to examine the general applicability of the algorithms. All results have been presented and no "unfavorable" cases have been omitted. The author agrees with the anonymous referees that a set of test problems for computationally expensive integer and mixed-integer global optimization problems should be developed in the future. These test problems do not necessarily have to be computationally expensive, but should have a wide variety of characteristics that are often encountered in real world application problems, such as multimodality, nonlinear and linear constraints, very steep minima, binary and non-binary variables, flat areas where several points have the same objective function values, etc., and whose global optima are known. This topic has also been addressed by Balaprakash *et al.* [15]. The test problems selected for SO-I and SO-MI were

chosen such that problems of different dimensions and with the mentioned characteristics were contained in the test set.

Various methods for treating computationally expensive black-box constraints should also be examined in the future in order to find efficient and effective methods. The penalty approach used in SO-I and SO-MI worked well, but can certainly be improved. Using a separate surrogate model for each constraint may be more accurate, and parallel implementations may be useful. Methods that use classifiers for determining feasible and infeasible regions of the variable domain have been considered by the author, and preliminary numerical experiments show promise. Moreover, if initially no feasible point is known, the algorithms should be able to find feasible points. Optimization phase 1 in SO-I could therefore further be improved in that respect.

In this thesis algorithms for deterministic objective functions have been developed. An extension of the algorithms for problems with noisy objective and constraint functions also constitutes a wide research area. It is expected that the developed concepts can be adapted for such problems, but the used surrogate models should probably be changed because interpolating models for noisy objective and constraint functions may not be useful. The mixture models may be successful for noisy problems since they are able to distinguish between "good" and "bad" models, and they are able to combine interpolating and noninterpolating models. Also the weights for the single surrogate models in the mixture may be changed such that they are depending on the variable $\mathbf{x}$, i.e.

$$s_{\text{mix}}(\mathbf{x}) = \sum_{r \in \mathcal{M}} w_r(\mathbf{x}) s_r(\mathbf{x}), \quad \text{where} \sum_{r \in \mathcal{M}} w_r(\mathbf{x}) = 1, \tag{7.1}$$

because some surrogate models may have a better fit in certain regions of the variable domain than in others. A challenge here is to determine these regions of the variable domain, which may become difficult for high-dimensional problems. Also it is expected that, similarly as for the continuous optimization problems, mixture models will work well for integer and mixed-integer problems. Moreover, sampling strategies based on minimizing the response surface or some other response surface related auxiliary problem may not be suitable for noisy problems, and also here random sampling strategies may be more successful. This topic is, however, left for future research.

Appendix A

Test Problems Chapter 2

A.1 Test Problem 1: Branin

This is a two-dimensional problem with $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. The function is defined by

$$f(x_1, x_2) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10. \quad \text{(A.1)}$$

The function has three global minima with the optimum value $f^* = 0.3979$ at the locations $(x_1^*, x_2^*) = (-\pi, 12.275)$, $(x_1^*, x_2^*) = (\pi, 2.275)$, and $(x_1^*, x_2^*) = (3\pi, 2.475)$. Other local optima do not exist.

A.2 Test Problem 2: Camelback

The variables of this bivariate function are constrained to $x_1 \in [-3, 3]$ and $x_2 \in [-2, 2]$, and the function is defined by

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3} \right) x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2. \quad \text{(A.2)}$$

The function has two global minima, one at $(x_1^*, x_2^*) = (-0.0898, 0.7127)$, and the other at $(x_1^*, x_2^*) = (0.0898, -0.7127)$, respectively, where it reaches the optimal value $f^* = -1.0316$. Other local optima do not exist.

A.3 Test Problem 3: Goldstein-Price Function

The variable bounds in this problem are $x_1, x_2 \in [-2, 2]$, and the function is defined by

$$f(x_1, x_2) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 31x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]. \tag{A.3}$$

The function has its global minimum at $(x_1^*, x_2^*) = (0, -1)$ and attains there the value $f^* = 3$. In addition, the function has several local minima.

A.4 Test Problems 4 and 5: Hartmann Functions

The class of Hartmann functions is defined by

$$f(\mathbf{x}) = -\sum_{\iota=1}^{m} c_\iota \exp\left\{-\sum_{i=1}^{k} a_{\iota i}(x_i - p_{\iota i})^2\right\}, \tag{A.4}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ and $x_i \in [0, 1], \forall i$. The value for m has been set to four. The parameters for the three-dimensional problem are given in Table A.1.

Table A.1: Parameter settings for three-dimensional Hartmann function.

ι	$a_{\iota i}$			c_ι	$p_{\iota i}$		
1	3.0	10	30	1.0	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3.0	10	30	3.0	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.0382	0.5743	0.8828

The three-dimensional Hartmann function attains its global optimum $f^* = -3.8628$ at the point $(x_1^*, x_2^*, x_3^*) = (0.1146, 0.5556, 0.8525)$. The function has four local minima with function values $-c_\iota$, $\iota = 1, \ldots, 4$.

The parameters of the six-dimensional problem are given in Table A.2.

Table A.2: Parameters for six-dimensional Hartmann function.

ι	$a_{\iota i}$						c_ι	$p_{\iota i}$					
1	10.0	3.0	17.0	3.5	1.7	8.0	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10.0	17.0	0.1	8.0	14.0	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3.0	3.5	1.7	10.0	17.0	8.0	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	17.0	8.0	0.05	10.0	0.1	14.0	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

The six-dimensional Hartmann function reaches its global minimum $f^* = -3.3224$ at $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*) = (0.2017, 0.1500, 0.4769, 0.2753, 0.3117, 0.6573)$ and it has four local optima with function values $-c_\iota$, $\iota = 1, \ldots, 4$.

A.5 Test Problem 6: Shekel

The general definition of the Shekel functions is

$$f(\mathbf{x}) = \sum_{\iota=1}^{m} \frac{1}{c_\iota + \sum_{i=1}^{k} (x_i - a_{i\iota})^2} \tag{A.5}$$

and the variables are bounded to $\mathbf{x} \in [0, 10]$. The parameters are

$$\mathbf{A} = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix} \text{ and } \mathbf{c} = \frac{1}{10} (1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T .$$

The Shekel functions have a very steep global minimum at $x_i = 4, \forall i$, and in the considered test problem there are 10 local minima.

Appendix B

Test Problems Chapter 3

B.1 Test Problem 1: Branin

This is a two-dimensional problem with $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. The function is defined by

$$f(x_1, x_2) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10. \quad \text{(B.1)}$$

The function has three global minima with the optimum value $f^* = 0.3979$ at the locations $(x_1^*, x_2^*) = (-\pi, 12.275)$, $(x_1^*, x_2^*) = (\pi, 2.275)$, and $(x_1^*, x_2^*) = (3\pi, 2.475)$. Other local optima do not exist.

B.2 Test Problems 2 and 3: Hartmann Functions

The class of Hartmann functions is defined by

$$f(\mathbf{x}) = -\sum_{\iota=1}^{m} c_\iota \exp\left\{ -\sum_{i=1}^{k} a_{\iota i}(x_i - p_{\iota i})^2 \right\}, \quad \text{(B.2)}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ and $x_i \in [0, 1], \forall i$. The value for m has been set to four. The parameters for the three-dimensional problem are given in Table B.1.

Table B.1: Parameter settings for three-dimensional Hartmann function.

ι	$a_{\iota i}$			c_ι	$p_{\iota i}$		
1	3.0	10	30	1.0	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3.0	10	30	3.0	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.0382	0.5743	0.8828

The three-dimensional Hartmann function attains its global optimum $f^* = -3.8628$ at the point $(x_1^*, x_2^*, x_3^*) = (0.1146, 0.5556, 0.8525)$. The function has four local minima with function values $-c_\iota$, $\iota = 1, \ldots, 4$.

The parameters of the six-dimensional problem are given in Table B.2.

Table B.2: Parameters for six-dimensional Hartmann function.

ι	$a_{\iota i}$						c_ι	$p_{\iota i}$					
1	10.0	3.0	17.0	3.5	1.7	8.0	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10.0	17.0	0.1	8.0	14.0	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3.0	3.5	1.7	10.0	17.0	8.0	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	17.0	8.0	0.05	10.0	0.1	14.0	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

The six-dimensional Hartmann function reaches its global minimum $f^* = -3.3224$ at $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*) = (0.2017, 0.1500, 0.4769, 0.2753, 0.3117, 0.6573)$, and it has four local optima with function values $-c_\iota$, $\iota = 1, \ldots, 4$.

B.3 Test Problems 4-6: Shekel

The general definition of the Shekel functions is

$$f(\mathbf{x}) = \sum_{\iota=1}^{m} \frac{1}{c_\iota + \sum_{i=1}^{k} (x_i - a_{i\iota})^2}, \tag{B.3}$$

and the variables are bounded to $\mathbf{x} \in [0, 10]$. The parameters are

$$\mathbf{A} = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix} \text{ and } \mathbf{c} = \frac{1}{10} (1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T .$$

The Shekel functions have a very steep global minimum at $x_i = 4, \forall i$. In the considered test problem there are 5, 7, and 10 local minima, respectively.

B.4 Test Problem 7: Ackley

The 15-dimensional Ackley function [5] has one global minimum at $x_i = 0$, $i = 1\ldots, 15$, where it attains the value $f^* = -22.72$. The function has several local minima and is defined as

$$f(\mathbf{x}) = -20\exp\left\{-0.2\sqrt{\frac{\sum_{i=1}^{15} x_i^2}{15}}\right\} - \exp\left\{\frac{\sum_{i=1}^{15}\cos(2\pi x_i)}{15}\right\}, \qquad \text{(B.4)}$$

and $x_i \in [-15, 30]\ \forall i$.

B.5 Test Problem 8: Schoen

The Schoen test functions [128] are in general defined as follows.

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{\eta} f_i \prod_{j\neq i} \|\mathbf{x} - \mathbf{z}_j\|_2^{\alpha_j}}{\sum_{i=1}^{\eta} \prod_{j\neq i} \|\mathbf{x} - \mathbf{z}_j\|_2^{\alpha_j}}, \qquad \text{(B.5)}$$

where $\mathbf{x} \in [0,1]^k$, $\eta \geq 0$, $\mathbf{z}_j \in [0,1]^k, \forall j = 1, \ldots, \eta$, $f_i \in \mathbb{R}, \forall i = 1, \ldots, \eta$, and $\alpha_j \in \mathbb{R}^+, \forall j = 1, \ldots, \eta$, and where $\|\cdot\|_2$ denotes the Euclidean norm. The best function value found for $k = 17$ is $f^* = 23.03$. For the Schoen functions it holds that

1. $f(\mathbf{z}_i) = f_i, i = 1, \ldots, \eta$,
2. $\min_{1\leq i\leq\eta} f_i \leq f(\mathbf{x}) \leq \max_{1\leq i\leq\eta} f_i$ for all $\mathbf{x} \in [0,1]^k$,
3. $\lim_{\mathbf{x}\to\mathbf{z}_j} \nabla f(\mathbf{x}) = 0$ for all $j = 1, \ldots, \eta$.

B.6 Test Problem 9: Levy

The 20-dimensional Levy function [83] has one global minimum at $x_i = 1$, $i = 1\ldots, 20$, where it attains the value $f^* = 19$. The function has several local minima and is defined as

$$f(\mathbf{y}) = \sin^2\{\pi y_1\} + \sum_{i=1}^{19} y_i^2\left(1 + 10\sin^2\{\pi y_i\}\right) + (y_{20} - 1)^2\left(1 + \sin^2\{2\pi y_{20}\}\right), \qquad \text{(B.6)}$$

where $y_i = 1 + (x_i - 1)/4$, and $x_i \in [-10, 10]\ \forall i$.

B.7 Test Problem 10: Powell

The 24-dimensional Powell function [109] has one global minimum at $x_{4i} = 1, x_{4i-1} = 0, x_{4i-2} = -1, x_{4i-3} = 3,\ i = 1\ldots,6$, where it attains the value $f^* = 0$. The function has no local minima and is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{6} \left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4 \tag{B.7}$$

and $x_i \in [-4, 5]\ \forall i$.

B.8 Test Problem 11: Michalewicz

The 25-dimensional Michalewicz function [93] has one global minimum for which the best objective function value found is $f^* = -16.49$. The function has 25! local minima and is defined as

$$f(\mathbf{x}) = -\sum_{i=1}^{25} \sin(x_i) \left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{20} \tag{B.8}$$

and $x_i \in [0, \pi]\ \forall i$.

B.9 Test Problem 12: Sphere

The 27-dimensional Sphere (the first function of De Jong's test set) function [34] has one global minimum at $x_i = 0$, $i = 1, \ldots, 27$, where the objective function value is $f^* = 0$. The function is convex and has no local minima, and is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{27} x_i^2 \tag{B.9}$$

and $x_i \in [-5.12, 5.12]\ \forall i$.

B.10 Test Problem 13: Rastrigin

The 30-dimensional Rastrigin function [97, 142] has one global minimum at $x_i = 0$, $i = 1, \ldots, 30$, where the objective function value is $f^* = -30$. The

function has several local minima. The variables are $x_i \in [-1, 3]\ \forall i$, and the function is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{30} x_i^2 - \cos(2\pi x_i). \tag{B.10}$$

Appendix C

Test Problems Chapter 4

C.1 Test Problem 1

This test problem is a modification of the problem used by [17]. The problem has several modes.

$$\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{u}) = -x_5 x_6 x_7 && \text{(C.1a)}\\
\text{s.t.} \quad & -u_1 - u_2 - u_3 + 1 \leq 0 && \text{(C.1b)}\\
& -u_4 - x_1 - x_2 + 1 \leq 0 && \text{(C.1c)}\\
& -x_3 - x_4 + 1 \leq 0 && \text{(C.1d)}\\
& 3u_1 + u_2 + 2u_3 + 3u_4 + 2x_1 + x_2 + 3x_3 + 2x_4 \leq 10 && \text{(C.1e)}\\
& \log(0.1)u_1 + \log(0.2)u_2 + \log(0.15)u_3 - \log(1 - x_5) \leq 0 && \text{(C.1f)}\\
& \log(0.05)u_4 + \log(0.2)x_1 + \log(0.15)x_2 - \log(1 - x_6) \leq 0 && \text{(C.1g)}\\
& \log(0.02)x_3 + \log(0.06)x_4 - \log(1 - x_7) \leq 0 && \text{(C.1h)}\\
& u_{i_2} \in \{0,1\},\ i_2 = 1,\ldots,4 && \text{(C.1i)}\\
& x_{i_1} \in [0,1],\ i_1 = 1,\ldots,4, && \text{(C.1j)}\\
& x_5 \in [0,0.997], x_6 \in [0,0.9985], x_7 \in [0,0.9988] && \text{(C.1k)}
\end{aligned}$$

The objective function value at the global optimum is -0.94347.

C.2 Test Problem 2

This is an 8-dimensional convex test problem. The problem is unimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = 3.1u_1^2 + 7.6u_2^2 + 6.9u_3^2 + 0.004u_4^2 + 19x_1^2 + 3x_2^2 + x_3^2 + 4x_4^2 \tag{C.2a}$$

$$\text{s.t.} \quad u_{i_2} \in \{-10, 9, \ldots, 9, 10\}\,,\ i_2 = 1, \ldots, 4 \tag{C.2b}$$

$$x_{i_1} \in [-10, 9, \ldots, 9, 10]\,,\ i_1 = 1, \ldots, 4 \tag{C.2c}$$

The objective function value at the global optimum is 0.

C.3 Test Problem 3

This is an altered version of the problem ex1221 from MINLPLib [23] (see http://www.gamsworld.org/minlp/minlplib/ex1221.htm) where equality constraints have been replaced by $\leq$ constraints, and integrality constraints have been imposed on some variables. The problem seems to be unimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = 2u_1 + 3u_2 + 1.5x_1 + 2x_2 - 0.5x_3 \tag{C.3a}$$

$$\text{s.t.} \quad u_1^2 + x_1 \leq 1.25 \tag{C.3b}$$

$$u_2^{1.5} + 1.5x_2 \leq 3 \tag{C.3c}$$

$$u_1 + x_1 \leq 1.6 \tag{C.3d}$$

$$1.333u_2 + x_2 \leq 3 \tag{C.3e}$$

$$-x_1 - x_2 + x_3 \leq 0 \tag{C.3f}$$

$$u_1, u_2 \in \{0, 1, \ldots, 10\}\,,\ x_1 \in [0, 10],\ x_2, x_3 \in [0, 1] \tag{C.3g}$$

The best known solution has the objective function value 0 at the point $[0, 0, 0, 0, 0]^T$.

C.4 Test Problem 4

This test problem goes back to [46]. The problem seems to be multimodal when the discrete variable is fixed to the value 0.

$$\min \quad f(\mathbf{x},u) = 5(x_1 - 0.2)^2 + 0.8 - 0.7u \tag{C.4a}$$

$$\text{s.t.} \quad -\exp\{x_1 - 0.2\} - x_2 \leq 0 \tag{C.4b}$$

$$x_2 + 1.1u + 1 \leq 0 \tag{C.4c}$$

$$x_1 - 1.2u \leq 0 \tag{C.4d}$$

$$u \in \{0, 1\},\ x_1 \in [0.2, 1]\,,\ x_2 \in [-2.22554, -1] \tag{C.4e}$$

The global optimum has the objective function value 1.07654.

C.5 Test Problem 5

This test problem is an alteration of the test problem G2 [77], where integrality constraints have been added for several variables. The problem dimension is $k = 25$. The problem has several modes.

$$\max \quad f(\mathbf{x}, \mathbf{u}) = \left| \frac{\sum_{i_2=1}^{6} \cos^4(u_{i_2}) + \sum_{i_1=1}^{19} \cos^4(x_{i_1}) - 2\prod_{i_2=1}^{6} \cos^2(u_{i_2}) \prod_{i_1=1}^{19} \cos^2(x_{i_1})}{\sqrt{\sum_{i_2=1}^{6} i_2 u_{i_2}^2 + \sum_{i_1=1}^{19} i_1 x_{i_1}^2}} \right| \tag{C.5a}$$

$$\text{s.t.} \quad \prod_{i_2=1}^{6} u_{i_2} \prod_{i_1=1}^{19} x_{i_1} \geq 0.75 \tag{C.5b}$$

$$\sum_{i_2=1}^{6} u_{i_2} + \sum_{i_1=1}^{19} x_{i_1} \leq 7.5 \cdot k \tag{C.5c}$$

$$u_{i_2} \in \{0, 1, 2, \ldots, 10\}, \; i_2 = 1, 2, \ldots, 6 \tag{C.5d}$$

$$x_{i_1} \in [0, 10], \; i_1 = 1, 2, \ldots, 19 \tag{C.5e}$$

The best known objective function value is 0.4022.

C.6 Test Problem 6

This test problem is an alteration of the test problem G4 [77], where integrality constraints have been added for several variables. The problem is convex and seems to have a flat area where several points achieve the same objective function value.

$$\min \quad f(\mathbf{x}, \mathbf{u}) = 5.3578547x_1^2 + 0.8356891u_1x_3 + 37.293239u_1 - 40792.141 \tag{C.6a}$$

$$\text{s.t.} \quad 0 \leq 85.334407 + 0.0056858u_2x_3 + 0.0006262u_1x_2 - 0.0022053x_1x_3 \leq 92 \tag{C.6b}$$

$$90 \leq 80.51249 + 0.0071317u_2x_3 + 0.0029955u_1u_2 + 0.0021813x_1^2 \leq 110 \tag{C.6c}$$

$$20 \leq 9.300961 + 0.0047026x_1x_3 + 0.0012547u_1x_1 + 0.0019085x_1x_2 \leq 25 \tag{C.6d}$$

$$u_1 \in \{78, 79, \ldots, 102\}, \; u_2 \in \{33, 34, \ldots, 45\}, \tag{C.6e}$$

$$x_{i_1} \in [27, 45], \; i_1 = 1, \ldots, 3 \tag{C.6f}$$

The objective function value at the global optimum is -30665.5.

C.7 Test Problem 7

This two-dimensional test problem is an alteration of the test problem G6 [77], where integrality constraints have been added for one variable. The problem seems to be multimodal.

$$\begin{aligned}
\min \quad & f(x,u) = (u-10)^3 + (x-20)^3 && \text{(C.7a)}\\
\text{s.t.} \quad & (u-5)^2 + (x-5)^2 - 100 \geq 0 && \text{(C.7b)}\\
& -(u-6)^2 - (x-5)^2 + 82.81 \geq 0 && \text{(C.7c)}\\
& u \in \{13, 14 \ldots, , 100\}\,, \; x \in [0, 100] && \text{(C.7d)}
\end{aligned}$$

The best known objective function value is -4241.96.

C.8 Test Problem 8

This test problem is an alteration of the test problem G9 [77], where integrality constraints have been added for some variables. The problem seems to be unimodal when the integers are fixed.

$$\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{u}) = (u_1-10)^2 + 5(u_2-12)^2 + u_3^4 + 3(x_1-11)^2 + && \text{(C.8a)}\\
& \qquad 10x_2^6 + 7x_3^2 + x_4^4 - 4x_3x_4 - 10x_3 - 8x_4 && \text{(C.8b)}\\
\text{s.t.} \quad & 2u_1^2 + 3u_2^4 + u_3 + 4x_1^2 + 5x_2 \leq 127 && \text{(C.8c)}\\
& 7u_1 + 3u_2 + 10u_3^2 + x_1 - x_2 \leq 282 && \text{(C.8d)}\\
& 23u_1 + u_2^2 + 6x_3^2 - 8x_4 \leq 196 && \text{(C.8e)}\\
& 4u_1^2 + u_2^2 - 3u_1u_2 + 2u_3^2 + 5x_3 - 11x_4 \leq 0 && \text{(C.8f)}\\
& u_{i_2} \in \{-10, -9 \ldots, , 10\}\,, \; i_2 = 1,2,3 && \text{(C.8g)}\\
& x_{i_1} \in [-10, 10]\,, \; i_1 = 1,2,3,4 && \text{(C.8h)}
\end{aligned}$$

The best known objective function value is 686.34.

C.9 Test Problem 9

This is an altered version of the problem ex1221 from MINLPLib [23] (see http://www.gamsworld.org/minlp/minlplib/ex1221.htm) where integrality

constraints have been imposed on some variables, and the nonlinear constraints have been left out in order to obtain a purely linear test problem. The problem seems to be unimodal.

$$\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{u}) = 2u_1 + 3u_2 + 1.5u_3 + 2x_1 - 0.5x_2 && \text{(C.9a)}\\
\text{s.t.} \quad & u_1 + u_3 \leq 1.6 && \text{(C.9b)}\\
& 1.333u_2 + x_1 \leq 3 && \text{(C.9c)}\\
& -u_3 - x_1 + x_2 \leq 0 && \text{(C.9d)}\\
& u_1, u_2, u_3 \in \{0, 1, \ldots, 10\}\,,\ x_1, x_2 \in [0, 1] && \text{(C.9e)}
\end{aligned}$$

The best known solution has the objective function value 0 at the point $[0, 0, 0, 0, 0]^T$.

C.10 Test Problem 10

This test problem can be found at http://www.aridolan.com/ga/gaa/MultiVarMin.html. Integrality constraints have been added for some variables. The test problem is multimodal.

$$\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{u}) = u_1 \sin(u_1) + 1.7u_2 \sin(u_1) - 1.5x_1 - && \text{(C.10a)}\\
& \qquad 0.1x_2 \cos(x_2 + x_3 - u_1) + 0.2x_3^2 - u_2 - 1 && \text{(C.10b)}\\
& u_1, u_2 \in \{-100, -99, \ldots, 100\} && \text{(C.10c)}\\
& x_1, x_2, x_3 \in [-100, 100] && \text{(C.10d)}
\end{aligned}$$

The best known solution has the objective function value -529.07.

C.11 Test Problem 11

This is the problem nvs09 from MINLPLib [23] (see http://www.gamsworld.org/minlp/minlplib/nvs09.htm). In the original problem formulation all variables are discrete. The problem has been

altered such that several variables may assume continuous values. The problem is unimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = \sum_{i_2=1}^{5} \log(u_{i_2}-2)^2 + \sum_{i_1=1}^{5} \log(x_{i_1}-2)^2 + \sum_{i_2=1}^{5} \log(10-u_{i_2})^2 + \tag{C.11a}$$

$$\sum_{i_1=1}^{5} \log(10-x_{i_1})^2 - \prod_{i_2=1}^{5} u_{i_2}^{0.2} \prod_{i_1=1}^{5} x_{i_1}^{0.2} \tag{C.11b}$$

$$u_{i_2} \in \{3,4\ldots,9\}\,,\ i_2 = 1,\ldots,5,\ x_{i_1} \in [3,9]\,,\ i_1 = 1,\ldots,5 \tag{C.11c}$$

The function value at the global optimum is -43.13. The global optimum of the continuous relaxation of the problem has integer variable values.

C.12 Test Problem 12

This is an altered version of the problem nvs09 from MINLPLib [23] (see http://www.gamsworld.org/minlp/minlplib/nvs09.htm). In the original problem formulation all variables are discrete. The problem has been altered such that several variables may assume continuous values. In addition, the variable domains have been increased and the objective function has been adjusted accordingly. The problem is unimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = \sum_{i_2=1}^{5} \log(u_{i_2}-2)^2 + \sum_{i_1=1}^{5} \log(x_{i_1}-2)^2 + \sum_{i_2=1}^{5} \log(100-u_{i_2})^2 + \tag{C.12a}$$

$$\sum_{i_1=1}^{5} \log(100-x_{i_1})^2 - \prod_{i_2=1}^{5} u_{i_2}^{0.2} \prod_{i_1=1}^{5} x_{i_1}^{0.2} \tag{C.12b}$$

$$u_{i_2} \in \{3,4\ldots,99\}\,,\ i_2 = 1,\ldots,5,\ x_{i_1} \in [3,99]\,,\ i_1 = 1,\ldots,5 \tag{C.12c}$$

The best known objective function value is -9591.72. The global optimum of the continuous relaxation of the problem has integer variable values.

C.13 Test Problem 13

This is the 12-dimensional Rastrigin function [142] where integrality constraints have been imposed on several variables. The problem is multimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = \sum_{i_2=1}^{5} \left(u_{i_2}^2 - \cos(2\pi u_{i_2})\right) + \sum_{i_1=1}^{7} \left(x_{i_1}^2 - \cos(2\pi x_{i_1})\right) \tag{C.13a}$$

$$\text{s.t. } u_{i_2} \in \{-1,0,1,2,3\}\,,\ i_2 = 1,\ldots,5, \tag{C.13b}$$

$$x_{i_1} \in [-1,3]\,,\ i_1 = 1,\ldots,7 \tag{C.13c}$$

The objective function value at the global optimum is -12.

C.14 Test Problem 14

This is an altered version of the 12-dimensional Rastrigin function [142] where integrality constraints have been imposed on several variables. Here the variable domains have been increased. The problem is multimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = \sum_{i_2=1}^{5} \left(u_{i_2}^2 - \cos(2\pi u_{i_2})\right) + \sum_{i_1=1}^{7} \left(x_{i_1}^2 - \cos(2\pi x_{i_1})\right) \tag{C.14a}$$

$$\text{s.t. } u_{i_2} \in \{-10,0,1,2,30\}\,,\ i_2 = 1,\ldots,5, \tag{C.14b}$$

$$x_{i_1} \in [-10,30]\,,\ i_1 = 1,\ldots,7 \tag{C.14c}$$

The objective function value at the global optimum is -12.

C.15 Test Problem 15

This is the 30-dimensional Rastrigin function [142] where integrality constraints have been imposed on several variables. The problem is multimodal.

$$\min \quad f(\mathbf{x},\mathbf{u}) = \sum_{i_2=1}^{10} u_{i_2}^2 - \cos(2\pi u_{i_2}) + \sum_{i_1=1}^{20} x_{i_1}^2 - \cos(2\pi x_{i_1}) \tag{C.15a}$$

$$\text{s.t. } u_{i_2} \in \{-1,0,1,2,3\}\,,\ i_2 = 1,\ldots,10, \tag{C.15b}$$

$$x_{i_1} \in [-1,3]\,,\ i_1 = 1,\ldots,20 \tag{C.15c}$$

The objective function value at the global optimum is -30.

C.16 Test Problem 16

This test problem has been introduced by Yuan *et al.* [159]. The problem seems to be multimodal and has several points that take on the same function value.

$$\begin{aligned}
\min\ & f(\mathbf{x},\mathbf{u}) = (x_1-1)^2 + (x_2-2)^2 + (x_3-1)^2 - \log(x_4+1) + \\
& \qquad (x_5-1)^2 + (x_6-2)^2 + (x_7-3)^2 && \text{(C.16a)}\\
\text{s.t. } & u_1+u_2+u_3+x_5+x_6+x_7 \le 5 && \text{(C.16b)}\\
& x_3^2+x_5^2+x_6^2+x_7^2 \le 5.5 && \text{(C.16c)}\\
& u_1+x_5 \le 1.2 && \text{(C.16d)}\\
& u_2+x_6 \le 1.8 && \text{(C.16e)}\\
& u_3+x_7 \le 2.5 && \text{(C.16f)}\\
& u_4+x_5 \le 1.2 && \text{(C.16g)}\\
& x_2^2+x_6^2 \le 1.64 && \text{(C.16h)}\\
& x_3^2+x_7^2 \le 4.25 && \text{(C.16i)}\\
& x_2^2+x_7^2 \le 4.64 && \text{(C.16j)}\\
& x_1-u_1 \le 0 && \text{(C.16k)}\\
& x_2-u_2 \le 0 && \text{(C.16l)}\\
& x_3-u_3 \le 0 && \text{(C.16m)}\\
& x_4-u_4 \le 0 && \text{(C.16n)}\\
& u_{i_2} \in \{0,1\},\ i_2 = 1,\ldots,4 && \text{(C.16o)}\\
& x_{i_1} \in [0,1]\,,\ i_1 = 1,\ldots,4 && \text{(C.16p)}\\
& x_{i_1} \in [0,10]\,,\ i_1 = 5,6,7 && \text{(C.16q)}
\end{aligned}$$

The globally optimal solution has the objective function value 4.5796.

C.17 Test Problem 17: Eleven Element Truss

The planar truss shown in Figure C.1 is considered, which is an alteration of the example in [25, Chapter 4]. The structure has 11 elements (an element is a truss member located between two nodes). There are three load cases, $F_1 = 280\,\text{kN}$ acting on nodes 1 and 5, $F_2 = 210\,\text{kN}$ acting on node 3, and $F_3 = 310\,\text{kN}$ acting on node 7. The coordinates of nodes 1, 3, 5, and 7 are fixed, but the height x of the structure is variable. The goal is to minimize the total mass of the structure while satisfying the condition that the maximal displacement is at most 8 mm for all nodes. Therefore, the cross

sectional areas of all truss elements have to be determined, as well as the height x of nodes 2, 4, and 6, where each node may have a different value for x. It is assumed that the steel bars have cross-sections of equal-sided angle type, and therefore one discrete variable is associated with every truss member, i.e. there are totally 11 discrete variables. There are three continuous variables describing the vertical locations of nodes 2, 4, and 6.

The nodal coordinates are given in Table C.1. Young's modulus is $E = 200$ GPa, and the density of the steel used is $\rho = 7850$ kg/ m^3. The horizontal and vertical displacements of node 1 are zero, and the horizontal displacement of node 7 is zero. The side length of the bars, i.e. the discrete variables, are $u_{i_2} \in \{10, 11, \ldots, 60\}$[mm], $i_2 = 1, 2, \ldots, 11$, and the vertical coordinates of the nodes 2, 4, and 6, i.e. the continuous variables, are $x_{i_1} \in [2000, 3200]$[mm], $i_1 = 1, 2, 3$. The best known solution has the objective function value 88197.2[cm^3].

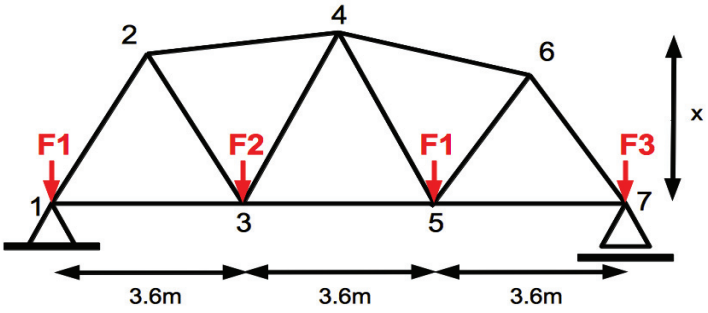


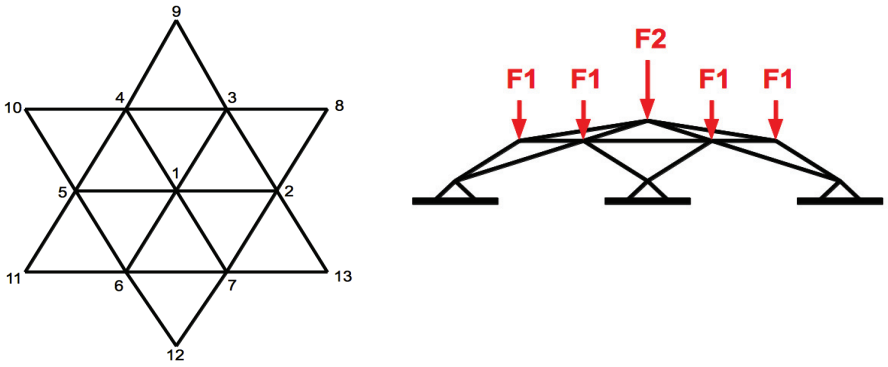
Figure C.1: Eleven element plane truss (test problem 17): The goal is to minimize the weight of the structure while satisfying nodal displacement constraints.

Table C.1: Geometry of the eleven element truss structure.

Node number	Coordinate [mm] x	y
1	0	0
2	1800	x_1
3	3600	0
4	5400	x_2
5	7200	0
6	9000	x_3
7	10800	0

C.18 Test Problem 18: Three-Dimensional Truss Dome

The three-dimensional truss dome is illustrated in Figure C.2. The coordinates of nodes 8-13 are fixed, and the height of nodes 1-7 has to be determined (continuous variables). There are two load cases. Load $F_1 = 20$ kN acts vertically on nodes 2-7, and load $F_2 = 40$ kN acts vertically on node 1. The truss consists of 24 tubular elements that all have the same outer diameter and whose inner diameter has to be determined (discrete variables). The goal is to minimize the total mass while limiting the displacement of node 1 to at most 10 mm. The truss dome has 24 elements, and thus there are 24 discrete variables $u_{i_2} \in \{1, 2, \ldots, 10\}[\,\mathrm{mm}]$, $i_2 = 1, 2, \ldots, 24$, describing the wall thicknesses. The vertical coordinates of nodes 1-7 are represented by the continuous variables $x_{i_1} \in [0, 1000][\,\mathrm{mm}]$, $i_1 = 1, 2, \ldots, 7$. The geometry of the truss dome is summarized in Table C.2. Young's modulus is $E = 200$ GPa, and the density of the steel is $\rho = 7850$ kg/ m^3. The best known solution has the objective function value 63833.2[cm^3].



(a) Top view. (b) Side view.

Figure C.2: Truss dome (test problem 18): The goal is to minimize the weight of the truss dome while satisfying displacement constraints for node 1.

Table C.2: Geometry of the truss dome.

Node number	Coordinate [mm] x	y	z
1	0	0	x_1
2	250	0	x_2
3	125	216.51	x_3
4	-125	216.51	x_4
5	-250	0	x_5
6	-125.0	-216.51	x_6
7	125.0	-216.51	x_7
8	433.01	250	0
9	0	500	0
10	-433.01	250	0
11	-433.01	-250	0
12	0	-500	0
13	433.01	-250	0

C.19 Test Problems 19-21: Reliability-Redundancy Allocation

Reliability-redundancy allocation leads to a mixed-integer nonlinear optimization problem where the discrete variables represent the levels of redundancy and the continuous variables are the component reliabilities that are assumed to be known. The general mathematical formulation of such reliability-redundancy allocation problems is as follows [80].

$$\max_{\mathbf{x},\mathbf{u}} R_s(\mathbf{x},\mathbf{u}) \tag{C.17a}$$

$$\text{s.t. } g_j(\mathbf{x},\mathbf{u}) \leq b_j,\ j = 1,\ldots,m \tag{C.17b}$$

$$u_{i_2} \in \{u^l_{i_2},\ldots,u^u_{i_2}\},\ x_{i_1} \in [x^l_{i_1}, x^u_{i_1}],\ i_1, i_2 = 1,\ldots,k_1, \tag{C.17c}$$

where $R_s(\mathbf{x},\mathbf{u})$ is the system reliability that depends on the system configuration, and the constraints (C.17b) can be, for example, cost constraints or weight restrictions. The continuous variables $\mathbf{x}$ reflect the reliability of each component, whereas the integer variables $\mathbf{u}$ describe the redundancy for each component.

C.19.1 Test Problem 19: Bridge system

A small example of a bridge network is shown in Figure C.3. It is a complex system that is neither a series nor parallel configuration. The system can operate independently of component 5 if either paths 1-2 or 3-4 are working. If component 5 fails, the system cannot continue operating successfully if either the components 1 and 3, 1 and 4, 2 and 3, or 2 and 4 fail simultaneously. The reliability $R_i(x_i, u_i)$ at stage i is defined by

$$R_i = R_i(x_i, u_i) = 1 - (1 - x_i)^{u_i}. \tag{C.18}$$

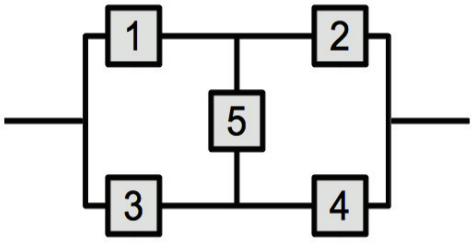


Figure C.3: Bridge configuration.

The resulting reliability-redundancy allocation problem is then defined as follows.

$$\max_{\mathbf{x},\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) = R_1R_2 + R_3R_4 + R_1R_4R_5 + R_2R_3R_5 - R_1R_2R_3R_4 - R_1R_2R_3R_5 - R_1R_2R_4R_5 - R_1R_3R_4R_5 - R_2R_3R_4R_5 + 2R_1R_2R_3R_4R_5 \tag{C.19a}$$

$$\text{s.t.} \quad \sum_{i=1}^{5} p_i u_i^2 \leq P \tag{C.19b}$$

$$\sum_{i=1}^{5} c_i(x_i)\left[u_i + \exp\left(\frac{u_i}{4}\right)\right] \leq C \tag{C.19c}$$

$$\sum_{i=1}^{5} w_i u_i \exp\left(\frac{u_i}{4}\right) \leq W \tag{C.19d}$$

$$0 \leq x_i \leq 1 - 10^{-6},\ i = 1,\ldots,5 \tag{C.19e}$$

$$u_i \in \{1,\ldots,10\}, i = 1,\ldots,5 \tag{C.19f}$$

where $P = 110$, $W = 200$, $C = 175$ and the cost-reliability relation is defined as

$$c_i(x_i) = \alpha_i \left(\frac{-t}{\log x_i}\right)^{\beta_i},\ i = 1,\ldots,5. \tag{C.20}$$

The time for which the system is required to work is set to $t = 1000$. The constants α_i and β_i describe the characteristics of each component at stage i and are given together with the parameters w_i and p_i in Table C.3. The best known solution has the objective function value 0.999659.

Table C.3: Parameters for the bridge configuration.

i	$\alpha_i \times 10^5$	p_i	w_i	β_i
1	2.330	1	7	1.5
2	1.450	2	8	1.5
3	0.541	3	8	1.5
4	8.050	4	6	1.5
5	1.950	2	9	1.5

C.19.2 Test Problem 20: Overspeed Protection System

The overspeed protection system [38] for a gas turbine is modeled as a series system with four components (valves). A mechanical and electrical system is used to detect overspeed. If overspeed occurs, the four control valves must close and cut off the supply of fuel to the gas turbine. The reliability-redundancy allocation problem for the overspeed detection system is defined as follows.

$$\max_{\mathbf{x},\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) = \prod_{i=1}^{4} R_i \tag{C.21a}$$

$$\text{s.t.} \quad \sum_{i=1}^{4} p_i u_i^2 \leq P \tag{C.21b}$$

$$\sum_{i=1}^{4} c_i(x_i)\left[u_i + \exp\left(\frac{u_i}{4}\right)\right] \leq C \tag{C.21c}$$

$$\sum_{i=1}^{4} w_i u_i \exp\left(\frac{u_i}{4}\right) \leq W \tag{C.21d}$$

$$0.5 \leq x_i \leq 1 - 10^{-6},\ i = 1,\ldots,4 \tag{C.21e}$$

$$u_i \in \{1,\ldots,10\}, i = 1,\ldots,4 \tag{C.21f}$$

where $P = 250$, $W = 500$, $C = 400$, and the cost-reliability relation is defined as

$$c_i(x_i) = \alpha_i \left(\frac{-t}{\log x_i}\right)^{\beta_i},\ i = 1,\ldots,4. \tag{C.22}$$

The time for which the system is required to work is set to $t = 1000$, and R_i is defined as in equation (C.18). The constants α_i and β_i describe the

characteristics of each component at stage i and are given together with the parameters w_i and p_i in Table C.4. The best known solution has the objective function value 0.999889.

Table C.4: Parameters for the overspeed detection system.

i	$\alpha_i \times 10^5$	p_i	w_i	β_i
1	1.0	1	6	1.5
2	2.3	2	6	1.5
3	0.3	3	8	1.5
4	2.3	2	7	1.5

C.19.3 Test Problem 21: Series-Parallel System

A small example of a series-parallel system is illustrated in Figure C.4. The system fails if subsystem 1-2 and subsystem 3-4-5 fail simultaneously.

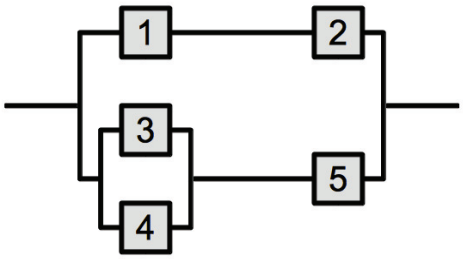


Figure C.4: Series-parallel system.

The reliability-redundancy allocation problem for the series-parallel configuration is defined as follows.

$$\max_{\mathbf{x},\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) = 1 - (1 - R_1 R_2)\,(1 - (1 - R_3)\,(1 - R_4)\,R_5) \tag{C.23a}$$

$$\text{s.t.} \quad \sum_{i=1}^{5} p_i u_i^2 \le P \tag{C.23b}$$

$$\sum_{i=1}^{5} c_i(x_i)\left[u_i + \exp\left(\frac{u_i}{4}\right)\right] \le C \tag{C.23c}$$

$$\sum_{i=1}^{5} w_i u_i \exp\left(\frac{u_i}{4}\right) \le W \tag{C.23d}$$

$$0 \le x_i \le 1 - 10^{-6},\ i = 1,\ldots,5 \tag{C.23e}$$

$$u_i \in \{1,\ldots,10\}, i = 1,\ldots,5, \tag{C.23f}$$

where R_i, $i = 1,\ldots,5$, are defined as in equation (C.18), and c_i, $i = 1,\ldots,5$, are defined as in equation (C.20). The system is required to work for the time $t = 1000$. The parameters are given in Table C.5, and $P = 180$, $W = 100$, and $C = 175$. The best known solution has the objective function value 0.999725.

Table C.5: Parameters for the series-parallel configuration.

i	$\alpha_i \times 10^5$	p_i	w_i	β_i
1	2.500	2	3.5	1.5
2	1.450	4	4.0	1.5
3	0.541	5	4.0	1.5
4	0.541	8	3.5	1.5
5	2.100	4	4.5	1.5

Appendix D

Test Problems Chapter 5

D.1 Test Problem 1

This test problem is an alteration of the test problem G6 by [77], where integrality constraints for each variable have been added. The global optimum is -3971.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 && \text{(D.1a)}\\
\text{s.t.} \quad & (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0 && \text{(D.1b)}\\
& -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0 && \text{(D.1c)}\\
& x_1 \in \{13, 14 \ldots, , 100\} && \text{(D.1d)}\\
& x_2 \in \{0, 1, 2, \ldots, 100\} && \text{(D.1e)}
\end{aligned}$$

D.2 Test Problem 2

This five variable test function has been taken from http://www.aridolan.com/ga/gaa/MultiVarMin.html. The best solution found so far has the objective function value -525.77.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1 \sin(x_1) + 1.7 x_2 \sin(x_1) - 1.5 x_3 - && \text{(D.2a)}\\
& 0.1 x_4 \cos(x_4 + x_5 - x_1) + (0.2 x_5^2 - x_2) - 1 && \text{(D.2b)}\\
\text{s.t.} \quad & x_1, x_2, x_3, x_4, x_5 \in \{-100, -99, \ldots, 99, 100\} && \text{(D.2c)}
\end{aligned}$$

D.3 Test Problem 3

This is an altered version of the problem ex1221 given on http://www.gamsworld.org/minlp/minlplib/ex1221.htm [23] where in-

tegrality constraints for each variable have been added and equality constraints have been replaced by inequality constraints. The best function value found by the compared algorithms is 0.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 2x_1 + 3x_2 + 1.5x_3 + 2x_4 - 0.5x_5 && \text{(D.3a)}\\
\text{s.t.} \quad & x_1^2 + x_3 \leq 1.25 && \text{(D.3b)}\\
& x_2^{1.5} + 1.5x_4 \leq 3 && \text{(D.3c)}\\
& x_1 + x_3 \leq 1.6 && \text{(D.3d)}\\
& 1.333x_2 + x_4 \leq 3 && \text{(D.3e)}\\
& -x_3 - x_4 + x_5 \leq 0 && \text{(D.3f)}\\
& x_1, x_2, x_3 \in \{0, 1, \ldots, 10\} && \text{(D.3g)}\\
& x_4, x_5 \in \{0, 1\} && \text{(D.3h)}
\end{aligned}$$

D.4 Test Problem 4

This test problem is an alteration of the test problem G4 [77], where integrality constraints for each variable have been added. The best solution found by the compared algorithms is -30512.44.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 && \text{(D.4a)}\\
\text{s.t.} \quad & 0 \leq 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leq 92 && \text{(D.4b)}\\
& 90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110 && \text{(D.4c)}\\
& 20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25 && \text{(D.4d)}\\
& x_1 \in \{78, 79 \ldots, 102\} && \text{(D.4e)}\\
& x_2 \in \{33, 34, \ldots, 45\} && \text{(D.4f)}\\
& x_3, x_4, x_5 \in \{27, 28, \ldots, 45\} && \text{(D.4g)}
\end{aligned}$$

D.5 Test Problem 5

This test problem is an alteration of the test problem G9 [77], where integrality constraints for each variable have been added. The best solution found so far has the objective function value 700.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + && \text{(D.5a)}\\
& 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 && \text{(D.5b)}\\
\text{s.t.} \quad & 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0 && \text{(D.5c)}\\
& 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0 && \text{(D.5d)}\\
& 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0 && \text{(D.5e)}\\
& -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0 && \text{(D.5f)}\\
& x_i \in \{-10, -9 \ldots, 9, 10\}\,,\ i = 1, \ldots, 7 && \text{(D.5g)}
\end{aligned}$$

D.6 Test Problem 6

This test problem is an alteration of the test problem G1 [77], where integrality constraints for each variable have been added. The global minimum is -15.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i && \text{(D.6a)}\\
\text{s.t.} \quad & 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10 && \text{(D.6b)}\\
& 2x_1 + 2x_3 + x_{10} + x_{12} \leq 10 && \text{(D.6c)}\\
& 2x_2 + 2x_3 + x_{11} + x_{12} \leq 10 && \text{(D.6d)}\\
& -8x_1 + x_{10} \leq 0 && \text{(D.6e)}\\
& -8x_2 + x_{11} \leq 0 && \text{(D.6f)}\\
& -8x_3 + x_{12} \leq 0 && \text{(D.6g)}\\
& -2x_4 - x_5 + x_{10} \leq 0 && \text{(D.6h)}\\
& -2x_6 - x_7 + x_{11} \leq 0 && \text{(D.6i)}\\
& -2x_8 - x_9 + x_{12} \leq 0 && \text{(D.6j)}\\
& x_i \in \{0, 1\}\,,\ i = 1, \ldots, 9, 13 && \text{(D.6k)}\\
& x_i \in \{0, 1, \ldots, 100\}\,,\ i = 10, 11, 12 && \text{(D.6l)}
\end{aligned}$$

D.7 Test Problem 7

This is the problem nvs09 from the MINLPLib http://www.gamsworld.org/minlp/minlplib/nvs09.htm [23]. The solution value at the global optimum is -43.13.

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{10} \log(x_i - 2)^2 + \sum_{i=1}^{10} \log(10 - x_i)^2 - \prod_{i=1}^{10} x_i^{0.2} \tag{D.7a}$$

$$x_i \in \{3, 4 \ldots, 9\}, \; i = 1, \ldots, 10 \tag{D.7b}$$

D.8 Test Problem 8

This is the problem hmittelman from the MINLPLib http://www.gamsworld.org/minlp/minlplib/hmittelman.htm [23]. The function value at the global optimum is 13.

$$\min \quad f(\mathbf{x}) = 10y_1 + 7y_2 + y_3 + 12y_4 + 8y_5 + 3y_6 + y_7 + 5y_8 + 3y_9 \tag{D.8a}$$

$$\text{s.t.} \quad 3y_1 - 12y_2 - 8y_3 + y_4 - 7y_9 + 2y_{10} \leq -2 \tag{D.8b}$$

$$y_2 - 10y_3 - 5y_5 + y_6 + 7y_7 + y_8 \leq -1 \tag{D.8c}$$

$$5y_1 - 3y_2 - y_3 - 2y_8 + y_{10} \leq -1 \tag{D.8d}$$

$$3y_2 - 5y_1 + y_3 + 2y_8 - y_{10} \leq 1 \tag{D.8e}$$

$$-4y_3 - 2y_4 - 5y_6 + y_7 - 9y_8 - 2y_9 \leq -3 \tag{D.8f}$$

$$9y_2 - 12y_4 - 7y_5 + 6y_6 + 2y_8 - 15y_9 + 3y_{10} \leq -7 \tag{D.8g}$$

$$5y_2 - 8y_1 + 2y_3 - 7y_4 - y_5 - 5y_7 - 10y_9 \leq -1 \tag{D.8h}$$

$$y_1 = x_5 x_7 x_9 x_{10} x_{14} x_{15} x_{16} \tag{D.8i}$$

$$y_2 = x_1 x_2 x_3 x_4 x_8 x_{11} \tag{D.8j}$$

$$y_3 = x_3 x_4 x_6 x_7 x_8 \tag{D.8k}$$

$$y_4 = x_3 x_4 x_8 x_{11} \tag{D.8l}$$

$$y_5 = x_6 x_7 x_8 x_{12} \tag{D.8m}$$

$$y_6 = x_6 x_7 x_9 x_{14} x_{16} \tag{D.8n}$$

$$y_7 = x_9 x_{10} x_{14} x_{16} \tag{D.8o}$$

$$y_8 = x_5 x_{10} x_{14} x_{15} x_{16} \tag{D.8p}$$

$$y_9 = x_1 x_2 x_{11} x_{12} \tag{D.8q}$$

$$y_{10} = x_{13} x_{14} x_{15} x_{16} \tag{D.8r}$$

$$x_i \in \{0, 1\}, i = 1, \ldots, 16 \tag{D.8s}$$

D.9 Test Problem 9

This is the 12-dimensional Rastrigin function [142] where integrality constraints for each variable have been added. The function value at the optimum is -12.

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - \cos(2\pi x_i) \tag{D.9a}$$

$$\text{s.t.} \quad x_i \in \{-1, 0, 1, 2, 3\}, \ i = 1, \ldots, 12 \tag{D.9b}$$

D.10 Test Problem 10

This is the 30-dimensional Rastrigin function [142] where integrality constraints for each variable have been added. The optimal function value is -30.

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - \cos(2\pi x_i) \tag{D.10a}$$

$$\text{s.t.} \quad x_i \in \{-1, 0, 1, 2, 3\}, \ i = 1, \ldots, 30 \tag{D.10b}$$

D.11 Test Problem 11

This test problem is an alteration of the test problem G2 [77], where integrality constraints for each variable have been added. The problem dimension used is $k = 25$. The best function value found by the compared algorithms is 0.3225.

$$\max \quad f(\mathbf{x}) = \left| \frac{\sum_{i=1}^{k} \cos^4(x_i) - 2\prod_{i=1}^{k} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{k} i x_i^2}} \right| \tag{D.11a}$$

$$\text{s.t.} \quad \prod_{i=1}^{k} x_i \geq 0.75 \tag{D.11b}$$

$$\sum_{i=1}^{k} x_i \leq 7.5k \tag{D.11c}$$

$$x_i \in \{0, 1, 2, \ldots, 10\}, \ i = 1, 2, \ldots, 25 \tag{D.11d}$$

D.12 Test Problem 12

This is the Schoen test function [128] with 20 variables, altered to binary variable constraints. The best solution found by the compared algorithms is -13.36.

$$\min \quad f(\mathbf{x}) = \frac{\sum_{i=1}^{\eta} f_i \prod_{j \neq i} \|\mathbf{x} - \mathbf{z}_j\|_2^{\alpha_j}}{\sum_{i=1}^{\eta} \prod_{j \neq i} \|\mathbf{x} - \mathbf{z}_j\|_2^{\alpha_j}}, \tag{D.12}$$

where $\mathbf{x} \in \{0,1\}^{20}$, $\eta = 6$, $\mathbf{z}_j \in [0,1]^{20}, \forall j = 1,\ldots,6$, $f_i \in \mathbb{R}, \forall i = 1,\ldots,6$, and $\alpha_i \in \mathbb{R}^+, \forall i = 1,\ldots,6$, and where $\|\cdot\|_2$ denotes the Euclidean norm. For the Schoen functions it holds that

1. $f(\mathbf{z}_i) = f_i, i = 1,\ldots,\eta$,
2. $\min_{1 \leq i \leq \eta} f_i \leq f(\mathbf{x}) \leq \max_{1 \leq i \leq \eta} f_i$ for all $\mathbf{x} \in \{0,1\}^k$,
3. $\lim_{\mathbf{x} \to \mathbf{z}_j} \nabla f(\mathbf{x}) = 0$ for all $j = 1,\ldots,\eta$.

D.13 Test Problem 13

This is an altered version of the problem nvs09 given on http://www.gamsworld.org/minlp/minlplib/nvs09.htm [23]. The best objective function value found by the compared algorithms is -9591.72.

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{10} \log(x_i - 2)^2 \sum_{i=1}^{10} \log(100 - x_i)^2 - \prod_{i=1}^{10} x_i^{0.2} \tag{D.13a}$$

$$\text{s.t.} \quad x_i \in \{3, 4 \ldots, 99\}, \; i = 1,\ldots,10 \tag{D.13b}$$

D.14 Test Problem 14

This test problem is an alteration of the test problem G1 [77], where integrality constraints for each variable have been added and the variable domains have been changed. The best solution found by the compared algorithms is -50200.

$$\min \quad f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{D.14a}$$

$$\text{s.t.} \quad 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10 \tag{D.14b}$$

$$2x_1 + 2x_3 + x_{10} + x_{12} \leq 10 \tag{D.14c}$$

$$2x_2 + 2x_3 + x_{11} + x_{12} \leq 10 \tag{D.14d}$$

$$-8x_1 + x_{10} \leq 0 \tag{D.14e}$$

$$-8x_2 + x_{11} \leq 0 \tag{D.14f}$$

$$-8x_3 + x_{12} \leq 0 \tag{D.14g}$$

$$-2x_4 - x_5 + x_{10} \leq 0 \tag{D.14h}$$

$$-2x_6 - x_7 + x_{11} \leq 0 \tag{D.14i}$$

$$-2x_8 - x_9 + x_{12} \leq 0 \tag{D.14j}$$

$$x_i \in \{0, 1, \ldots, 100\}, \; i = 1, \ldots, 13 \tag{D.14k}$$

D.15 Test Problem 15

This is the 12-dimensional Rastrigin function [142] where integrality constraints for each variable have been added and the variable domains have been increased. The global minimum is -12.

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - \cos(2\pi x_i) \tag{D.15a}$$

$$\text{s.t.} \quad x_i \in \{-10, 9, \ldots, 29, 30\}, \; i = 1, \ldots, 12 \tag{D.15b}$$

D.16 Test Problem 16

This is an 8-dimensional convex function. The global minimum is 0.

$$\min \quad f(\mathbf{x}) = 3.1x_1^2 + 7.6x_2^2 + 6.9x_3^2 + 0.004x_4^2 + 19x_5^2 + 3x_6^2 + x_7^2 + 4x_8^2 \tag{D.16a}$$

$$\text{s.t.} \quad x_i \in \{-10, 9, \ldots, 9, 10\}, \; i = 1, \ldots, 8 \tag{D.16b}$$

D.17 Test Problem 17

This is an altered version of the problem ex1221 given on http://www.gamsworld.org/minlp/minlplib/ex1221.htm [23] where integrality constraints for all variables have been added and the nonlinear constraints have been omitted. The best solution found by the algorithms is -8.

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = -2x_1 - 3x_2 - 1.5x_3 - 2x_4 + 0.5x_5 && \text{(D.17a)}\\
\text{s.t.} \quad & x_1 + x_3 \leq 1.6 && \text{(D.17b)}\\
& 1.333x_2 + x_4 \leq 3 && \text{(D.17c)}\\
& -x_3 - x_4 + x_5 \leq 0 && \text{(D.17d)}\\
& x_1, x_2, x_3 \in \{0, 1, \ldots, 10\} && \text{(D.17e)}\\
& x_4, x_5 \in \{0, 1\} && \text{(D.17f)}
\end{aligned}$$

D.18 Test Problems 18 and 19: Throughput Maximization

The best feasible objective function value found for problem 18 is 9.8965. The feasible objective function value found for problem 19 is 5.1966. Both results were found by SO-I.

D.19 Test Problems 20a-20c and 21a-21c: Hydropower Maximization

The best feasible objective function values found for the problems are summarized in Table D.1.

Table D.1: Best feasible function values for hydropower maximization problems.

Problem	20a	20b	20c	21a	21b	21c
Best value	758.25	2022.50	4172.50	1681.80	4148.00	8325.00

Bibliography

[1] M.A. Abramson and C. Audet. Convergence of mesh adaptive direct search to second-order stationary points. *SIAM Journal on Optimization*, 17:606–619, 2006.

[2] M.A. Abramson, C. Audet, J. Chrissis, and J. Walston. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3:35–47, 2009.

[3] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis Jr, and S. Le Digabel. The NOMAD project. Software available at http://www.gerad.ca/nomad. 2011.

[4] M.A. Abramson, C. Audet, and J.E. Dennis Jr. Filter pattern search algorithms for mixed variable constrained optimization problems. *SIAM Journal on Optimization*, 11:573–594, 2004.

[5] D.H. Ackley. *A connectionist machine for genetic hillclimbing*. Boston: Kluwer Academic Publishers, 1987.

[6] C.S. Adjiman, I.P. Androulakis, and C.A. Floudas. Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46:1769–1797, 2000.

[7] M.J. Appel, R. Labarre, and D. Radulović. On accelerated random search. *SIAM Journal on Optimization*, 14:708–731, 2003.

[8] I. Argatov, P. Rautakorpi, and R. Silvennoinen. Estimation of the mechanical energy output of the kite wind generator. *Renewable Energy*, 34:1525–1532, 2009.

[9] J.G. Arnold, R. Srinivasan, R.R. Muttiah, and J.R. Williams. Large area hydrologic modeling and assessment part 1: model development. *Journal of the American Water Resources Association*, 34:73–89, 1998.

[10] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41:299–318, 2008.

[11] C. Audet and J.E. Dennis Jr. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11:573–594, 2000.

[12] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[13] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[14] P. Balaprakash, S.M. Wild, and P.D. Hovland. An experimental study of global and local search algorithms in empirical performance tuning. In *10th International Meeting on High-Performance Computing for Computational Science (VECPAR 2012)*, 2012.

[15] P. Balaprakash, S.M. Wild, and B. Norris. SPAPT: Search problems in automatic performance tuning. *Procedia Computer Science*, 9:1959–1968, 2012.

[16] C. Bazelton and K. Smetters. Discounting inside the Washington, D.C. beltway. *Journal of Economic Perspectives*, 13:213–228, 1999.

[17] O. Berman and N. Ashrafi. Optimization models for reliability of modular software systems. *IEEE Transactions on Software Engineering*, 19:1119–1123, 1993.

[18] M. Björkman and K. Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373–397, 2001.

[19] N. Bliznyuk, D. Ruppert, C.A. Shoemaker, R.G. Regis, S.M. Wild, and P. Mugunthan. Bayesian calibration of computationally expensive models using optimization and radial basis function approximation. *Journal of Computational and Graphical Statistics*, 17:1–25, 2008.

[20] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.

[21] A.J. Booker, J.E. Dennis Jr, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Multidisciplinary Optimization*, 17:1–13, 1999.

[22] J. Breukels. *An engineering methodology for kite design*. PhD thesis, Delft University of Technology, 2011.

[23] M.R. Bussieck, A. Stolbjerg Drud, and A. Meeraus. MINLPLib – a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15:114–119, 2003.

[24] G. Cao, Z. Cai, Z. Liu, and G. Wang. Daily optimized model for long-term operation of the Three Gorges-Gezhouba Cascade Power Stations. *Science in China series E: Technological Sciences*, 50:98–110, 2007.

[25] T.R. Chandrupatla and A.D. Belegundu. *Introduction to Finite Elements in Engineering*. Prentice Hall, 2002.

[26] T.-C. Chen. IAs based approach for reliability redundancy allocation problems. *Applied Mathematics and Computation*, 182:1556–1567, 2006.

[27] D.W. Coit and A.E. Smith. Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, 45:254–266, 1996.

[28] A.R. Conn, K. Scheinberg, and P.L. Toint. A derivative free optimization algorithm in practice. In *Proceedings of AIAA St Louis Conference*, 1998.

[29] A.R. Conn, K. Scheinberg, and L.N. Vicente. Global convergence of general derivative free trust-region algorithms to first and second order critical points. *SIAM Journal on Optimization*, 20:387–415, 2009.

[30] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, 2009.

[31] L. Costa and P. Oliveira. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, 25:257–266, 2001.

[32] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. A Bayesian approach to the design and analysis of computer experiments. Technical report, Oak Ridge National Laboratory, Oak Ridge, TN, 1988.

[33] E. Davis and M. Ierapetritou. Kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *Journal of Global Optimization*, 43:191–205, 2009.

[34] K.A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[35] K. Deb. An efficient constraint handling method for genetic algorithms. *Computational Methods in Applied Mechanics and Engineering*, 186:311–338, 2000.

[36] T. Dede, S. Bekiroğlu, and Y. Ayvaz. Weight minimization of trusses with genetic algorithm. *Applied Soft Computing*, 11:2565–2575, 2011.

[37] A.P. Dempster. A generalization of Bayesian inference. *Journal of the Royal Statistical Society*, Series B 30:205–247, 1968.

[38] A.K. Dhingra. Optimal apportionment of reliability & redundancy in series systems under multiple objectives. *IEEE Transactions on Reliability*, 41:576–582, 1992.

[39] L.C.W. Dixon and G. Szegö. *The global optimization problem: an introduction. In: Towards Global Optimization*, volume 2. North-Holland, Amsterdam, 1978.

[40] J. Duchon. *Constructive Theory of Functions of Several Variables*. Springer-Verlag, Berlin, 1977.

[41] T. Dunne and R.D. Black. Partial area contributions to storm runoff in a small New England watershed. *Water Resources Research*, 6:1296–1311, 1970.

[42] T. Dunne and L. Leopold. *Water in Environmental Planning*. W.H. Freeman & Co., New York, p. 818, 1978.

[43] M.A. Duran and I.E. Grossmann. A mixed-integer nonlinear programming algorithm for process systems synthesis. *AIChE Journal*, 32:592–606, 1986.

[44] Z.M. Easton, D.R. Fuka, M.T. Walter, D.M. Cowan, E.M. Schneiderman, and T.S. Steenhuis. Re-conceptualizing the soil and water assessment tool (SWAT) model to predict runoff from variable source areas. *Journal of Hydrology*, 348:279–291, 2008.

[45] L. Fagiano. *Control of Tethered Airfoils for High-Altitude Wind Energy Generation*. PhD thesis, Politecnico di Torino, 2009.

[46] C.A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, Oxford, 1995.

[47] C.A. Floudas, O.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer, and C.A. Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1999.

[48] A. Forrester, A. Sóbester, and A. Keane. *Engineering Design via Surrogate Modelling - A Practical Guide*. Wiley, 2008.

[49] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–67, 1991.

[50] V.B. Gantovnik, Z. Gürdal, L.T. Watson, and C.M. Anderson-Cook. A genetic algorithm for mixed integer nonlinear programming problems using separate constraint approximations. *AIAA Journal*, 43:1844–1849, 2005.

[51] S.B. Gershwin and J.E. Schor. Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 93:117–144, 2000.

[52] B. Glaz, P.P. Friedmann, and L. Liu. Surrogate based optimization of helicopter rotor blades for vibration reduction in forward flight. *Structural and Multidisciplinary Optimization*, 35:341–363, 2008.

[53] T. Goel, R.T. Haftka, W. Shyy, and N.V. Queipo. Ensemble of surrogates. *Structural Multidisciplinary Optimization*, 33:199–216, 2007.

[54] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[55] K. Gopal, K.K. Aggarwal, and J.S. Gupta. A method for solving reliability optimization problem. *IEEE Transactions on Reliability*, R-29:36–38, 1980.

[56] I. Griva, S.G. Nash, and A. Sofer. *Linear and Nonlinear Optimization, 2nd Edition*. SIAM, George Mason University, Fairfax, VA, 2009.

[57] H.-M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[58] P. Hansen and N. Mladenović. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[59] R.L. Haupt. Antenna design with a mixed integer genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 55:577–582, 2007.

[60] T. Hemker. *Derivative Free Surrogate Optimization for Mixed-Integer Nonlinear Black Box Problems in Engineering*. PhD thesis, TU Darmstadt, 2008.

[61] F. Herrera, M. Lozano, and J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12:265–319, 1998.

[62] M. Hikita, Y. Nakagawa, K. Nakashima, and H. Narihisa. Reliability optimization of systems by a surrogate-constraints algorithm. *IEEE Transactions on Reliability*, R-41:473–480, 1992.

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

[64] K. Holmström, N.-H. Quttineh, and M.M. Edvall. An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. *Journal of Global Optimization*, 41:447–464, 2008.

[65] B. Horowitz, L.J. do Nascimento Guimaräes, V. Dantas, and S.M. Bastos Afonso. A concurrent efficient global optimization algorithm applied to polymer injection strategies. *Journal of Petroleum Science and Engineering*, 71:195–204, 2010.

[66] S. Hosder, L.T. Watson, B. Grossman, W.H. Mason, H. Kim, R.T. Haftka, and S.E. Cox. Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. *Optimization and Engineering*, 2:431–452, 2001.

[67] B. Houska. Robustness and stability optimization of open-loop controlled power generating kites. Master's thesis, Ruprecht-Karls-Universität Heidelberg, Fakultät für Mathematik und Informatik, 2007.

[68] Y.C. Hsieh, T.C. Chen, and D.L. Bricker. Genetic algorithms for reliability design problems. Technical report, Department of Industrial Engineering, University of Iowa, 1997.

[69] J. Hu, U.Y. Ogras, and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25:2919–2933, 2006.

[70] T. Inagaki. Interdependence between safety-control policy and multiple-sensor schemes via Dempster-Shafer theory. *IEEE Transactions on Reliability*, 40:182–188, 1991.

[71] G. Jekabsons. ARESLab: Adaptive Regression Splines toolbox for Matlab. *available at http://www.cs.rtu.lv/jekabsons/*, 2010.

[72] H.A. Jensen and J.G. Sepulveda. Structural optimization of uncertain dynamical systems considering mixed-design variables. *Probabilistic Engineering Mechanics*, 26:269–280, 2011.

[73] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[74] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[75] J.-C. Jouhaud, P. Sagaut, M. Montagnac, and J. Laurenceau. A surrogate-model based multidisciplinary shape optimization method with application to a 2D subsonic airfoil. *Computers & Fluids*, 36:520–529, 2007.

[76] G.R. Kocis and I.E. Grossmann. Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial & Engineering Chemistry Research*, 27:1407–1421, 1988.

[77] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7:19–44, 1999.

[78] E.C. Kuipers. http://www.mathworks.com/matlabcentral/fileexchange/95. 2000.

[79] W. Kuo, H. Lin, Z. Xu, and W. Zhang. Reliability optimization with Lagrange multiplier and branch-and-bound technique. *IEEE Transactions on Reliability*, R-36:624–630, 1987.

[80] W. Kuo, V.R. Prasad, F.A. Tillman, and C.-L. Hwang. *Optimal Reliability Design: Fundamentals and Applications*. Cambridge University Press, 2001.

[81] X.B. Lam, Y.S. Kim, A.D. Hoang, and C.W. Park. Coupled aerostructural design optimization using the kriging model and integrated multiobjective optimization algorithm. *Journal of Optimization Theory and Applications*, 142:533–556, 2009.

[82] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[83] A. Levy, A. Montalvo, S. Gomez, and A. Galderon. *Topics in Global Optimization*. Springer-Verlag, New York, 1981.

[84] C. Li, F.-L. Wang, Y.-Q. Chang, and Y. Liu. A modified global optimization method based on surrogate model and its application in packing profile optimization of injection molding process. *The International Journal of Advanced Manufacturing Technology*, 48:505–511, 2010.

[85] F. Li, C.A. Shoemaker, J. Wei, and X. Fu. Estimating maximal annual energy given heterogeneous hydropower generating units with application to the Three Gorges System. *Journal of Water Resources Planning and Management*, DOI: 10.1061/(ASCE)WR.1943-5452.0000250, 2012.

[86] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35:561–569, 2008.

[87] G. Liuzzi, S. Lucidi, and F. Rinaldi. Derivative-free methods for bound constrained mixed-integer optimization. *Computational Optimization and Applications*, DOI 10.1007/s10589-011-9405-3, 2011.

[88] S.N. Lophaven, H.B. Nielsen, and J. Søndergaard. DACE a Matlab kriging toolbox. Technical report, IMM-TR-2002-12, 2002.

[89] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press*, 1:281–297, 1967.

[90] J.I. Madsen, W. Shyy, and R.T. Haftka. Response surface techniques for diffuser shape optimization. *AIAA Journal*, 38:1512–1518, 2000.

[91] W.A. Magat and W.K. Viscusi. Effectiveness of the EPA's regulatory enforcement: the case of industrial effluent standards. *Journal of Law and Economics*, 33:331–360, 1990.

[92] G. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

[93] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[94] Z. Michalewicz and D. Fogel. *How To Solve It: Modern Heuristics*. Springer, 2004.

[95] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[96] R.C. Morgans, A.C. Zander, C.H. Hansen, and D.J. Murphy. EGO shape optimization of horn-loaded loudspeakers. *Optimization and Engineering*, 9:361–374, 2008.

[97] H. Mühlenbein, D. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.

[98] J. Müller and R. Piché. Mixture surrogate models based on Dempster-Shafer theory for global optimization problems. *Journal of Global Optimization*, 51:79–104, 2011.

[99] J. Müller, C.A. Shoemaker, and R. Piché. SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers and Operations Research*, http://dx.doi.org/10.1016/j.cor.2012.08.022, 2012.

[100] J. Müller, C.A. Shoemaker, and R. Piché. A stochastic mixture surrogate model algorithm for computationally expensive black-box global optimization problems. In *Proceedings of the global optimization workshop 2012*, 2012.

[101] R.H. Myers and D.C. Montgomery. *Response Surface Methodology, Process and Product Optimization using Designed Experiments*. Wiley-Interscience Publication, 1995.

[102] B.A. Needelman, W.J. Gburek, G.W. Petersen, A.N. Sharpley, and P.J.A. Kleinman. Surface runoff along two agricultural hillslopes with contrasting soils. *Soil Science Society of America journal*, 68:914–923, 2004.

[103] S.L. Neitsch, J.G. Arnold, J.R. Kiniry, and J.R. . Williams. Soil and water assessment tool theoretical documentation version 2005. Technical report, US Department of Agriculture – Agricultural Research Service, Temple, Texas., 2004.

[104] A. Neumeier. Complete search in constrained global optimization. *Acta Numerica*, 13:271–369, 2004.

[105] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Birkhäuser, 2005.

[106] I. Nowak and S. Vigerske. LaGO - a (heuristic) branch and cut algorithm for nonconvex MINLPs. *Central European Journal of Operations Research*, 16:127–138, 2008.

[107] E.M. Owens, S.W. Effler, S.M. Doerr, R.K. Gelda, E.M. Schneiderman, D.G. Lounsbury, and C.L. Stepczuk. A strategy for reservoir model forecasting based on historic meteorological conditions. *Lake and Reservoir Management*, 14:322–331, 1998.

[108] J. Pichitlamken, B.L. Nelson, and L.J. Hong. A sequential procedure for neighborhood selection-of-the-best in optimization via simulation. *European Journal of Operational Research*, 173:283–298, 2006.

[109] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7:155–162, 1964.

[110] M.J.D. Powell. *The Theory of Radial Basis Function Approximation in 1990*. Advances in Numerical Analysis, vol. 2: wavelets, subdivision algorithms and radial basis functions. Oxford University Press, Oxford, pp. 105-210, 1992.

[111] M.J.D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, Series B 92:555–582, 2002.

[112] M.J.D. Powell. Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, 28:649–664, 2008.

[113] M.J.D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Centre for Mathematical Sciences, University of Cambridge, UK, 2009.

[114] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 2005.

[115] S.D. Rajan and D.T. Nguyen. Design optimization of discrete structural systems using MPI-enabled genetic algorithm. *Structural and Multidisciplinary Optimization*, 27:1–9, 2004.

[116] K. Rashid, S. Ambani, and E. Cetinkaya. An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. *Engineering Optimization*, DOI:10.1080/0305215X.2012.665450, 2012.

[117] R.G. Regis. Convergence guarantees for generalized adaptive stochastic search methods for continuous global optimization. *European Journal of Operational Research*, 207:1187–1202, 2010.

[118] R.G. Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers & Operations Research*, 38:837–853, 2011.

[119] R.G. Regis and C.A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19:497–509, 2007.

[120] R.G. Regis and C.A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37:113–135, 2007.

[121] R.G. Regis and C.A. Shoemaker. Parallel radial basis function methods for the global optimization of expensive functions. *European Journal of Operational Research*, 182:514–535, 2007.

[122] R.G. Regis and C.A. Shoemaker. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing*, 21:411–426, 2009.

[123] R.G. Regis and C.A. Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, DOI:10.1080/0305215X.2012.687731, 2012.

[124] R.G. Regis and C.A. Shoemaker. A quasi-multistart framework for global optimization of expensive functions using response surface models. *Journal of Global Optimization*, DOI 10.1007/s10898-012-9940-1, 2012.

[125] N.V. Sahinidis and M. Tawarmalani. *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs, User's manual*, 2010.

[126] K. Schittkowski, C. Zillober, and R. Zotemantel. Numerical comparison of nonlinear programming algorithms for structural optimization. *Structural Optimization*, 7:1–19, 1994.

[127] M. Schlüter, M. Gerdts, and J.-J. Rückmann. MIDACO: New global optimization software for MINLPs. http://www.midaco-solver.com/about.html, 2010.

[128] F. Schoen. A wide class of test functions for global optimization. *Journal of Global Optimization*, 3:133–137, 1993.

[129] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[130] S. Shan and G.G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41:219–241, 2010.

[131] B. Shankar, E.A. DeVuyst, D.C. White, J.B. Braden, and R.H. Hornbaker. Nitrate abatement practices, farm profits and lake water quality: a central Illinois case study. *Journal of Soil and Water Conservation*, 55:296–304, 2000.

[132] F. Smarandache and J. Dezert. *Advances and Applications of DSmT for Information Fusion, Collected Works, Volume 2*. American Research Press, 2006.

[133] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.

[134] D.D. Spinellis and C.T. Papadopoulos. A simulated annealing approach for buffer allocation in reliable production lines. *Annals of Operations Research*, 93:373–384, 2000.

[135] M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24:656–667, 1994.

[136] R. Stocki, K. Kolanek, S. Jendo, and M. Kleiber. Study on discrete optimization techniques in reliability-based optimization of truss structures. *Computers and Structures*, 79:2235–2247, 2001.

[137] M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[138] F.A. Tillman, C.L. Hwang, and W. Kuo. Determining component reliability and redundancy for optimum system reliability. *IEEE Transactions on Reliability*, R-26:162–165, 1977.

[139] B. Tolson and C.A. Shoemaker. Cannonsville reservoir watershed SWAT2000 model development, calibration and validation. *Journal of Hydrology*, 337:68–89, 2007.

[140] B.A. Tolson, M. Asadzadeh, H.R. Maier, and A. Zecchin. Hybrid discrete dynamically dimensioned search (HD-DDS) algorithm for water distribution system design optimization. *Water Resources Research*, pages W12416, DOI:10.1029/2008WR007673, 2009.

[141] B.A. Tolson and C.A. Shoemaker. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resources Research*, 43:W01413, 16 pages, 2007.

[142] A. Törn and A. Zilinskas. *Global Optimization. Lecture Notes in Computer Science, 350.* Springer-Verlag, Berlin, 1989.

[143] L.S. VanDyke, J.W. Pease, D.J. Bosch, and J.C. Baker. Nutrient management planning on four Virginia livestock farms: impacts on nutrient losses and farm income. *Journal of Soil and Water Conservation*, 54:499–506, 1999.

[144] F.A.C. Viana and R.T. Haftka. Using multiple surrogates for minimization of the RMS error in metamodeling. In *Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference DETC2008-49240*, 2008.

[145] J. Viswanathan and I.E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers & Chemical Engineering*, 14:769–782, 1990.

[146] E. Wang, W.L. Harman, J.R. Williams, and J.M. Sweeten. Profitability and nutrient losses of alternative manure application strategies with conservation tillage. *Journal of Soil and Water Conservation*, 57:221–229, 2002.

[147] T. Westerlund and F. Petterson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19:131–136, 1995.

[148] T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3:253–280, 2002.

[149] S.M. Wild, R.G. Regis, and C.A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30:3197–3219, 2007.

[150] J.R. Williams, P.M. Clark, and P.G. Balch. Streambank stabilization: an economic analysis from the landowner's perspective. *Journal of Soil and Water Conservation*, 59:252–259, 2004.

[151] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[152] J. Woodbury. *Multi-models, calibration, and optimization for cost based water quality management of watershed models.* PhD thesis, Cornell University, Ithaca, NY, 2012.

[153] G.A.A. Wossink and D.L. Osmund. Farm economics to support the design of cost-effective Best Management Practice (BMP) programs to improve water quality: nitrogen control in the Neuse River Basin, North Carolina. *Journal of Soil and Water Conservation*, 57:213–221, 2002.

[154] R.R. Yager. On the Dempster-Shafer framework and new combination rules. *Information Sciences*, 41:93–137, 1987.

[155] R.J. Yang, N. Wang, C.H. Tho, J.P. Bobineau, and B.P. Wang. Metamodeling development for vehicle frontal impact simulation. *Journal of Mechanical Design*, 127:1014–1021, 2005.

[156] K.Q. Ye, W. Li, and A. Sudjianto. Algorithmic construction of optimal symmetric Latin hypercube designs. *Journal of Statistical Planning and Inference*, 90:145–159, 2000.

[157] T. Yokota, M. Gen, and Y.-X. Li. Genetic algorithm for nonlinear mixed-integer programming problems and its application. *Computers & Industrial Engineering*, 30:905–917, 1996.

[158] J.-H. Yoon and C.A. Shoemaker. Comparison of optimization methods for groundwater bioremediation. *Journal of Water Resources Planning Management*, 125:54–63, 1999.

[159] X. Yuan, S. Zhang, L. Piboleau, and S. Domenech. Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés. *R.A.I.R.O. Operation Research*, 22:331–346, 1988.

[160] Y. Yuan, S.M. Dabney, and R.L. Bingner. Cost effectiveness of agricultural BMPs for sediment reduction in the Mississippi Delta. *Journal of Soil and Water Conservation*, 57:259–268, 2002.

[161] L.A. Zadeh. Review of book: A mathematical theory of evidence. *The AI Magazine*, 5:81–83, 1984.

[162] J. Zhang, H.S.-H. Chung, and W.-L. Lo. Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 11:326–335, 2007.

[163] P. Zhu, Y. Zhang, and G.-L. Chen. Metamodel-based lightweight design of an automotive front-body structure using robust optimization. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, DOI: 10.1243/09544070JAUTO1045, 2009.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-2959-7
ISSN 1459-2045