VILNIUS UNIVERSITY

Gražina Gimbutienė

ALGORITHMS FOR NON-CONVEX GLOBAL OPTIMIZATION BASED ON THE STATISTICAL AND LIPSCHITZ OBJECTIVE FUNCTION MODELS

Doctoral dissertation
Physical sciences, informatics (09P)

Vilnius, 2017

The dissertation work was carried out at Vilnius University from 2013 to 2017.

Scientific supervisor:

Prof. Dr. Habil. Antanas Žilinskas (Vilnius University, physical sciences, informatics - 09P).

VILNIAUS UNIVERSITETAS

Gražina Gimbutienė

STATISTINIAIS IR LIPŠICO TIKSLO FUNKCIJOS MODELIAIS PAGRĮSTI NEIŠKILOS GLOBALIOSIOS OPTIMIZACIJOS ALGORITMAI

Daktaro disertacija
Fiziniai mokslai, informatika (09P)

Vilnius, 2017

Disertacija rengta 2013–2017 metais Vilniaus universitete.

Mokslinis vadovas:

prof. habil. dr. Antanas Žilinskas (Vilniaus universitetas, fiziniai mokslai, informatika - 09P).

Abstract

Global optimization problems arise in practice whenever there is a need to select a collection of variables corresponding to the best value of some objective function, e. g. the lowest price. In a typical "black-box" situation, when an analytic expression of the function is unavailable, algorithms based on statistical or Lipschitz objective function models can be applied. Statistical models are especially useful when function evaluations are expensive, therefore the efficiency of the algorithms has to be increased in terms of the number of trials. To this end, two approaches combining the statistical global search with local search techniques were suggested and their efficiency solving difficult multimodal problems was experimentally demonstrated. The optimization efficiency is influenced by the selected statistical model as well. Model selection problem was investigated experimentally and respective guidelines were formulated based on a priori information about the objective function complexity. In order to ensure not only efficient, but also theoretically justified search for optimal solutions, theoretical investigation of two algorithms was performed. First, asymptotic properties of a simplicial statistical model were investigated, allowing to relate a heuristic simplex selection criterion to the probability of improvement. Second, an optimal algorithm was derived, justifying the use of a certain trisection procedure in a set of Lipschitz optimization algorithms.

Santrauka

Globaliosios optimizacijos uždaviniai iškyla praktikoje atsiradus poreikiui surasti kintamųjų kombinaciją, atitinkančią geriausią tam tikros tikslo funkcijos reikšmę, pvz. mažiausią kainą. Dažnoje "juodosios dėžės" situacijoje, kai funkcija gali būti apskaičiuojama apibrėžimo srityje, tačiau jos analitinė išraiška nežinoma, sėkmingai taikomi algoritmai, pagrįsti statistiniais ar Lipšico tikslo funkcijos modeliais. Statistiniai modeliai taikytini, kai funkcijos įvertinimai brangūs, dėl to svarbu didinti algoritmų efektyvumą, mažinant globaliojo minimumo paieškai sunaudotą jų skaičių. Tuo tikslu disertacijoje pasiūlyti du statistinės globaliosios optimizacijos derinimo su lokaliąja paieška būdai bei eksperimentiškai pademonstruotas jų efektyvumas sprendžiant sudėtingus neiškilosios optimizacijos uždavinius. Optimizacijos algoritmo efektyvumą lemia ir statistinio modelio pasirinkimas. Modelio pasirinkimo uždavinys disertacijoje tirtas remiantis pasiūlyta eksperimentine metodika, suformuluotos rekomendacijos atsižvelgiant į numatomą tikslo funkcijos sudėtingumą. Siekiant, kad optimalių sprendimų paieška būtų ne tik efektyvi, tačiau ir teoriškai pagrįsta, atlikti du teoriniai algoritmų savybių tyrimai. Pirma, ištyrus simpleksinio statistinio modelio asimptotiką, euristinis simplekso išrinkimo kriterijus susietas su pagerinimo tikimybe. Antra, išvestas optimalus algoritmas, pagrindžiantis stačiakampių posričių dalijimą į tris lygias dalis Lipšico optimizacijos algoritmuose.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Notation

A	the feasible region
d	the dimension of the problem
$f^* = \min_{\mathbf{x} \in A} f(\mathbf{x})$	the global minimum of the objective function
$X^* = \{\mathbf{x}^* \in A : f(\mathbf{x}^*) = f^*\}$	the set of points at which the global minimum is attained
$\mathbf{x}^* \in X^*$	the global minimizer
n	the number of already performed objective function evaluations
$y_i = f(\mathbf{x}_i), i = 1, \ldots, n$	the performed objective function evaluations (trials)
$y_{on} = \min_{i=1,\ldots,n} y_i$	the best function value found (the record), used as an approximation to the global minimum
$\mathbf{x}_{on} = \arg\min_{\mathbf{x}_i, i=1,\ldots,n} f(\mathbf{x}_i)$	the trial point corresponding to the record value
$x_1^n < \cdots < x_n^n$	the ordered univariate trial points
$y_i^n = f(x_i^n), i = 1, \ldots, n$	the corresponding objective function values
$\xi(\mathbf{x}), \mathbf{x} \in A$	a statistical model of $f(\mathbf{x})$, e. g. a stochastic function
$\mathbb{P}(\cdot\|\cdot)$	the conditional probability
$\mathbb{E}(\cdot\|\cdot)$	the conditional expectation
$m(\mathbf{x}\|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$	the conditional mean of a random variable $\xi(\mathbf{x}), \mathbf{x} \in A$, with respect to the previous objective funtion evaluations
$s^2(\mathbf{x}\|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$	the conditional variance of a random variable $\xi(\mathbf{x}), \mathbf{x} \in A$, with respect to the previous objective function evaluations

S_i	a simplex
R	a hyper-rectangle
D	a hyper-rectangular decomposition of the feasible region
$V(R)$	the volume of a hyper-rectangle
$v = \min_{R \in D} V(R)$	the minimum volume of hyper-rectangles in the current decomposition
$\eta(R)$ or $\hat{\eta}(R)$	the criterion computed for the hyper-rectangle R
$\rho(\tau), \tau \geqslant 0$	a correlation function of a stationary isotropic stochastic function
$\| \cdot \|_2$	Euclidean norm
$\mathbb{1}_S(\mathbf{x})$	Indicator function for a set S

Chapter 1

Introduction

1.1 Research Context

A broad range of applications depends on selecting a collection of certain decision variables corresponding to the best value of some quantifiable objective, e. g. the price of production items [39]. The interest usually lies in finding the alternative that is the best globally, as contrasted to being optimal only in a certain subset of the feasible region which is considered unsatisfactory. This problem is known as the global optimization problem and can be formulated as follows:

$$\min_{\mathbf{x} \in A} f(\mathbf{x}), \tag{1.1}$$

where $f(\mathbf{x})$ is the objective function defined over the feasible region A and depending on the decision vector $\mathbf{x} \in A$. The case $A = \{\mathbf{x} \in \mathbb{R}^d : a_i \leqslant x_i \leqslant b_i, i = 1, \dots, d\}$ is addressed in this thesis. Moreover, $f(\mathbf{x})$ that is non-convex is of special interest. Usually the problem (1.1) is approached by numerical algorithms, as the analytical solutions are known to exist only in exceptional cases.

The problem (1.1) is difficult in two respects. First, making the distinction between the global and local minimum points is complicated from a theoretical point of view. Second, from the computational viewpoint, the presence of many other local minima besides the global one greatly complicates its discovery.

There are situations when a single objective does not completely define the choice of the decision vector as several conflicting goals are involved. The problem statement is generalized to a set of objectives $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$, $k \geqslant 2$, $\mathbf{x} \in A$:

$$\min_{\mathbf{x} \in A} (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})). \tag{1.2}$$

In general case, due to the conflicting nature of the objectives, there does not exist a single decision vector $\mathbf{x}^*$, resulting in the best possible values of all objectives at once. Instead, a solution to (1.2) is represented by a set of optimal compromises between the objectives. The solutions are required to comply with the concept of Pareto optimality, which means that for a given solution no objective could be improved without sacrificing some other objective. The resulting set of decision vectors is called the Pareto set. In practice, obtaining a full Pareto set is problematic, since its analytical expression can rarely be derived and, therefore, a numerical approximation is sought.

There exist various methods for solving the described optimization problems using both theoretical and heuristic reasoning. Depending on the method, certain assumptions are made about the problems (1.1) and (1.2). In this thesis, the attention is confined to the optimization methods which employ statistical and Lipschitz objective function models. They are applicable in a rather common engineering situation where the objective function is available as a *black-box* computer code that produces a value for a given input, but its analytical expression is unknown.

In case the *black-box* function involves a time-consuming simulation or a complex experiment, the optimization process must use costly objective evaluations rationally to ensure a satisfactory result with minimum resources. In this situation, it is beneficial to exploit a statistical objective function model. After some function evaluations have already been obtained, the statistical model can be viewed as a predictor of function values with an associated uncertainty measure at locations yet unexplored. These characteristics are then used by a global optimization algorithm to assess the suitability of $\mathbf{x} \in A$ for becoming the next trial point, meaning, however, that an expensive auxiliary optimization problem over A needs to be solved at each step of the algorithm. Two broad research directions of such an approach have emerged since its introduction by Kushner [53] in 1962. The first direction covers approximations of the optimal Bayesian algorithm, ensuring the minimal expected error in n steps ahead with respect to the chosen statistical model. The second direction consists of variations of the *P-algorithm*, developed in the axiomatic framework of the rational choice theory and defined as a sequence of rational choices under uncertainty. The state-of-the-art developments strive to expand the applicability of the methods by working around the expensive auxiliary optimization problems in the original form of the algorithms. Statistical models adjusted for the feasible region decomposition into simplices or hyper-rectangles are promising in this respect because computations can be simplified and theoretical investigation conducted.

In the *black-box* situation, only minimal assumptions regarding the objective function can be made. Among the mildest ones is the key assumption, used in the Lipschitz global optimization, that the objective function rate of change is bounded. Objective functions satisfying this assumption are said to comply with the Lipschitz model. The bounded rate of change assumption is rather realistic in real-world applications, moreover, it is attractive in many respects. For example, it helps to conduct a theoretical investigation, prove convergence properties of algorithms and define meaningful stopping conditions based on the proximity of the global minimizer. Furthermore, Lipschitz optimization methods are usually deterministic requiring no repeated runs. The main problem of the model is a generally unknown Lipschitz constant, i. e. the rate-of-change bound. The univariate single-objective Lipschitz optimization is well explored theoretically, and very good optimization algorithms are known, including the famous *Pijavskij-Shubert* algorithm. Straightforward generalizations of the best ideas to the multivariate case did not prove equally attractive. As a result, the multivariate case is under investigation, involving variations within the branch-and-bound framework with both hyper-rectangular and simplicial subsets. Among the most successful algorithms is the *DIRECT* algorithm, relying on a hyper-rectangular decomposition of the feasible region and the way of considering a range of rate-of-change constants simultaneously. Based on these core ideas, state-of-the-art methods aiming to accelerate the approximation of the global minimum are developed.

1.2 Relevance of the Study

Various statistical models were proposed in the context of global optimization, such as Gaussian stochastic functions or decomposition-adjusted statistical models. The choice of the model should be based on the a priori available information about the problem at hand, as well as computational complexity considerations. Therefore a systematic investigation of the impact that different statistical models have on the optimization results of objectives with various characteristics, would be useful. Situations when the assumed model precisely corresponds to the actual objective function, as well as when it does not, are of special interest.

Optimization methods using statistical objective function models devote considerable effort planning the trial points, that makes them suitable for optimizing expensive objective functions. The high cost of the objective function evaluations motivates to reduce the number of trials as much as possible to achieve the global minimum approximation of desired accuracy. A recent algorithm, rooted in statistical theory of global optimization, and relying on an iteratively refined hyper-rectangular feasible region decomposition, is a state-of-the-art method with theoretically established convergence properties. However, when it comes to practical performance, the discovery of the global minimizer is impeded by the presence of other local minimizers. The algorithm places many trials in their vicinity, wasting resources. As these trials are costly, it is necessary to find ways of preventing such behaviour.

A simplicial statistical model was used in a recent global optimization algorithm, where the feasible region decomposition is obtained by means of the Delaunay triangulation. The algorithm arises interest as the authors provide its convergence rate. The algorithm operates by selecting a simplex to be partitioned at each iteration, based on a relatively simple heuristic criterion. The intuition behind the criterion suggests that it should relate to the probability of finding a better function value inside the considered simplex. Formalizing this conjecture would provide a theoretical justification for the heuristic criterion in question.

Recently single-objective Lipschitz optimization algorithms based on adaptive diagonal partitions proved successful for both univariate and multivariate problems. These methods apply a trisection procedure to divide hyper-rectangles. Moreover, the same trisection procedure was used in the bivariate bi-objective Lipschitz optimization. A theoretically interesting task is to demonstrate the optimality of the trisection procedure in question, starting from a relatively simple case of the univariate bi-objective Lipschitz optimization. Although, in general, the applicability of optimal algorithms is narrow, they can be used as benchmarks for comparison; moreover, some of their properties might be shared by other practically applicable algorithms.

1.3 Objectives and Tasks of the Thesis

The present thesis aims at efficient and theoretically justified search for optimal solutions. Analysis of theoretical properties, as well as heuristic extensions in the single-objective global optimization using statistical models are provided. Furthermore, the worst-case optimality in the univariate bi-objective Lipschitz optimization is investigated.

The objectives of the study are:

1. Investigate the effect of the assumed statistical model on the performance of global optimization algorithms.
2. Increase the global minimum approximation efficiency in terms of the number of function evaluations of a recent global optimization algorithm, relying on a hyper-rectangular decomposition-adjusted statistical objective function model.
3. Theoretically support the definition of a simplicial global optimization algorithm, using properties of a simplicial statistical model.
4. Investigate the worst-case optimality in univariate bi-objective Lipschitz optimization.

The following tasks were identified:

1. Propose an experimental methodology and develop guidelines for the selection of a statistical model to be used for constructing classical global optimization algorithms.

2. Suggest ways to hybridize the recent global search strategy with local search techniques, allowing to increase the efficiency of approximating the global minimum in terms of the number of function evaluations.

3. Investigate asymptotic properties of a simplicial statistical model and use them to establish a theoretical link between the improvement probability-related criterion and a heuristically defined, but efficiently computable simplex selection criterion in the simplicial global optimization.

4. Investigate the one-step worst-case optimal trisection of an interval in the univariate bi-objective Lipschitz optimization and implement a corresponding optimal algorithm.

1.4 Scientific Novelty and Results

An experimental methodology was proposed aimed at a systematic assessment of the impact that the selected statistical model of the objective function has on the optimization of objective functions with various characteristics. Guidelines for the model selection were formulated based on the experimental results with a number of 1- and 2-dimensional Gaussian stochastic functions considered as statistical models used to construct optimization algorithms as well as generate objective functions.

Two heuristic extensions for a recent global search algorithm, iteratively refining a hyper-rectangular decomposition of the feasible region based on a statistical objective function model, were proposed, as an attempt to obtain an approximation of the global minimum in a moderate number of function evaluations. The suggested modifications combine global and local search techniques. The experimental comparison demonstrates that the resulting modifications perform better than the original and other contemporary algorithms under consideration.

Theoretical support for using a heuristically defined simplex selection criterion in a recent simplicial global optimization algorithm was provided. This was achieved by establishing a link between the criterion in question and the probability of improvement-related criterion. The latter corresponds to the case when a stationary isotropic Gaussian random field, defined over a simplex with known objective function values at the vertices, is used as a statistical model of the objective function.

The one-step worst-case optimal trisection of an interval in the univariate bi-objective Lipschitz optimization was investigated with respect to the tolerance of the local lower Lipschitz bound. It was theoretically shown that trisection of an interval into three equal parts satisfies the considered definition of optimality. A corresponding optimal bi-objective optimization algorithm was implemented and its performance was demonstrated.

1.5 Statements to Be Defended

1. The *P-algorithm*, constructed assuming a stationary isotropic Gaussian stochastic function with an exponential correlation, performs the best for a variety of univariate and bivariate objective functions.

2. The *Maximum expected improvement algorithm*, constructed assuming a stationary isotropic Gaussian stochastic function with a correlation structure ensuring low short-range variability, is appropriate to use for optimizing relatively simple objective functions.

3. Balancing the local and global search strategies in proposed hybrid algorithms *GB* and *Cluster* consumes fewer function evaluations compared to the original global optimization algorithm *Rect-1* that they extend, when difficult multi-modal global optimization problems are optimized.

4. A heuristically defined simplex selection criterion is asymptotically equivalent to a probability of improvement-related expression.

5. The trisection of an interval into three equal parts in the univariate bi-objective Lipschitz optimization is one-step worst-case optimal.

1.6 Structure of the Thesis

The thesis includes 6 chapters, general conclusions and 3 appendixes, covering conditional characteristics of the Gaussian stochastic functions, parameter estimation statistics for an experimental investigation in Chapter 3 and the proofs of the lemmas and the theorem of Chapter 6. Additionally, the bibliography and a list of publications by the author of this thesis are included. This thesis contains 159 pages with 19 figures, 23 tables and 4 algorithms.

Chapter 2

Global Optimization Based on Statistical and Lipschitz Objective Function Models

In this chapter we present a survey of the developments in two broad research directions in the global optimization of continuous problems, namely methods involving statistical and Lipschitz models of the objective functions.

Both types of models are applicable in a situation where the objective function is given as a *black-box* computer code that provides no information other than output values for given input vectors, and only minimal assumptions regarding it are possible.

2.1 The Global Optimization Problem

The formal statement of the unconstrained global optimization problem is

$$\min_{\mathbf{x} \in A} f(\mathbf{x}), \tag{2.1}$$

where $f(\mathbf{x})$ is the *objective function* and A is a hyper-rectangular *feasible region*, i. e. $A = \{\mathbf{x} \in \mathbb{R}^d : a_i \leqslant x_i \leqslant b_i, i = 1, \ldots, d\}$. Alternatively, A can be called the *search space* or the *domain*. The objective function $f(\mathbf{x})$ is said to be *black-box* if it is possible to evaluate it on the points of the domain, however, its analytical expression is unavailable.

The notation for the *global minimum* $f^* = \min_{\mathbf{x} \in A} f(\mathbf{x})$ and the set of *global minimizers* $X^* = \{\mathbf{x}^* : f(\mathbf{x}^*) = f^*\}$ is used in the discussion that follows.

A *global minimization algorithm* is a procedure of generating a sequence of locations $\mathbf{x}_i, i = 1, \ldots, n,$ such that an approximation of the global minimum f^*

$$y_{on} = \min_{i=1,\ldots,n} f(\mathbf{x}_i) \tag{2.2}$$

tends to f^*, as n tends to infinity. The value y_{on} is sometimes called the *record*. In addition, an approximation of some *global minimizer* $\mathbf{x}^* \in X^*$ is produced by the algorithm as well.

The points $(\mathbf{x}_i, y_i), y_i = f(\mathbf{x}_i), i = 1, \ldots, n,$ are called the *objective function evaluations* or *trials*.

The locations $\mathbf{x}_i$, $i = 1, \ldots, n$, are referred to as *objective function evaluation points* (*locations*) or *trial points*. In the univariate case, it is sometimes useful to consider the ordered trial points x_i^n, $i = 1, \ldots, n$, satisfying $x_j^n < x_k^n$, if $j < k$.

According to the way that the trial points $\mathbf{x}_i, i = 1, \ldots, n,$ are obtained, the algorithms are classified as either *passive* (*non-adaptive*) or *sequential* (*adaptive*). In the first case, the points $\mathbf{x}_i \in A, i = 1, \ldots, n,$ are fixed a priori, before the optimization process starts. In the second case, the selection of each new location $\mathbf{x}_{n+1}$ is governed by a certain function of the already obtained results, i. e. $(\mathbf{x}_i, y_i), i = 1, \ldots, n.$

2.2 Global Optimization Based on Statistical Models

The case when a *black-box* function $f(\mathbf{x})$ is expensive or time-consuming to evaluate, calls for a need to carefully plan objective function evaluations. Examples of such problems include the search for the best design options in engineering, as well as optimization of non-linear control systems, involving time-consuming simulations.

The global optimization approach discussed in the present section employs a statistical model $\xi(\mathbf{x}), \mathbf{x} \in A$, of the objective function $f(\mathbf{x}), \mathbf{x} \in A$, in the described situation. The statistical model can be viewed as an approximation of the objective function, built using the already performed function evaluations $(\mathbf{x}_i, y_i), y_i = \xi(\mathbf{x}_i) = f(\mathbf{x}_i)$, $i = 1, \ldots, n$. The model provides a conditional predictor of the function values $m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$ at locations $\mathbf{x} \in A, \mathbf{x} \neq \mathbf{x}_i, i = 1, \ldots, n$, yet unexplored, as well as a measure of uncertainty $s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$ in the result of the prediction. Depending on the specific statistical model used, the computation of these characteristics has a different level of complexity. The most common statistical models are introduced in Section 2.2.1.

In essence, the global optimization algorithms relying on a statistical objective function model assess the suitability of $\mathbf{x} \in A$ to become the next trial point, based on a certain criterion, involving the respective predictor and uncertainty values at $\mathbf{x}$. Originally, the idea of using a statistical objective function model in global optimization appeared in [53]. The development of the respective algorithms evolved in two main directions. The first direction includes approximations of the optimal Bayesian algorithm, defined in [66] and ensuring the minimal expected error in n steps ahead with respect to the chosen statistical model. The second direction consists of further developments of the so-called *P-algorithm*, axiomatically defined in the framework of rational choice theory [95]. These algorithms are described in Section 2.2.2.

The term *surrogate model* of the objective function is used by some authors [24, 47] to refer generally to possible predictors of the function values, including statistical models as a subcategory, alongside polynomial interpolators, regression models, etc. The use of a statistical model of a specific structure [75] in optimization is referred to as *kriging* by the same authors, however, it should be regarded a special case of the more general approach considered in this thesis.

For the information on the earliest developments in the field the reader is referred to the monographs [67, 89, 93, 96, 105], whereas more recent developments are covered in [85, 103, 109].

2.2.1 Common Statistical Models

The present section aims to introduce a variety of statistical models used in the literature to model the objective functions. The choice of the specific statistical model is usually based on the previous practice, subjective assessment of its adequacy to the problem and computational complexity considerations.

The basics of the stochastic functions, which were the first statistical models used in global optimization, are recalled. The most notable representatives of this class of models are the stationary isotropic Gaussian random fields and the Wiener process, considered in Chapter 3.

Further, a generalized statistical model, originally developed in [108], is described, which allows to model the objective function as a more general family of random variables than that embodied by a stochastic function. This generalization allowed to overcome the computational restrictions emanating from the use of stochastic functions as models in the multi-dimensional case.

Finally, decomposition-adjusted statistical models, defined in the context of global optimization based on simplicial and hyper-rectangular decompositions of the feasible region and derived from the generalized statistical models, are described. The algorithms based on the decomposition-adjusted models are the subject of Chapters 4 and 5 of this thesis.

2.2.1.1 Basics of Stochastic Functions

A model of a function under uncertainty in probability theory is the stochastic function, making it a natural choice for modelling a *black-box* objective function with unknown properties.

Let us start with some basic definitions, provided in [56], complying with the customary notation in the probabilistic literature.

A *stochastic (or random) process* is formally defined to be a collection of random variables defined on a common probability space $(\Omega, \mathcal{F}, P)$ and indexed by the elements of a *parameter set* T. It is customary to perceive $t \in T$ as time. If $T = \mathbb{R}^d$ with $d > 1$, the process is referred to as a *random field*. The random variables of the process must have the same measurable *range space* S.

Indeed the process is a function of two variables, say, $\xi = \xi(t, \omega)$, where $t \in T, \omega \in \Omega$, and where for each fixed t the function $\xi(t, \cdot)$ is measurable with respect to $\mathcal{F}$. If instead of t we fix an $\omega \in \Omega$, we obtain a function $\xi(\cdot, \omega) : T \to S$ which is called a *realization* (or, alternatively, a *sample function*) of the process. The process ξ might be thought of as a single random variable taking values in a space of the functions on T, as emphasized by the term *stochastic (random) function*.

The latter interpretation is precisely the one used in the context of global optimization, expressed as the problem (2.1). Let the parameter set $T = A \subset \mathbb{R}^d$, and the indexing variable be denoted by $\mathbf{x}$ instead of t. Moreover, let the range space $S = \mathbb{R}$. Then the objective function under uncertainty $f(\mathbf{x})$ is modeled as a realization $\xi(\cdot, \omega) : A \to \mathbb{R}$ of the stochastic function $\xi(\mathbf{x}, \omega) : A \times \Omega \to \mathbb{R}$. The simplified notation $\xi(\mathbf{x}), \mathbf{x} \in A$, will be used throughout this thesis.

A simplifying assumption that the probabilistic structure of a stochastic function is invariant under transformations of $A = \mathbb{R}^d$, such as translation, rotation and reflection, is expressed through the notions of *stationarity* and *isotropy* of the random field, as defined in [87].

Let the *mean* and *covariance functions* of $\xi(\mathbf{x})$ be defined, respectively, as

$$\mu(\mathbf{x}) = \mathbb{E}(\xi(\mathbf{x})), \tag{2.3}$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = \mathbb{E}((\xi(\mathbf{x}_1) - \mu(\mathbf{x}_1))(\xi(\mathbf{x}_2) - \mu(\mathbf{x}_2))). \tag{2.4}$$

If a function $K_1 : \mathbb{R}^d \to \mathbb{R}$ exists such that $K(\mathbf{x}_1, \mathbf{x}_2) = K_1(\mathbf{x}_1 - \mathbf{x}_2)$, $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, moreover, the random field possesses finite second moments and a constant mean function $\mu(\mathbf{x}) = \mu$, it is called *weakly stationary*.

Further, if a function $K_2 : [0, +\infty) \to \mathbb{R}$ exists such that $K(\mathbf{x}_1, \mathbf{x}_2) = K_1(\mathbf{x}_1 - \mathbf{x}_2) = K_2(\|\mathbf{x}_1 - \mathbf{x}_2\|_2)$, $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, where $\|\cdot\|_2$ denotes the Euclidean norm, and $\mu(\mathbf{x}) = \mu$, the random field is called *weakly isotropic*. A weakly isotropic random field is always weakly stationary.

The correlation function $\rho : [0, +\infty) \to \mathbb{R}$ of a weakly isotropic random field is denoted by $\rho(\tau) = \rho(\|\mathbf{x}_1 - \mathbf{x}_2\|_2) = K_2(\|\mathbf{x}_1 - \mathbf{x}_2\|_2)/K_2(0), K_2(0) > 0, \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, $\tau \in [0, +\infty)$.

A real-valued *Gaussian stochastic function* is a stochastic function $\xi(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$ for which the finite-dimensional distributions of $(\xi(\mathbf{x}_1), \dots, \xi(\mathbf{x}_n))$ are multivariate Gaussian for each $1 \leqslant n \leqslant +\infty$ and each $(\mathbf{x}_1, \dots, \mathbf{x}_n) \in (\mathbb{R}^d)^n$.

Every Gaussian stochastic function $\xi(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$ is strictly stationary (isotropic) if and only if it is weakly stationary (isotropic) [87]. Therefore, the modifier *weak* will be omitted referring to the stationary isotropic Gaussian random fields.

The Gaussian stochastic function is determined by the mean and covariance functions. Given any set A, a function $\mu : A \to \mathbb{R}$, and a non-negative definite function $K : A^2 \to \mathbb{R}$, there exists a Gaussian stochastic function $\xi(\mathbf{x}), \mathbf{x} \in A$ with the mean function $\mu(\mathbf{x})$ and the covariance function $K(\mathbf{x}_1, \mathbf{x}_2)$ [5]. Alternatively, one can specify the mean function $\mu(\mathbf{x})$ (constant μ for stationary isotropic Gaussian processes), standard deviation $\sigma = \sqrt{K_2(0)}$ and the correlation function $\rho(\tau), \tau \in (0, +\infty]$ to define a Gaussian stochastic function.

When some values $\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n$ of a realization of a Gaussian stochastic function are known, the conditional distribution of $\xi(\mathbf{x}), \mathbf{x} \neq \mathbf{x}_i$ is also Gaussian with the conditional mean and variance, respectively, denoted by $m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n)$ and $s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n)$.

Consider a stationary isotropic n-dimensional Gaussian random field with mean μ, standard deviation σ and a correlation function $\rho(\tau), \tau \geqslant 0$. The conditional characteristics can be expressed as:

$$m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n) = \mu + \Sigma^T C^{-1}(\mathbf{y} - m\mathbf{I}), \tag{2.5}$$

$$s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n) = \sigma^2(1 - \Sigma^T C^{-1}\Sigma), \tag{2.6}$$

where $\mathbf{y} = (y_1, \dots, y_n) = (\xi(\mathbf{x}_1), \dots, \xi(\mathbf{x}_n))$, $\Sigma = (\rho(\|\mathbf{x} - \mathbf{x}_1\|_2, \dots, \rho(\|\mathbf{x} - \mathbf{x}_n\|_2)^T$, $C = (\rho(\|\mathbf{x}_i - \mathbf{x}_j\|_2))_{ij}, i, j = 1, \dots, n$, $\mathbf{I} = (1, \dots, 1)_n^T$.

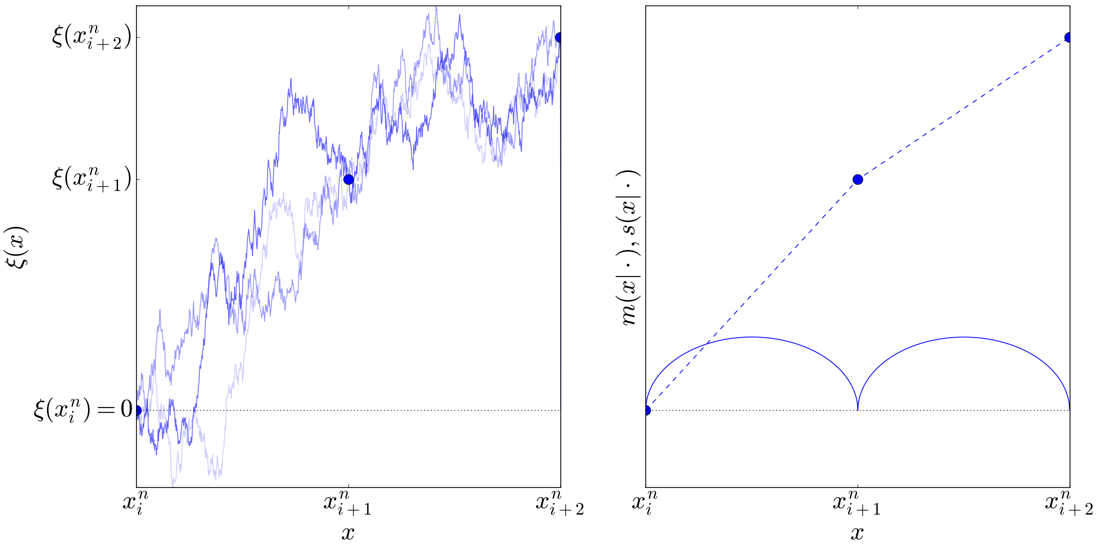

Figure 2.1: An example of assuming that the objective function is a realization of the Wiener process. **Left:** three possible realizations of the Wiener process passing through the points $(x^n_j, \xi(x^n_j))$, $j = i, i+1, i+2$, corresponding to the performed objective function evaluations. **Right:** Graphs of conditional mean $m(x|\xi(x_i) = y_i, i = 1, \dots, n)$ (dashed) and standard deviation $s(x|\xi(x_i) = y_i, i = 1, \dots, , i = 1, \dots, n)$ (solid) of the Wiener process.

2.2.1.2 The Wiener Process

The first statistical model of the objective function known in the global optimization literature was the Wiener process (also known as Brownian-motion process), as used in the seminal paper by Kushner [53].

The Wiener process is a 1-dimensional Gaussian stochastic function $\xi(x), x \in [0, +\infty)$ with stationary independent increments such that $\xi(0) = 0$, $\xi(x) \sim N(0, \sigma^2 x)$, where σ is a parameter. The stationarity and independence of increments means that for $x^n_1 < \cdots < x^n_n$, $x^n_i \in [0, +\infty)$, the random variables $\xi(x^n_{i+1}) - \xi(x^n_i)$, $i = 1, \dots, n-1$, $n \in \mathbb{N}$, are mutually independent and have a joint distribution which is unchanged if each x^n_i is replaced by $x^n_i + h$, $h \in \mathbb{R}$ [56]. The covariance of the Wiener process is $K(x^n_i, x^n_j) = \sigma^2 x^n_i$, $\forall i < j$.

Figure 2.1 illustrates an assumption that the objective function is a realization of the Wiener process. The dots denote three performed objective function evaluations $(x^n_j, \xi(x^n_j))$, $j = i, i+1, i+2$. An infinite number of the Wiener process realizations might pass through these fixed points out of which three possible sample functions are shown on the left subfigure. The values of $\xi(x)$ over the intervals (x_i^n, x_{i+1}^n) and (x_{i+1}^n, x_{i+2}^n) are unknown and are modeled by conditional probability distributions with mean and standard deviation functions $m(x|\xi(x_i) = y_i, i = 1, \ldots, n)$ and $s(x|\xi(x_i) = y_i, i = 1, \ldots, , i = 1, \ldots, n)$, respectively. $m(x|\cdot)$ corresponds to the piece-wise linear dashed graph on the right subfigure. It passes through the points $(x_j^n, \xi(x_j^n)), j = i, i+1, i+2$. $s(x|\cdot)$ is displayed by the solid line on the right subfigure. The uncertainty described by $s(x|\cdot)$ equals to zero at the known evaluation locations and reaches the maximum value at the midpoints of the intervals (x_i^n, x_{i+1}^n) and (x_{i+1}^n, x_{i+2}^n).

The applicability of the Wiener process as a model of a univariate objective function is disputable due to the fact that the realizations of this process are continuous everywhere but differentiable nowhere. This contradicts the nature of objective functions arising in practice.

However, the independence of increments over disjoint intervals is an acceptable assumption for complicated optimization problems with many local minima [106]. Moreover, the process is computationally attractive, since it possesses the Markov property. From the global optimization viewpoint, this results in the computational simplifications, as for ordered trial points x_i^n, $i = 1, \ldots, n$, the conditional characteristics of $x \in [x_j^n, x_{j+1}^n]$, $j = 1, \ldots, n-1$, depend on $(x_j^n, \xi(x_j^n))$ and $(x_{j+1}^n, \xi(x_{j+1}^n))$ only:

$$m(x|\xi(x_i^n) = y_i^n, i = 1, \ldots, n) = m(x|\xi(x_i^n) = y_i^n, i = j, j+1), \tag{2.7}$$

$$s^2(x|\xi(x_i^n) = y_i^n, i = 1, \ldots, n) = s^2(x|\xi(x_i^n) = y_i^n, i = j, j+1). \tag{2.8}$$

Expression (2.7) is piecewise linear, while (2.8) is piecewise quadratic (see Appendix A).

A further advantage of assuming the Wiener process as an objective function model is its attractiveness to analytical treatment of resulting global optimization algorithms [12, 15, 106]. However, there are no global optimization algorithms relying on generalizations of the Wiener process to higher dimensions.

All things considered, the Wiener process should serve as a global description of the objective function but not as a local one. For example, a dual model using a quadratic local description of the objective function was introduced in [94]. Moreover, the non-differentiability issue was addressed in [14], where the integrated Wiener process was employed.

2.2.1.3 A Generalized Statistical Model

The applicability of stochastic functions as models of objective functions is limited in the multivariate case due to the need to invert a correlation matrix of dimensionality $n \times n$, where n is the number of already performed trials, in order to compute the characteristics of the conditional distribution of $\xi(\mathbf{x}), \mathbf{x} \neq \mathbf{x}_i, i = 1, \ldots, n,$ in (2.5) and (2.6). This operation is required to determine every new trial point $\mathbf{x}_{n+1}$ in algorithms discussed in Section 2.2.2.

To address this problem, the validity of using a general family of random variables $\xi(\mathbf{x}), \mathbf{x} \in A$, as a model of $f(\mathbf{x})$ was established axiomatically, as described in [93]. Typically a priori information about $f(\mathbf{x})$ arising from the experience of solving similar engineering problems in the past does not contradict a set of comparative probability axioms, i. e. certain assumptions regarding the possible value intervals of $f(\mathbf{x})$. The compliance of $f(\mathbf{x})$ with these axioms implies the existence of a unique probability density of a random variable $\xi(\mathbf{x}), \mathbf{x} \in A$, making it suitable to model $f(\mathbf{x}), \mathbf{x} \neq \mathbf{x}_i, i = 1, \ldots, n$. Due to computational as well as intuitive reasons, $\xi(\mathbf{x})$ are defined to be the Gaussian random variables. The stochastic functions could be considered a special case of the new generalized model.

Further, the conditional characteristics of the generalized model, $m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$ and $s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)$, were defined in [107] based on the axioms of rationality of extrapolation under uncertainty. $m(\mathbf{x}|\cdot)$ can be viewed as a predictor of values at unexplored locations, while $s^2(\mathbf{x}|\cdot)$ is interpreted as an extent of deviation from the predicted value. The only extrapolators compatible with the axioms are

$$\begin{aligned} m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n) &= \sum_{i=1}^{n} w_i(\mathbf{x}, \mathbf{x}_i, i = 1, \ldots, n) \cdot y_i, \\ s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n) &= \gamma_n \sum_{i=1}^{n} w_i(\mathbf{x}, \mathbf{x}_i, i = 1, \ldots, n) \|\mathbf{x} - \mathbf{x}_i\|_2, \end{aligned} \tag{2.9}$$

where $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, are the trials performed, $w_i(\cdot)$ are some weights defined by heuristic reasoning and experimental results, and $\gamma_n > 0$ might depend on the trials performed.

For a special case of $\xi(\mathbf{x})$ being a stochastic function, expressions (2.9) are equivalent to its conditional mean and variance. Otherwise they are easier to compute as the definition of weights can avoid matrix inversion.

2.2.1.4 Decomposition-Adjusted Statistical Models

Let us recall from [39] that for $A \subset \mathbb{R}^d$ and a finite set of indices I, a set $D = \{S_i : i \in I\}$ of subsets of A is called a *decomposition* (or *partition*) of A if

$$A = \bigcup_{i \in I} S_i \text{ and } S_i \cap S_j = \delta S_i \cap \delta S_j, \forall i, j \in I, i \neq j, \tag{2.10}$$

where δS_i denotes the boundary of S_i .

Building upon the generalized statistical models of $f(\mathbf{x})$, discussed in Section 2.2.1.3, an idea to decompose the feasible region A into simplices S_i over which independent statistical models of $f(\mathbf{x})$ are constructed, was presented in [110], followed by later developments in [16, 113, 114]. Similarly, statistical model versions for the hyper-rectangular decompositions instead of the simplicial ones were defined in [11, 30].

The approach provided a way to break an auxiliary optimization problem over A, defining the next trial point $\mathbf{x}_{n+1}$ (see Section 2.2.2), down into subproblems over subsets of A, that can be solved individually. Consequently, it can be viewed as elimination of the main computational obstacles to the wider application of the optimization algorithms based on the statistical models of objective functions.

In the case of simplicial subsets S_i, the model of $f(\mathbf{x})$ as a family of Gaussian random variables $\xi(\mathbf{x})$ with conditional characteristics $m(\mathbf{x}|\cdot)$ and $s^2(\mathbf{x}|\cdot)$ is defined over a simplex. The collection of all such simplices, spanning the entire feasible region, provides the statistical model of $f(\mathbf{x}), \mathbf{x} \in A$. The analogue to the Markov property of the Wiener process and the linearity with respect to $\mathbf{x}$ is ensured, when the mean of $\xi(\mathbf{x})$ is defined as

$$m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n) = \sum_{i=1}^{d+1} \nu_i(\mathbf{x}, \boldsymbol{\omega}_j, j = 1, \ldots, d+1) \cdot \phi_i, \tag{2.11}$$

where $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, are the performed trials, $(\boldsymbol{\omega}_i, \phi_i)$, $\phi_i = \xi(\boldsymbol{\omega}_i), i = 1, \ldots, d+1$, are the trials at the vertices of the simplex and $\nu_i(\cdot)$ are the weights, obtained from the equality $\mathbf{x} = \sum_{i=1}^{d+1} \nu_i(\mathbf{x}, \boldsymbol{\omega}_j, j = 1, \ldots, d+1) \cdot \boldsymbol{\omega}_i$.

Denoting by $\mathbf{x}_S$ the point equidistant to the vertices of the simplex and by Δ_S the corresponding distance, the quadratic function of variance is defined as

$$s^2(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n) = \sigma^2 \cdot (\Delta_S^2 - \|\mathbf{x}_S - \mathbf{x}\|^2), \tag{2.12}$$

where σ is a parameter. $s^2(\mathbf{x}|\cdot)$ vanishes at the vertices and is maximum at $\mathbf{x}_S$, analogously to that of the Wiener process.

2.2.2 Seminal Conceptual Algorithms

This section introduces two conceptual algorithms, namely the *Maximum expected improvement algorithm* (the name *One-step Bayesian algorithm* could be used as an equivalent) and the *P-algorithm*, employing statistical models of the objective function in the process of the search for the global minimum. The former originated as a practically feasible simplification of an optimal Bayesian algorithm [66], optimal with respect to the class of stochastic functions as statistical models of $f(\mathbf{x})$. The latter appeared in the first paper [53], modeling $f(\mathbf{x})$ as the Wiener process and was later theoretically justified as a rational algorithm by an axiomatic definition in [95]. These are the seminal algorithms further developments of which constitute the body of literature on statistical models in global optimization.

Let a family of random variables $\xi(\mathbf{x}), \mathbf{x} \in A$, be assumed as a statistical model of $f(\mathbf{x})$. In both algorithms a new point for the function evaluation is selected, optimizing some statistically-defined criterion based on the distribution of $\xi(\mathbf{x})$, conditioned on previously performed function evaluations. Some background information and general expressions for both of these algorithms are given in the present section. Note that specific versions of the algorithms can be obtained by fixing a certain statistical model $\xi(\mathbf{x}), \mathbf{x} \in A$, for instance, a Gaussian stochastic function. Moreover, each single implementation is influenced by the selected way of solving the auxiliary optimization problems in these equations. Finally, it is important to stress the fact that the algorithms to be presented shortly provide a deterministic result for a deterministic objective function, since the assumed statistical model is only used to motivate the design of the algorithms, but the steps of the algorithms are not randomized.

2.2.2.1 The Maximum Expected Improvement Algorithm

A simplification to a global optimization algorithm, introduced in [66] as an optimal Bayesian algorithm, possessing the property of the average-case optimality with respect to a statistical model $\xi(\mathbf{x}), \mathbf{x} \in A$, led to the development of the well-known *Maximum expected improvement algorithm*.

Specifically, let a class Z of algorithms be defined by the budget of n objective function evaluations. An algorithm $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_n) \in Z$ defines a sequence of function evaluations

$$\mathbf{x}_1 = \zeta_1, \tag{2.13}$$

$$\mathbf{x}_k = \zeta_k(x_i, \xi(\mathbf{x}_i), i = 1, \ldots, k-1), k = 2, \ldots, n, \tag{2.14}$$

and accepts $\mathbf{x}_{on}(\boldsymbol{\zeta}) \in A : \xi(\mathbf{x}_{on}) = \min_{i=1,\ldots,n} \xi(\mathbf{x}_i)$, as an approximation to the global minimizer. The algorithm ensuring the smallest expected error after n function evaluations is called the Bayesian algorithm and is defined by the following formula:

$$\hat{\boldsymbol{\zeta}} = \arg\min_{\boldsymbol{\zeta}} \mathbb{E}(\xi(\mathbf{x}_{on}(\boldsymbol{\zeta})) - \min_{\mathbf{x} \in A} \xi(\mathbf{x})), \tag{2.15}$$

where $\mathbb{E}(\cdot)$ denotes the expectation with respect to the probability measure of a stochastic function $\xi(\mathbf{x})$. Due to implementation complexity of the above algorithm, a simplification under the title *One-step Bayesian algorithm* was proposed in [66], formulated as

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}\in A} \mathbb{E}(\min(\xi(\mathbf{x}), y_{on})|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n), \tag{2.16}$$

where $\mathbb{E}(\cdot|\cdot)$ denotes the conditional expectation. An equivalent and more popular version of (2.16) is expressed in terms of the expected improvement over the current record value and referred to as the *Maximum expected improvement algorithm*:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in A} \mathbb{E}(\max(\xi(\mathbf{x}) - y_{on}, 0)|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n). \tag{2.17}$$

The algorithm (2.17) was popularized by the appearance of the paper [49], where its specific implementation was suggested, involving the *DACE* statistical model [75] (polynomial regression with additive stationary non-isotropic Gaussian noise) and the optimization of the expected improvement by means of a branch-and-bound technique. The resulting implementation was termed *Efficient global optimization algorithm (EGO)*, enjoying higher popularity, although obscuring the actual origins of the conceptual algorithm.

2.2.2.2 The P-algorithm

One possible theoretical justification of a global optimization algorithm design is the optimality of the resulting algorithm with respect to a certain criterion, e. g. the minimum expected error over the class of stochastic functions in the case of the optimal Bayesian algorithm [66].

An alternative approach to defining the global optimization algorithm was taken in [95] where the problem of the definition of a rational algorithm was addressed. An algorithm was formulated as a sequence of rational decisions under uncertainty according to the general practice in the theory of rational choice. The name *P-algorithm* was assigned to this algorithm.

The *P-algorithm* defines the next function evaluation location as

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in A} \mathbb{P}(\xi(\mathbf{x}) \leqslant y_{on} - \epsilon_n | \xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n), \tag{2.18}$$

where $\mathbb{P}(\cdot|\cdot)$ denotes the conditional probability. Equation (2.18) means that the *P-algorithm* places a new function evaluation at a point of the feasible region, where the probability to improve upon the current record y_{on} by a parameter $\epsilon_n > 0$ is the greatest. The balance of the global and local search can be regulated using parameter ϵ_n: the greater its value, the more global the search.

A 1-dimensional version of the *P-algorithm*, where $\xi(\mathbf{x})$ is the Wiener process, appeared in [53] as the first global optimization algorithm using a statistical objective function model.

The summary of an axiomatic justification of this algorithm in [95] follows. Assuming the generalized statistical model of $f(\mathbf{x})$ to be the family of Gaussian random variables $\xi(\mathbf{x})$ with conditional probability densities $p_{\mathbf{x}}(\cdot)$ with respect to previous trials $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, the choice of the next trial location $\mathbf{x} \in A$ can be interpreted as the choice between probability densities $p_{\mathbf{x}}(\cdot)$. If the relation of preference, denoted $p_{\mathbf{x}_1} \succeq p_{\mathbf{x}_2}$ (when $p_{\mathbf{x}_1}$ is not less preferable than $p_{\mathbf{x}_2}$), agrees with certain rationality axioms, there exists a utility function $u(t)$ such that

$$p_{\mathbf{x}_1} \succeq p_{\mathbf{x}_2} \iff \int_{-\infty}^{\infty} u(t)p_{\mathbf{x}_1}(t)dt \geqslant \int_{-\infty}^{\infty} u(t)p_{\mathbf{x}_2}(t)dt. \tag{2.19}$$

I. e. the expected value of utility of choosing $\mathbf{x}_1$ is not smaller than that of choosing $\mathbf{x}_2$, if one is to prefer $\mathbf{x}_1$ at least as much as $\mathbf{x}_2$ as a candidate for the next function evaluation location. It is proved that the only utility function that satisfies the rationality axioms is $u(t) = I(\hat{y}_{on} - t)$, where $\hat{y}_{on} < min_{i=1,\ldots,n} y_i$ and $I(\cdot)$ is a unit-step function. Thus, the next function evaluation should be performed at $\mathbf{x}_{n+1} = max_{\mathbf{x}} P(\xi(\mathbf{x}) < \hat{y}_{on})$.

Under certain general assumptions on the optimization problem and the statistical model, the *P-algorithm* converges to the global minimum, i. e. $y_{on} \to \min_{\mathbf{x}\in A} f(\mathbf{x})$. The required conditions are satisfied by the Wiener process and the generalized statistical model, discussed in Section 2.2.1.3 [93].

2.2.3 Further Developments

The conceptual algorithms, discussed in Section 2.2.2, continue to inspire a variety of emerging modifications, motivated primarily by decreasing the complexity and expanding the applicability of these methods.

The use of a statistical model necessitates the specification of its parameters, e. g. for a stationary isotropic Gaussian random field the mean as well as parameters of the covariance function have to be specified. A possible approach is to estimate these parameters from the already performed trials $(\mathbf{x}_i, y_i), i = 1, \ldots, n$. For example, the maximum likelihood estimates are used in [62, 75]. The estimation of parameters for small n is problematic. It is demonstrated in [51] that performance of the *Maximum expected improvement algorithm* is positively affected by applying the parametric bootstrapping to model parameter estimation.

The problem of noise in the computed values of $f(\mathbf{x})$ is addressed from the very beginning of the statistical model-based approach to global optimization in [53]. Implementation difficulties arising in such a case are addressed in [17]. In [42], a development of the *Maximum expected improvement algorithm* is proposed where the expected improvement criterion is augmented to account for random noise in the measured outputs of $f(\mathbf{x})$. Systematic errors, arising from the availability of multiple-fidelity versions of $f(\mathbf{x})$, are incorporated in the maximum expected improvement expression in [41].

The auxiliary optimization problems in (2.18) and (2.17) over the original domain A might be multimodal, imposing a need for a supplementary global optimization method to solve them. A divide-and-conquer approach, inspired by the algorithms of the branch-and-bound type, was adapted to address this problem in a series of papers [11, 16, 30, 110, 113, 114]. Specifically, the methods share the idea of decomposing the feasible region into a set of subsets (either simplices or hyper-rectangles) with known objective function values at the vertices. For each subset, an analogue to the improvement probability maximum is computed as a selection criterion. A subset with the highest criterion value is selected and partitioned at each step of the algorithm, refining the previous decomposition. Thus the auxiliary problem (2.18) is reduced to a set of problems over subsets and the need for an additional global optimization algorithm is eliminated. The resulting research direction seems to deserve further attention.

2.3 Global Optimization Based on Lipschitz Models

A real-valued function $f(\mathbf{x})$ is said to satisfy the Lipschitz condition, if

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leqslant L\|\mathbf{x}_1 - \mathbf{x}_2\|, \forall \mathbf{x}_1, \mathbf{x}_2 \in A, \tag{2.20}$$

where L is a constant (called Lipschitz constant), $A = \{\mathbf{x} \in \mathbb{R}^d : a_i \leqslant x_i \leqslant b_i, i = 1, \ldots, d\}$ is a hyper-rectangle, and the norm $\|\cdot\|$ might be Euclidean or a different one.

The theoretically attractive assumption (2.20) means that the rate of change of the function is bounded, facilitating the investigation of the convergence properties of the algorithms. The assumption is also rather realistic for practical *black-box* problems, resulting in applications of the respective algorithms to various real-world problems.

The condition (2.20) is usually exploited by deterministic algorithms, requiring no repeated runs. Moreover, meaningful stopping conditions can be defined based on the bounds on the proximity to the global minimum. However, the main problem of the model is the generally unknown Lipschitz constant L. Its overestimate might result in a search that is overly exhaustive, wasting computational resources while its underestimate might result in a missed global minimum. Moreover, it might differ over subregions of A. Furthermore, even if the Lipschitz constant is known, the search space might be too large to obtain a sufficiently precise solution. The algorithms either rely on an a priori supplied constant value or maintain its (global or local) estimate updated in the course of the optimization process, or consider a set of its possible values simultaneously.

Alternative global optimization problem formulations have been addressed in the literature. Since the problem of finding

$$\mathbf{x}^* \in A : f(\mathbf{x}^*) = f^*, \tag{2.21}$$

is solvable in a finite number of function evaluations only under very restrictive assumptions on $f(\mathbf{x})$ and in the ideal case of a known Lipschitz constant, an alternative problem statement requires to find

$$\mathbf{x}' \in A : f(\mathbf{x}') \leqslant f^* + \epsilon, \epsilon > 0. \tag{2.22}$$

Algorithms, providing a solution to (2.22) after a finite number of function evaluations, are called ϵ-convergent, while the respective point $\mathbf{x}'$ is called ϵ-optimal.

The present section describes the univariate and multivariate approaches to Lipschitz optimization. For a comprehensive treatment of the Lipschitz optimization methods, the reader is referred to [34, 39, 73].

2.3.1 The Univariate Case

Let us assume that the Lipschitz constant L and ordered objective function evaluations $(x_i^n, y_i^n), i = 1, \ldots, n$, are known. Then the lower-bounding function of $f(x)$

$$F_n(x) = \max_{i=1,\ldots,n} (y_i^n - L|x - x_i^n|), \tag{2.23}$$

is often referred to as the *saw-tooth cover* of $f(x)$ due to its shape. The successive points x_i^n and x_{i+1}^n are called the *basis points* of the tooth i. The minimum of $F_n(x)$ over $[x_i^n, x_{i+1}^n]$ and the corresponding point $\hat{x}_i$ are called the *height* and the *peak point* of the tooth i, respectively. Moreover, the *region of indeterminacy* is defined as the set, where the global minimum x^* might still be located, and corresponds to the union of the intervals $\bigcup_{i=1,\ldots,n}[\alpha_i, \beta_i]$ obtained by intersecting the horizontal line at height y_{on} with the saw-tooth cover. Figure 2.2 illustrates the introduced concepts.

The univariate Lipschitz optimization has been theoretically very well-explored and efficient optimization algorithms are known.

Let us cover some of the univariate ϵ-convergent algorithms, starting with the passive one that evaluates the objective function at equally-spaced locations

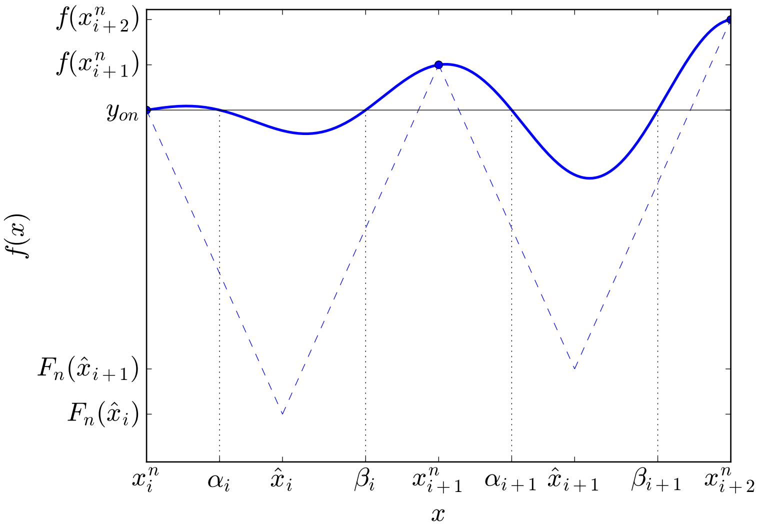

Figure 2.2: Illustration of the common concepts in the univariate Lipschitz optimization. The objective function $f(x)$ is displayed as a thick solid line, while the saw-tooth cover $F_n(x)$ is shown by a dashed line.

over $A = [a, b]$ in order to guarantee ϵ-convergence in $n = \lceil \frac{L(b-a)}{2\epsilon} \rceil$ objective function evaluations. The function is evaluated at $a + \frac{2k\epsilon}{L}$, $k = 1, \ldots, n$.

The *best possible* algorithm by Danilin [20, 21] depends on the knowledge of the global minimum value f^* which is generally unknown, however, it serves only to provide a minimum number of function evaluations n_B required to guarantee that the ϵ-optimal value is found. The obtained number serves to study the efficiency of other sequential optimization algorithms, for which the global minimum value is unknown. It is worth noting that no analogous benchmark algorithm is known in the multivariate case.

In the worst case scenario, which corresponds to the constant function, the *best possible* algorithm requires the same number of function evaluations as the passive algorithm. That means that the passive algorithm is worst-case optimal and no sequential algorithm can surpass it. In a typical scenario, however, sequential algorithms perform more efficiently than the passive one, and their performance is assessed with respect to the best possible algorithm.

The first and the most popular globally ϵ-convergent sequential optimization algorithm in the univariate case is the *Pijavskij-Shubert* algorithm [70, 84]. It is a sequential algorithm, performing every new function evaluation at a point corresponding to the global minimum of the current saw-tooth cover $F_n(x)$:

$$x_{n+1} = \arg\min_{x \in A} F_n(x). \tag{2.24}$$

The algorithm terminates when the record value y_{on} does not exceed the global minimum value of the saw-tooth cover by more than ϵ. The algorithm was shown to be one-step worst-case optimal in [44, 45, 91]. It requires up to four times more objective function evaluations to guarantee ϵ-convergence than the *best possible* algorithm [35]. Moreover, in the worst-case scenario, the *Pijavskij-Shubert* algorithm requires up to two times more function evaluations than the passive algorithm. Interestingly, when the Lipschitz constant approaches infinity, the algorithm degenerates to the grid search, i. e. after $2^k+1, k = 1, 2, \ldots$ function evaluations, the trial points form a uniform grid over the feasible interval [48].

Modifications of the *Pijavskij-Shubert* algorithm addressing efficiency in the worst case were suggested in [77, 92]. In essence, only the points of a certain passive strategy are eligible as function evaluation locations and, as a result, the next function evaluation location, determined by the original algorithm, is replaced by the closest point on a grid of a passive algorithm.

A two-phase approach by *Hansen, Jaumard and Lu* [36] combines the *Pijavskij-Shubert* algorithm for discarding large portions of the search space with an approximation to the *best possible* algorithm for exploring the remaining region of indeterminacy. The efficiency of the approach was shown to be close to that of the *best possible* algorithm when a high precision of the solution was required, i. e. for small ϵ [34].

2.3.2 The Multivariate Case

In the multivariate case, several approaches to deal with the Lipschitz optimization were used.

First, a multivariate problem can be reduced to a univariate one. A nested approach, proposed in [70], restates the original multivariate problem as a sequence of nested univariate optimization problems:

$$\min_{\mathbf{x}\in A} = \min_{x_1\in X_1}\{\min_{x_2\in X_2(x_1)}\{\dots \min_{x_d\in X_d(x_1,\dots,x_{d-1})} f(\mathbf{x})\}\}, \tag{2.25}$$

where $X_k(x_1,\dots,x_{k-1}), k = 1,\dots,d$, is an interval $[a_k, b_k]$, and the minimization is performed when the values $x_k, k = 1,\dots,d-1$, are fixed.

Moreover, Peano curves for conversion to the univariate case were proposed in [10, 90]. A notable disadvantage of this approach is that the distances on the resulting curve might be considerably greater than those in the original domain, resulting in several local minima where only one is actually present.

In the second category of multivariate methods, the univariate *Pijavskij-Shubert* algorithm is generalized to the multivariate case by constructing a single lower-bounding function over the feasible region as an analogue to the univariate saw-tooth cover [8, 46, 63, 65, 70, 104]. The objective function is then evaluated at the peaks of the lower bound, which are, however, expensive to locate. The approaches in this category rely on solving systems of linear and quadratic equations to identify the set of all local minima and, consequently, the global minimum of the lower bound and suggest various simplifications of these systems.

The third category consists of the branch-and-bound techniques, corresponding to the general framework, described in [38, 39], with specific algorithms obtained by defining different ways of selecting subproblems, branching and computing the bounds [28, 32, 64, 72]. The algorithms in this category generally require more objective function evaluations to ensure ϵ-convergence than the algorithms using a single lower-bounding function; however, the latter spend a very long time in search of the global minimum of the lower bound as the number of performed function evaluations increases [34]. The case of the simplicial subsets in the branch-and-bound framework is thoroughly discussed in [68], including the effect of various norms in (2.20) as well as different sources of the Lipschitz constant.

Among the various partitioning schemes in the branch-and-bound literature, the trisection-based ones seem to be the most efficient. In particular, when the subsets over which the subproblems are defined are hyper-rectangular, each of them might be represented by two objective function values at the endpoints of the main diagonal, as originally suggested in [71]. A special approach of adaptive diagonal partitions [79], characterized by dividing each hyper-rectangle into three equal parts, ensures that the redundancies in the description of the objective function over the subsets are eliminated and the already performed trials reused. Moreover, the generated trial points can be viewed as lying on an alternative type of a space filling curve, and the multivariate optimization problem can thus be interpreted as a set of univariate optimization problems over the diagonals of the hyper-rectangular subsets. The efficiency of a different trisection-based partitioning rule has been pointed out by other authors as well [32].

2.3.3 The *DIRECT* Algorithm and Its Extensions

The *DIRECT* algorithm [48] was introduced as a multivariate extension to the univariate *Pijavskij-Shubert* algorithm [70, 84]. However, instead of relying on an overestimate of the Lipschitz constant, the algorithm considers all possible rate-of-change constants simultaneously, ensuring an efficient balance between the global and local search. The algorithm is a direct search type technique, meaning that it uses only the objective function values in the optimization process and no knowledge of derivatives. Due to good performance and a small number of parameters the algorithm gained great popularity.

The algorithm iteratively refines a hyper-rectangular decomposition of the feasible region $A = [0, 1]^d$ until the budget of function evaluations is exceeded. Each hyper-rectangle in the decomposition is characterized by the objective function value at the center location $\mathbf{c}_i$ and the distance Δ_i from the center to the vertices, corresponding to some point on the plane in Figure 2.3. At each iteration a set of potentially optimal hyper-rectangles are selected for partitioning according to their lower bound of $f(\mathbf{x})$. A hyper-rectangle i is considered potentially optimal if a rate-of-change constant $\tilde{K} > 0$ and some $\epsilon > 0$ exist such that

$$f(\mathbf{c}_i) - \tilde{K}\Delta_i \leqslant f(\mathbf{c}_j) - \tilde{K}\Delta_j, \forall j, \tag{2.26}$$

$$f(\mathbf{c}_i) - \tilde{K}\Delta_i \leqslant y_{on} - \epsilon|y_{on}|. \tag{2.27}$$

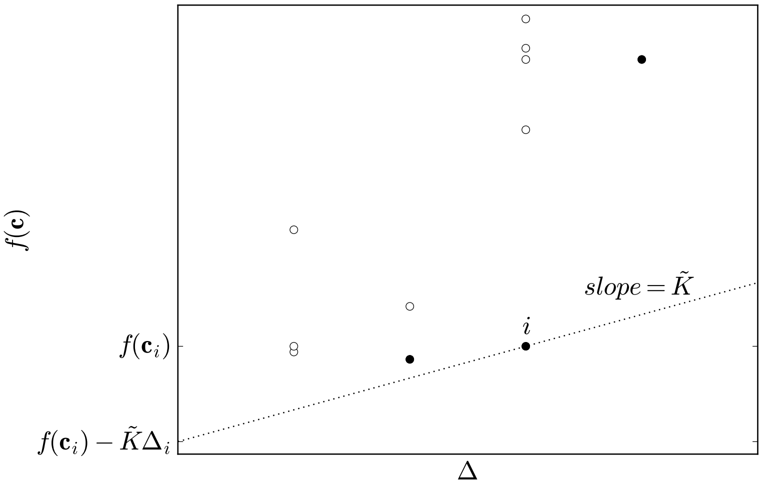

Figure 2.3: Illustration of the selection of hyper-rectangles in *DIRECT*.

The condition (2.26) means that for a potentially optimal hyper-rectangle i, a slope $\tilde{K}$ exists such that the lower bound of $f(\mathbf{x})$ over this hyper-rectangle is the lowest. The condition (2.27) means that a nontrivial improvement in the potentially optimal hyper-rectangle is still possible. Note that the vertical intercept of a line with slope $\tilde{K}$ passing through a point in Figure 2.3 corresponds to the lower bound of the function with a rate-of-change constant $\tilde{K}$ over the respective hyper-rectangle. In the visual representation, the condition (2.26) translates to all the points in the diagram being above the line with slope $\tilde{K}$, passing through point i. Consequently, all potentially optimal hyper-rectangles are located on the lower-right of the convex hull of the points in Figure 2.3 and are shown as black points.

The selected hyper-rectangles are divided into thirds along the longest edges in a certain way, which determines that the diagonal lengths of the produced hyper-rectangles assume values in a discrete set, resulting in vertical groups of points in Figure 2.3.

Hyper-rectangles of different sizes are selected for partitioning at each iteration, as can also be seen in the figure, including at least one of the largest hyper-rectangles. Thus at each iteration some effort is spent both on the global and local search, which is considered a strength of the algorithm. Moreover, the approach based on considering a set of admissible rate-of-change constants $K > 0$ in the definition of the potentially optimal hyper-rectangles eliminates the need for specifying a single Lipschitz constant, which is also a great advantage.

However, several disadvantages of *DIRECT* were pointed out in the literature. First, it locates the regions of local optima quickly, but further convergence to the global minimum is slow. This happens due to an excessive partitioning of the hyper-rectangles in the vicinity of sub-optimal local minimizers. Moreover, no meaningful stopping criterion could be applied to *DIRECT* apart from the limit on the number of function evaluations.

Various modifications were proposed in order to overcome the weaknesses of the algorithm.

An approach automatically balancing explicitly defined local and global phases in the operation of *DIRECT* algorithm was initially proposed in [78]. The already known best solution is refined during the local phase, whereas large hyper-rectangles are partitioned in the global phase in search of better local minima. Moreover, the partitioning strategy of *DIRECT* is replaced with that of the adaptive diagonal partitions. A similar two-phase approach was later adapted in the context of simplicial global optimization in [69].

A globally-biased version *aggressive DIRECT* was proposed in [7]. All hyper-rectangles with the lowest value $f(\mathbf{c})$ within each group are subdivided irrespective of the potential optimality condition. A locally-biased version *DIRECT-l* was introduced in [27], targeting low-dimensional problems with a few local minimizers. The essential features are the grouping of hyper-rectangles by their longest edges, dividing at most one hyper-rectangle per group and allowing even the trivial improvements. Thus the number of groups and divisions within each group is reduced, predominantly in the unexplored regions.

In [57, 58], bi-level and multi-level versions of *DIRECT* were proposed where *DIRECT* is applied on different scales. On the coarsest scale, the whole set of produced hyper-rectangles is considered for further refinement, while on finer scales only a percentage of the smallest hyper-rectangles is considered. Thus the algorithms zoom in on the already explored regions, rendering the strategy suitable for approximating solutions with high accuracy.

The high precision of the solution was raised as the primary concern in [60, 61], where it was noticed that *DIRECT* exhausts the available memory by storing the produced hyper-rectangles before the required precision of the global minimum value is attained. To obtain the required precision, the usual iterations of *DIRECT* are enhanced with local searches started at potentially optimal hyper-rectangles, furthermore, the algorithm is re-executed a number of times after applying certain search space transformations. The resulting algorithms require a great number of function evaluations.

2.4 Chapter Summary and Conclusions

1. The described global optimization algorithms relying on a statistical model of the objective function are defined either as an approximation to the optimal Bayesian algorithm or by means of the rational choice theory. The original form of the algorithms involves expensive auxiliary optimization problems, that need to be simplified in order to expand the applicability of the algorithms. It is, therefore, promising to further explore the decomposition-adjusted statistical models and related algorithms, because computations can be simplified and the theoretical investigation can be conducted.

2. The discussed statistical methods spend considerable resources on careful planning of the trial points, which renders them applicable to engineering problems with expensive objective functions. The high cost of the objective function evaluations motivates to look for ways to accelerate the approximation of the global minimum in terms of the number of trials.

3. The choice of the statistical model in the definition of respective optimization algorithms is usually based on the previous practice, subjective assessment of its adequacy to the problem and computational complexity considerations. However, the systematic investigation of the impact that specific assumed models have on the optimization results, when the objective function precisely corresponds to the assumed model, as well as when it does not, would be interesting.

4. Additional theoretical justification of specific ad hoc choices in the design of algorithms continues to arise interest.

5. The univariate single-objective Lipschitz optimization has been theoretically well explored. The number of trials performed by the *best possible* algorithm constitutes a quality measure for univariate Lipschitz optimization algorithms. The algorithms approaching the efficiency of the *best possible* algorithm are known. Theoretical analysis of the univariate case with a greater number of objectives could be a focus of interest.

6. Theoretical analysis of the multivariate Lipschitz optimization is considerably poorer. The corresponding algorithms either generalize the *Pijavskij-Shubert* algorithm or interpret the branch-and-bound framework in a certain way. In the former approach, both rectangular and simplicial subsets are used.

7. A popular multivariate algorithm *DIRECT* combines the ideas of the univariate Lipschitz optimization and the branch-and-bound techniques with hyper-rectangular subsets. Ideas of addressing its weaknesses might be relevant for improving the algorithms using decomposition-adjusted statistical objective function models.

8. The worst-case analysis is a common theme in Lipschitz optimization, as opposed to the average-case analysis. Conversely, the reviewed algorithms based on statistical models of objectives are designed to provide the best average-case performance.

Chapter 3

Choice of Statistical Model

This chapter focuses on the statistical models of an objective function used in the definition of a global optimization algorithm. An experimental methodology employed to support the choice of the statistical model of the objective function, based on the results with the two conceptual algorithms - the *P-algorithm* and the *Maximum expected improvement algorithm* - is presented. After an introduction in the first section, the second section formulates the problem of the study. The considered objective function models are introduced in the third section. The fourth section deals with the specific forms of the conceptual algorithms defined for the considered models. The numerical experiments are detailed in the fifth section. The final section presents the conclusions.

3.1 Introduction

Intuitively, we expect that an assumed specific model of the objective function chosen for the definition of a global optimization algorithm does affect the optimization process. This model might be chosen with a number of considerations in mind such as the a priori available information on the objective function and the level of implementation complexity of an associated model. It is therefore interesting to find out which considerations matter, and what the effect might be for various models assumed as well as for various objective functions.

3.2 Statement of the Problem

Let us consider the global optimization problem (2.1). Let $\xi(\mathbf{x}), \mathbf{x} \in A$, be assumed as a statistical model of the objective function in the *P-algorithm* (2.18) and the *Maximum expected improvement algorithm* (2.17). Let the expensive *black-box* objective function $f(\mathbf{x})$ be a realization of some (possibly different) statistical model. Given a small budget of function evaluations, the goal is to quantify the expected outcome of finding/missing the global minimizer.

3.3 Considered Models

In global optimization it is customary to adapt a statistical objective function model that has been well-researched from the probability theory point of view. The family of Gaussian stochastic functions is the best-known example. The first model applied in the global optimization context was the *Wiener* process [53], which belongs to this family. Moreover, a common and computationally convenient subcategory is stationary isotropic Gaussian stochastic functions, characterized by the constant mean μ and standard deviation σ, as well as a correlation function $\rho(\tau), \tau \geqslant 0$, invariant with respect to the class of rigid motions:

$$corr(\mathbf{x}_1, \mathbf{x}_2) = \rho(\|\mathbf{x}_1 - \mathbf{x}_2\|) = \rho(\tau), \tau \geqslant 0, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d. \tag{3.1}$$

This chapter focuses on several specific 1- and 2-dimensional Gaussian stochastic functions used as objective function models. The characteristics of the considered stochastic functions are given in Tables 3.1 and 3.2. The names for the stationary isotropic Gaussian stochastic functions are selected to reflect the correlation function used. Respective realizations are shown in Figures 3.1 and 3.2.

The considered 1-dimensional models in Table 3.1 are well-known. The *Wiener* process was the first model applied to global optimization, while both stationary Gaussian processes are widely used in the geostatistics. The *Wiener* process and the stationary Gaussian process with the *Exponential* correlation function possess

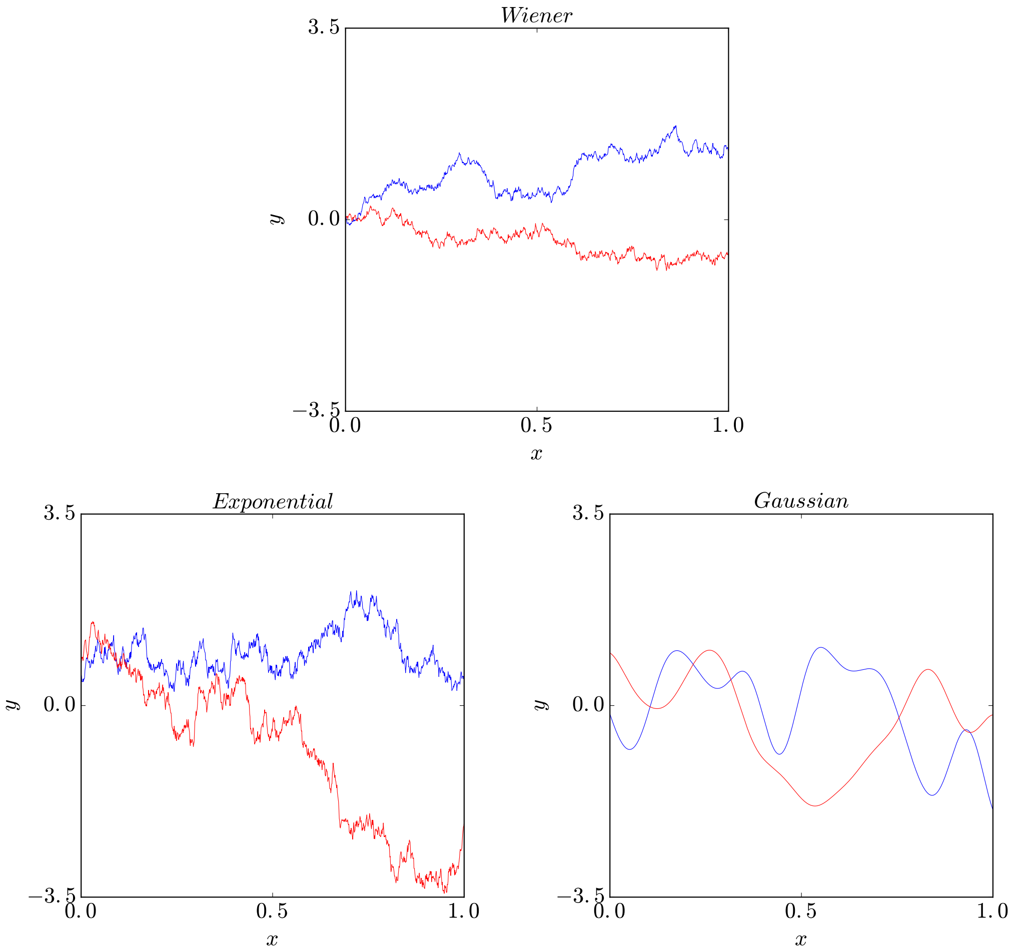

Figure 3.1: Examples of realizations of the 1-dimensional Gaussian random processes, listed in Table 3.1. **Top:** *Wiener* process with $\sigma = 1$. **Bottom:** stationary Gaussian processes with $\mu = 0$, $\sigma = 1$ and the following correlation functions: **bottom left:** $\rho(\tau) = \exp(-\tau/0.2)$ (*Exponential* correlation function), **bottom right:** $\rho(\tau) = \exp(-(\tau/0.2)^2)$ (*Gaussian* correlation function).

the Markov property, enabling to simplify expressions of certain conditional characteristics of the process (see Appendix A for specific expressions).

The concept of *short-range variability* (or, alternatively, *activity* of a function) is best perceived visually, e. g. using Figure 3.1. The sample functions of the *Wiener* process and the stationary Gaussian process with *Exponential* correlation function can be characterized as *rough*, *rugged* or *uneven*. It is obvious that the opposite could be stated with regard to the relatively *unwrinkled* realizations of the stationary Gaussian process with the *Gaussian* correlation function. In order

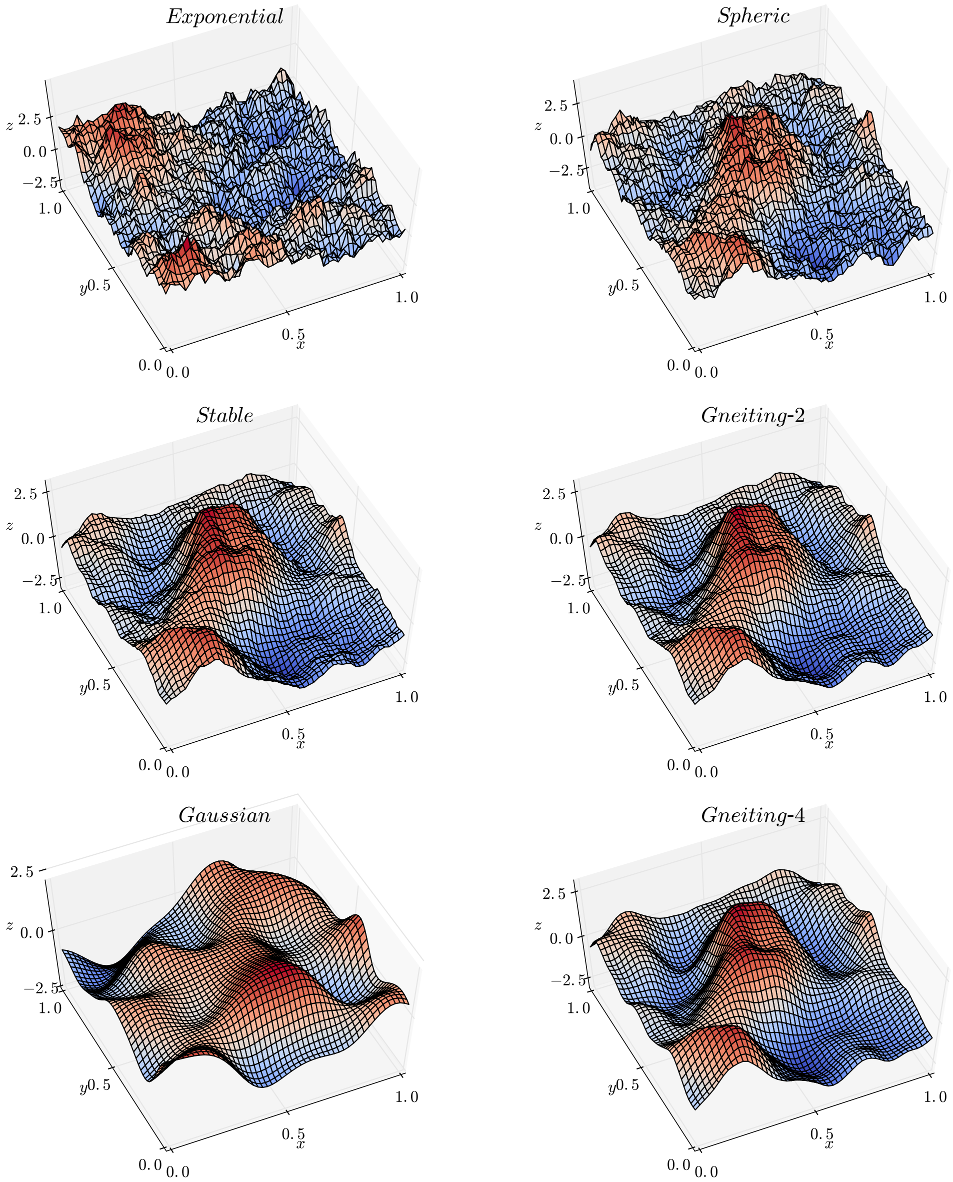

Figure 3.2: Examples of realizations of the 2-dimensional stationary isotropic Gaussian random fields with $\mu = 0$, $\sigma = 1$ and correlation functions $\rho(\tau), \tau \geqslant 0$, given in Table 3.2. The scale parameters of the correlation function are set to the following values: $c = 0.2$ for *Exponential*, $c = 0.2$ for *Stable*, $c = 0.2$ for *Gaussian*, $c = 0.4$ for *Spheric*, $c = 0.5$ for *Gneiting-2*, $c = 0.5$ for *Gneiting-4* fields.

Table 3.1: Characteristics of 1-dimensional random processes.

Process		
Wiener	Description:	Gaussian process with zero mean and covariance function $cov(\xi(x_1), \xi(x_2)) = \sigma^2 \min(x_1, x_2), x_1, x_2 \geqslant 0$; Gaussian increments $\xi(x + \delta) - \xi(x) \sim N(0, \sigma^2 \delta)$ are independent for disjoint intervals; Realizations are not differentiable anywhere with probability 1
	Parameters:	$\sigma > 0$
	Level of short-range variability:	High
	Markov property:	Yes
Exponential	Description:	Gaussian stationary process with exponential correlation model $\rho(\tau) = \exp\left(-\frac{\tau}{c}\right), \tau \geqslant 0$; Not mean-square differentiable
	Parameters:	$\mu, \sigma > 0, c > 0$
	Level of short-range variability:	High
	Markov property:	Yes
Gaussian	Description:	Gaussian stationary process with Gaussian (squared exponential) correlation model $\rho(\tau) = \exp\left(-\left(\frac{\tau}{c}\right)^2\right), \tau \geqslant 0$; Mean-square derivatives of all orders exist
	Parameters:	$\mu, \sigma > 0, c > 0$
	Level of short-range variability:	Very low
	Markov property:	No

to avoid confusion stemming from an improper use of *smoothness* and *differentiability* concepts with respect to the realizations of the stochastic functions, the term *short-range variability* will be used to refer to the relative *ruggedness* of the realizations of all the considered stochastic functions.

Table 3.2 lists a number of stationary isotropic Gaussian random fields, considered in this study as 2-dimensional objective function models. The models are arranged by decreasing level of short-range variability. Moreover, three of them (*Spheric*, *Gneiting-2* and *Gneiting-4*) have compactly-supported correlation functions, resulting in sparse correlation matrices that could be processed by special algorithms. The expressions of the correlations for these models in Table 3.2 are valid up to the third dimension only.

The number of derivatives at 0 of the correlation functions of models *Gneiting-2* and *Gneiting-4* is equal to 2κ, while the largest dimension where $\rho(\tau)$ qualifies as a correlation function is equal to 2γ (see Table 3.2). Specifically, $2\gamma = 3$ for both models means that $\rho(\tau)$ qualifies as a correlation function up to the third dimension. In addition, the correlation functions of models *Gneiting-2* and *Gneiting-4* are, respectively, 2 and 4 times differentiable at 0 (hence the names of

Table 3.2: Characteristics of the stationary isotropic Gaussian random fields with parameters μ, $\sigma > 0$, $c > 0$.

Field	**Correlation model** $\rho(\tau), \tau \geqslant 0$	**Compact support**	**Level of short-range variability**	**Dimension restriction**
Exponential	$\exp\left(-\frac{\tau}{c}\right)$	No	High	-
Spheric	$1 - 1.5\frac{\tau}{c} + 0.5\left(\frac{\tau}{c}\right)^3 \mathbb{1}_{[0,1]}\left(\frac{\tau}{c}\right)$	Yes	High	$\leqslant 3$
Stable	$\exp\left(-\left(\frac{\tau}{c}\right)^{\alpha}\right), \alpha = 1.8$	No	Low	-
Gneiting-2	$(1 + \beta\frac{\tau}{c})(1 - \frac{\tau}{c})^{\beta}\mathbb{1}_{[0,1]}(\frac{\tau}{c}),$ $\beta = \gamma + 2\kappa + 0.5, \gamma = 1.5, \kappa = 1$	Yes	Low	$\leqslant 3$
Gneiting-4	$\left(1 + \beta\frac{\tau}{c} + \frac{\beta^2-1}{3}(\frac{\tau}{c})^2\right)(1 - \frac{\tau}{c})^{\beta}\mathbb{1}_{[0,1]}(\frac{\tau}{c}),$ $\beta = \gamma + 2\kappa + 0.5, \gamma = 1.5, \kappa = 2$	Yes	Low	$\leqslant 3$
Gaussian	$\exp\left(-\left(\frac{\tau}{c}\right)^2\right)$	No	Very low	-

the models), and thus the sample surfaces of *Gneiting-4* exhibit relatively smaller short-range variability (see Figure 3.2).

3.4 Special Forms of the Algorithms

This section presents the specialized versions of the two conceptual algorithms - the *P-algorithm* (2.18) and the *Maximum expected improvement algorithm* (2.17), obtained for the case when the objective function model $\xi(\mathbf{x}), \mathbf{x} \in A$, is a Gaussian stochastic function, as the study presented in this chapter considers only this particular type of models (see Section 3.3). The experiments described in this chapter are based on the performance of these algorithms, aiming to evaluate the suitability of different assumed objective function models, used in the definition of the algorithms.

In the considered case, the *P-algorithm* can be expressed as

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x} \in A} \mathbb{P}(\xi(\mathbf{x}) \leqslant y_{on} - \epsilon_n \,|\, \xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n) = \Phi\left(\omega_n(\mathbf{x}, \epsilon_n)\right), \quad (3.2)$$

$$\Phi(z) = \frac{1}{2\pi}\int_{-\infty}^{z} \exp(t)dt, \quad (3.3)$$

$$\omega_n(\mathbf{x}, \epsilon) = \frac{y_{on} - \epsilon - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}, \quad (3.4)$$

where $\Phi(\cdot)$ is the Gaussian cumulative distribution function, while $m(\mathbf{x}|\cdot)$ and $s^2(\mathbf{x}|\cdot)$ denote the conditional mean and conditional variance of $\xi(\mathbf{x})$, respectively. Since $\Phi(\cdot)$ is a monotonically increasing function, an equivalent algorithm results by maximizing the argument $\omega_n(\mathbf{x}, \epsilon)$ directly:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in A} \omega_n(\mathbf{x}, \epsilon) = \arg\max_{\mathbf{x}\in A} \frac{y_{on} - \epsilon - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}. \tag{3.5}$$

The *Maximum expected improvement algorithm* for a Gaussian stochastic function $\xi(\mathbf{x}), \mathbf{x} \in A$, is expressed as

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in A} \left(v_n(\mathbf{x})\Phi(v_n(\mathbf{x})) + \frac{1}{2\pi} \exp\left(-\frac{v_n^2(\mathbf{x})}{2} \right) \right) \tag{3.6}$$

$$v_n(\mathbf{x}) = \frac{y_{on} - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}. \tag{3.7}$$

The expressions to compute the conditional characteristics $m(\mathbf{x}|\cdot)$ and $s^2(\mathbf{x}|\cdot)$ for stationary isotropic Gaussian stochastic functions are (2.5) and (2.6), respectively; some special cases, considered in this study, are given in Appendix A.

3.5 Numerical Experiments

This section introduces an experimental methodology to be used for evaluating the performance of the two considered algorithms, the *P-algorithm* and the *Maximum expected improvement algorithm*, with respect to the various combinations of the assumed and actual objective function models. More precisely, in the definition of an algorithm one of the objective function models, discussed in Section 3.3, is used and called the *assumed model*. The resulting version of the algorithm is applied to the objective function that is a realization of either the same or possibly different model, called the *actual model*.

3.5.1 The Objective Functions

In this study, the realizations of the models, i. e. Gaussian stochastic functions (described in Section 3.3), were used as objective functions. The R package RandomFields [76] was used to generate all realizations of the stationary isotropic Gaussian stochastic functions (procedure *RFsimulate*) and to obtain the maximum likelihood estimates (MLEs) of their parameters (procedure *RFfit*). The realizations of the *Wiener* process were generated using the *Wiener bridge* methodology, starting with values at the interval endpoints and generating a conditional random value at the midpoint of the interval.

The 1-dimensional case. In the 1-dimensional case, a total of 1000 realizations of each of the random processes considered in Table 3.1 were generated. Each realization was represented by its values at the points of the interval $[0,\ 1]$: $t_i = i/N$, $i = 0, \ldots, N$, $N = 1000$. The parameters for the processes were: $\sigma = 1$ for the *Wiener* process, $\mu = 0$ and $\sigma = 1$ for the stationary Gaussian processes. Based on the visual examination of the sample functions of the *Gaussian* model, its scale parameter $c = 0.07$ was set trying to ensure that the generated realizations contain at least one deceptive local minimum highly similar to the global minimum. To match the level of variation in the sample path values, we estimated the parameter c of the *Exponential* model by finding the MLEs of μ, σ and c from the uniformly-spaced values at $t_i = i/M$, $i = 0, \ldots, M$, $M = 10$, of each of the *Gaussian* sample functions. The scale parameter for the *Exponential* model $c = 0.06$ close to the mean of the corresponding estimates (see Table B.1 for parameter estimation statistics) was chosen.

The 2-dimensional case. Analogously, in the 2-dimensional case, 1000 realizations of each of the stationary isotropic Gaussian random fields in Table 3.2 were generated to be used as objective functions in the experiments. Each realization was represented by values on a grid of points $\mathbf{t}_{ij} = (i/N, j/N)$, $i, j = 0, \ldots, N$, $N = 100$. For all stochastic functions, $\mu = 0$ and $\sigma = 0$ were used. The scale parameters c were set according to the following argument. To generate the sample functions for experimentation, the *Gaussian* model scale parameter $c = 0.2$ was first selected by visual examination of the realizations. Then, according to this model, 1000 realizations were generated. The MLEs of scale parameters of the remaining models were determined from the values of these realizations at the points of a sparser grid: $\mathbf{k}_{ij} = (i/M, j/M)$, $i, j = 0, \ldots, M$, $M = 4$. The scale parameter c values for each of the models were set equal to the means of their estimates over 1000 *Gaussian* model realizations (see Table B.2 for parameter estimation statistics). The following values were used: $c = 0.2$ for *Exponential* model, $c = 0.4$ for *Spheric* model, $c = 0.2$ for *Stable* model, $c = 0.5$ for *Gneiting-2* model and $c = 0.5$ for *Gneiting-4* model. The statistics of MLEs of all combinations of the assumed and actual models are given in Table B.2.

3.5.2 The Implementation of the Algorithms

This section discusses the details of implementation of the *P-algorithm* and the *Maximum expected improvement algorithm*. First, since each of the algorithms assumes a certain parametric statistical model of the objective function, the details of the assumed model parameter estimation are presented. Second, the specifics of solving an auxiliary optimization problem in order to select a new function evaluation location at each step of the algorithms are given. The rationale behind the selection of the *P-algorithm* parameter ϵ_n is also discussed.

Parameter estimation. Each of the algorithms assumes a certain statistical model (*assumed model*) of the objective function that is generated according to a possibly another model (*actual model*). The assumed model has a set Θ of parameters that have to be estimated and set in order to complete the definition of the algorithm. Let us recall that the models considered in this study are Gaussian stochastic functions. One of them is the *Wiener* process, characterized by a single parameter, i. e. $\Theta = \{\sigma\}$. The rest of the models are stationary isotropic Gaussian stochastic functions, parametrized by the mean, standard deviation and the correlation function scale parameter, i. e. $\Theta = \{\mu, \sigma, c\}$.

Suppose an optimization algorithm assumes an objective function model with a set of parameters Θ, while the objective function is generated according to the *actual model*. The values of the objective function (i. e. the realization of the *actual model*) on a grid were used to estimate the parameters Θ of the *assumed* *model* to be used in the optimization of that specific objective function. Thus the parameters had to be estimated 1000 times for each combination of the *assumed* and *actual* models. The grid $k_i = i/M$, $i = 0, \dots, M$, $M = 10$, was used in the 1-dimensional case, while the grid $\mathbf{k}_{ij} = (i/M, j/M), i, j = 0, \dots, M, M = 4$, was used in the 2-dimensional case. For the stationary isotropic Gaussian stochastic functions the method of maximum likelihood estimation was used, while the *Wiener* process covariance function parameter σ was estimated according to the following formula [102]:

$$\bar{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \frac{\xi(x_i^n) - \xi(x_{i-1}^n)}{x_i^n - x_{i-1}^n}, \tag{3.8}$$

where $x_i^n, i = 1, \dots, n$, are the trial points arranged in an increasing order.

The mean and the standard deviation of the parameter estimates obtained are given in Tables B.1 and B.2.

Auxiliary optimization problem. Both considered algorithms optimize an auxiliary criterion for selecting the new location of function evaluation (see Equations (3.2) and (3.6)). The maximization over a grid of points where the values of the stochastic function realizations are known was used. Specifically, at each point of the grid the auxiliary criterion is evaluated, and the point, where the maximum is attained, becomes the next objective function evaluation location.

In addition, the criterion computation involves the inversion of the correlation matrix of the previous trials. It is well-known that this matrix is ill-conditioned when the assumed model is *Gaussian*. To deal with this problem, the Moore-Penrose matrix pseudo-inversion was performed instead of ordinary inversion for this model.

Definition of optimization success. In the 1-dimensional case, the algorithms were said to have found the global minimizer x^* if, for a budget of $N_{max} = 35$ function evalutions, some trial point $x_i, i \in \{1, \dots N_{max}\}$, was generated that satisfied the condition:

$$|x_i - x^*| \leqslant 0.002. \tag{3.9}$$

Table 3.3: ϵ_n strategies for the *P-algorithm* with every 1-dimensional assumed model.

Assumed model	ϵ_n
Wiener	$\epsilon_n = \begin{cases} 0.5, & \text{if } n \leqslant 10, \\ 0.3\sqrt{n-10}, & \text{if } 10 < n \leqslant 20, \\ 0.2\sqrt{n-20}, & \text{if } 20 < n \leqslant 35. \end{cases}$
Exponential	$\epsilon_n = \begin{cases} 0.5, & \text{if } n \leqslant 20, \\ 0.5\sqrt{n-20}, & \text{if } 20 < n \leqslant 35. \end{cases}$
Gaussian	$\epsilon_n = 0.05(1-t) + 0.001t, t = \frac{n}{35}.$

In the 2-dimensional case, the budget of $N_{max} = 50$ function evaluations was used. The global minimizer $\mathbf{x}^* = (x_1^*, x_2^*)$ was considered to be found if, for some generated trial point $x_i = (x_{i1}, x_{i2})$, $i \in \{1, \ldots, N_{max}\}$, it was true that:

$$|x_{ij} - x_j^*| \leqslant 0.002, j = 1, 2. \tag{3.10}$$

The *P-algorithm* ϵ_n parameter. The *P-algorithm* parameter ϵ_n was chosen experimentally for each of the assumed models. The specific formulas used are given in Tables 3.3 and 3.4. These formulas were selected as follows. Taking a specific assumed model, each algorithm was run on the realizations of the same model, i. e. the assumed and actual models were coincident. Various decaying ϵ_n strategies were employed heuristically, and those that brought about the criteria (3.9) and (3.10) to be satisfied the greatest percentage of times were selected.

3.5.3 Experimental Setup

Implementations of the *P-algorithm* and the *Maximum expected improvement algorithm*, resulting from different assumptions on the objective function models considered in this study (Tables 3.1 and 3.2), were run, providing the realizations of all the considered 1- and 2-dimensional models as objective functions. As a

Table 3.4: ϵ_n strategies for the *P-algorithm* with every 2-dimensional assumed model.

Assumed model	$\boldsymbol{\epsilon_n}$
Exponential	$\epsilon_n = \begin{cases} 3, & \text{if } n \leqslant 0.75 \times 50, \\ 3(1-t)+0.1t, t = \frac{n}{50}, & \text{if } 0.75 \times 50 < n \leqslant 50. \end{cases}$
Spheric	$\epsilon_n = \begin{cases} 3, & \text{if } n \leqslant 25, \\ 3(1-t)+0.1t, t = \frac{n}{50}, & \text{if } 25 < n \leqslant 50. \end{cases}$
Stable *Gneiting-2* *Gneiting-4* *Gaussian*	$\epsilon_n = 0.05(1-t) + 0.001t,\, t = \frac{n}{50}.$

result, 1000 runs per combination of an assumed and actual objective function model were performed.

A situation when the criteria (3.9) and (3.10) were not satisfied for a budget of trials ($N_{max} = 35$, if $d = 1$, and $N_{max} = 50$, if $d = 2$) was considered a failure. For each combination of the assumed and actual objective function model, the percentage of failures out of the 1000 optimization problems was recorded. The 1- and 2-dimensional results for the *P-algorithm* are presented in Tables 3.5 and 3.6, while the analogous results for the *Maximum expected improvement algorithm* are presented in Tables 3.7 and 3.8. To facilitate visual comparison, the darker shades of the cells correspond to higher numbers. The same results are presented graphically in Figure 3.3.

3.5.4 Results and Discussion

The results presented in Tables 3.5, 3.6, 3.7 and 3.8 (Figure 3.3) suggest that the analysis should concentrate on the complexity of the objective function. The higher the short-range variability of the model, the higher the complexity of the objective function generated by that model. Among the models considered, the

Table 3.5: The percentage of failures: *P-algorithm*, $d = 1$, $N = 35$ trials.

Assumed model	**Actual model**		
	Wiener	*Exponential*	*Gaussian*
Wiener	5.6	26.0	25.8
Exponential	13.5	22.4	18.6
Gaussian	56.6	67.0	9.1

Table 3.6: The percentage of failures: *Maximum expected improvement algorithm*, $d = 1$, $N = 35$ trials.

Assumed model	**Actual model**		
	Wiener	*Exponential*	*Gaussian*
Wiener	6.9	28.4	24.3
Exponential	8.6	27.9	33.8
Gaussian	55.4	68.2	6.4

Wiener and *Exponential* models in the 1-dimensional case, and *Exponential* and *Spheric* models in the 2-dimensional case, are the most complicated. This can be seen in Figures 3.1 and 3.2, as the realizations of these models have a great number of local minimizers.

3.5.4.1 The 1-dimensional case

As Tables 3.5 and 3.6 (Figure 3.3) show, in the 1-dimensional case the performance of both algorithms using various assumed objective function models is similar and is therefore described here jointly. When algorithms are constructed

Table 3.7: The percentage of failures: *P-algorithm*, $d = 2$, $N = 50$ trials.

Assumed model	**Actual model**					
	Exponential	*Spheric*	*Stable*	*Gneiting-2*	*Gneiting-4*	*Gaussian*
Exponential	32.4	27.1	8.6	8.4	7.8	3.8
Spheric	28.7	23.9	7.4	9.9	12.6	7.3
Stable	43.0	36.8	21.2	25.6	40.5	33.6
Gneiting-2	47.2	43.9	16.3	19.6	32.0	21.1
Gneiting-4	55.8	54.3	32.7	22.5	20.7	10.6
Gaussian	52.7	46.4	27.2	21.2	13.6	6.6

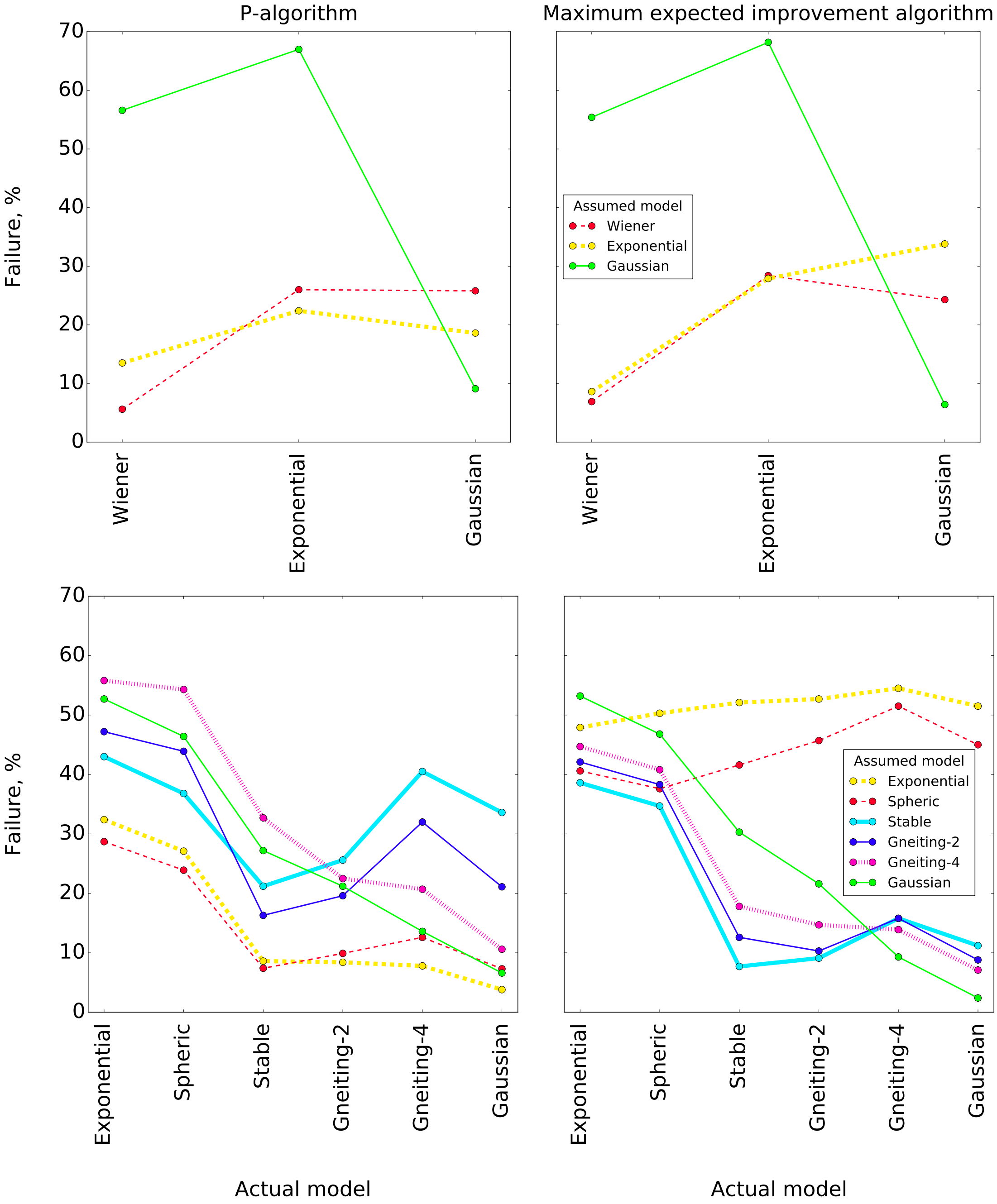

Figure 3.3: The percentage of cases when the global minimum was not found. **Top:** Results with 1-dimensional models, $N = 35$. **Bottom:** Results with 2-dimensional models, $N = 50$.

Table 3.8: The percentage of failures: *Maximum expected improvement algorithm*, $d = 2$, $N = 50$ trials.

Assumed model	**Actual model**					
	Exponential	*Spheric*	*Stable*	*Gneiting-2*	*Gneiting-4*	*Gaussian*
Exponential	47.9	50.3	52.1	52.7	54.5	51.5
Spheric	40.6	37.6	41.6	45.7	51.5	45.0
Stable	38.6	34.7	7.7	9.1	15.8	11.2
Gneiting-2	42.1	38.3	12.6	10.3	15.8	8.8
Gneiting-4	44.7	40.8	17.8	14.7	13.9	7.1
Gaussian	53.2	46.8	30.3	21.6	9.3	2.4

under the assumption of a complicated model, i. e. the *Wiener* or *Exponential* model, the percentages of unsolved cases fall in the range $5 - 34\%$, regardless of the actual model of the objective function. On the other hand, when the algorithm assumes a simple objective function model, *Gaussian* in this case, the percentage of unsolved cases is below 10%, when the assumption is true, and above 50% otherwise, i. e. for the *Wiener* and *Exponential* actual models.

Thus, for a 1-dimensional objective function with unknown characteristics, the *Wiener* and *Exponential* models could be recommended to assume in both algorithms. The *Gaussian* model is unacceptable when the objective function violates the model assumption and thus can not be used when no a priori information on the problem is available.

3.5.4.2 The 2-dimensional case

The results given in Tables 3.7 and 3.8 (Figure 3.3) demonstrate that when the objective functions correspond to the complicated actual models (*Exponential* or *Spheric*), the percentage of unsolved cases is $23 - 56\%$ for the *P-algorithm*, and $34 - 54\%$ for the *Maximum expected improvement algorithm*. This implies that both algorithms find these functions hard to optimize irrespective of the assumed model, even when the assumed model matches the actual model. The best results are obtained using the *P-algorithm* with the *Exponential* or *Spheric* assumed models.

When the algorithms are defined assuming a complicated objective function

Table 3.9: Summary of the guidelines for the choice of the assumed model. Abbreviations *P* and *MEI* stand for, respectively, the *P-algorithm* and the *Maximum expected improvement algorithm*.

Dimension	**Complexity of the objective function**	**Guidelines**
1	Unknown	*P/MEI + Wiener/Exponential*
	High	*P/MEI + Wiener/Exponential*
	Low	*P/MEI + Gauss*
2	Unknown	*P + Exponential/Spheric* *MEI + Stable*
	High	*P+ Exponential/Spheric*
	Low	*P/MEI + Gauss* *MEI + Gneiting-2/Gneiting-4*

model (*Exponential* or *Spheric* assumed models), the results differ. The percentage of unsolved cases is $3 - 33\%$ for the *P-algorithm* and $37 - 55\%$ for the *Maximum expected improvement algorithm* accross various actual models. Moreover, when the assumption is false, i. e. the objective function is actually less complicated, the percentage for the *P-algorithm* is $3 - 13\%$, while for the *Maximum expected improvement algorithm* it is $41 - 55\%$. This means that the *P-algorithm* performs considerably better assuming a complicated objective function, irrespective of the actual model, while assuming such a model for the *Maximum expected improvement algorithm* is not advisable in any case.

On the other hand, when in the definition of the algorithm one of the simple models (*Gneiting-2*, *Gneiting-4* or *Gaussian*) is assumed, and one of these models is the actual model, the following percentages of unsolved cases result: $6 - 32\%$ for the *P-algorithm*, and $2 - 16\%$ for the *Maximum expected improvement algorithm*, so the latter performs better. It can thus be concluded that it is beneficial for both algorithms to assume a simple model when the objective function is expected to be simple as well. However, when no information on the objective function is available, these models should not be assumed.

Finally, the results for the *Stable* model seem rather promising, as the percentage of unsolved cases for the *Maximum expected improvement algorithm* is $9 - 39\%$, across various actual models. This suggests that the model is acceptable to be used in the *Maximum expected improvement algorithm*, when no information on the objective function is available.

Guidelines for the usage of the assumed models corresponding to the a priori information about the objective function complexity are summarized in Table 3.9.

3.6 Chapter Summary and Conclusions

1. An experimental methodology is proposed with the view of supporting the selection of an appropriate statistical model to be used for constructing a global optimization algorithm.

2. When there is no a priori information about the objective function, it is recommended to assume a complicated objective function model (*Exponential* or *Wiener*) in the 1-dimensional versions of the *P-algorithm* and the *Maximum expected improvement algorithm*. The same recommendation holds in the 2-dimensional case for the *P-algorithm* (*Exponential* or *Spheric* models). Alternatively, the *Maximum expected improvement algorithm* can also be used with the assumption of the *Stable* model of average complexity.

3. When the objective function is expected to have low short-range variability, the assumption of the *Gaussian* model for both algorithms should be used with similar results to be expected. Alternatively, the models *Gneiting-2* and *Gneiting-4* could be assumed for the *Maximum expected improvement algorithm* in the 2-dimensional case.

4. An objective function with high short-range variability is going to be hard to optimize irrespective of the assumed model. In the experiments, the best results for the hard functions were obtained using the 1-dimensional versions of both algorithms with *Wiener* or *Exponential* assumed models, and the 2-dimensional version of the *P-algorithm* with the *Exponential* or *Spheric* assumed models.

5. On the whole the *P-algorithm* constructed assuming the *Exponential* model performs the best for a variety of univariate and bivariate objective func-

tions. Simple objective functions can successfully be optimized by the *Maximum expected improvement algorithm*, constructed using one of the simple objective function models.

Chapter 4

Extensions for Hyper-Rectangular Decomposition-Based Statistical Global Optimization

A global optimization algorithm, rooted in the statistical theory of global optimization, is described in the first section of this chapter. The algorithm relies on a hyper-rectangular decomposition of the feasible region that is iteratively refined at each step by performing objective function evaluations at the vertices of the newly formed hyper-rectangles. The algorithm has an established convergence rate, however, its practically noticeable disadvantage is that it spends too many objective function evaluations exploring the vicinity of the discovered promising local minima before finding the true global minimum. Thus, to speed up the approximation of the global minimum in terms of the number of objective function evaluations and obtain an algorithm comparable to other popular contemporary algorithms, two heuristic extensions are introduced in the second and third sections of this chapter. Each of the sections corresponding to the presented algorithms includes a pseudo-code of the algorithm together with a description and an illustrative example. The fourth section is devoted to numerical experiments demonstrating the performance of the presented algorithms. The last section states the conclusions.

4.1 A Multivariate Statistical Global Optimization Algorithm

4.1.1 Introduction

The problem of approximating the global minimum

$$f^* = \min_{\mathbf{x} \in A} f(\mathbf{x}), A = [0, 1]^d, d \geqslant 2, \tag{4.1}$$

of an objective function which is assumed to have partial derivatives up to order 2 and a unique global minimizer $\mathbf{x}^*$ in the interior of A is considered. In this section a multivariate derivative-free global optimization algorithm by Calvin [11] is described. The algorithm stems from the line of research investigating the convergence properties of global optimization algorithms based on statistical models [12, 13, 16, 18], and therefore has a theoretically established asymptotic bound on the convergence rate. However, this thesis focuses on its practical performance rather than theoretical properties, therefore no in-depth presentation of the convergence properties is included.

The algorithm maintains a decomposition of the feasible region, composed of hyper-rectangular subregions with known objective function values at the vertices. A statistical criterion for selecting a hyper-rectangle to be divided is used based on the previously computed objective function values. One hyper-rectangle is selected and divided at each iteration. The algorithm is referred to as *Rect* algorithm in the discussion that follows, the names of its different implementations corresponding to variations of computing the statistical criterion, are introduced in Section 4.1.2.3.

Algorithm *Rect* is similar to the multivariate algorithm [16] based on a simplicial partition of the feasible region, generated by the Delaunay triangulation of the function evaluation locations. As opposed to the latter, the *Rect* algorithm uses a hyper-rectangular decomposition of the feasible region. Moreover, the convergence rate for an arbitrary dimension was established in [11], whereas only a bivariate case was handled in [16].

The result on the convergence rate for the *Rect* algorithm is formulated in

Theorem 1 of [11]. Abstracting away rather complex expressions, it states that there is a number $n_0(f)$ such that for $n > n_0(f)$ the residual error after n objective function evaluations satisfies

$$\Delta_n = y_{on} - f^* \leqslant c_1(f,d)\exp(-c_2(f,d)\sqrt{n}), \tag{4.2}$$

where $c_2(f,d)$ decreases exponentially with dimension d. Although the convergence rate (4.2) holds for a specific class of functions, as defined by the assumptions on the problem, the algorithm itself converges to the global minimum for a broader class of functions, e. g. those that are continuous on $[0,1]^d$.

A disadvantage of the *Rect* algorithm is that the number of objective function evaluations can become rather large, well before the established convergence rate manifests itself.

4.1.2 Description

This section describes the operation of the *Rect* algorithm. Its pseudo-code is formally presented in Algorithm 1.

In short, the algorithm creates a rectangular decomposition of the feasible region and then updates it at each iteration. The update includes selection of a hyper-rectangle according to a certain criterion, defined below, and its partitioning into two equal parts that replace the original hyper-rectangle in the current decomposition.

4.1.2.1 Notation

First of all, some relevant notation used in the definition of the algorithm is introduced.

1. $A = [0,1]^d$, the feasible region that is assumed to be a unit hyper-cube;
2. D, the current hyper-rectangular decomposition of the feasible region;
3. $R = \prod_{i=0}^{d-1}[a_i, b_i]$, a hyper-rectangle;

Algorithm 1 The pseudo-code of the *Rect* algorithm.

```
1: Set the following variables:
     1. $A \leftarrow [0,1]^d$, the feasible region;
     2. $D \leftarrow \{A\}$, the set, holding the hyper-rectangles of the currect decomposition
        of the feasible region;
     3. $v \leftarrow 1$, the minimum hyper-rectangle volume.
2: Evaluate the objective function at the vertices of $A$ and set the number of objective
   function evaluations $n \leftarrow 2^d$.
3: while The stopping condition is not satisfied do
4:     Determine the best hyper-rectangle: $R^* \leftarrow \max_{R \in D} \eta(R)$.
5:     Form two hyper-rectangles $R_j^*, j = 1, 2$, from the best hyper-rectangle $R^*$ as
   follows. Suppose that $R^* = \prod_{i=0}^{d-1}[a_i, b_i]$ and $\gamma$ is the smallest index such that
   $b_\gamma - a_\gamma \geqslant b_i - a_i, \forall i = 0, \dots, d-1$. Then
       $$R_1^* \leftarrow [a_0, b_0] \times \cdots \times [a_\gamma, (a_\gamma + b_\gamma)/2] \cdots \times [a_{d-1}, b_{d-1}],$$
       $$R_2^* \leftarrow [a_0, b_0] \times \cdots \times [(a_\gamma + b_\gamma)/2, b_\gamma] \cdots \times [a_{d-1}, b_{d-1}].$$
6:     Evaluate $f$ at (at most) $2^{d-1}$ new locations $\mathbf{s}^j, j = 0, \dots, 2^{d-1} - 1$:
       $$\mathbf{s}^j = (s_0^j, s_1^j, \dots, (a_\gamma + b_\gamma)/2, \dots, s_{d-1}^j), \text{ where}$$
       $$s_i^j \leftarrow \begin{cases} a_i, & \text{if } (j \text{ div } 2^{d-1-i} \bmod 2) = 0; \\ b_i, & \text{otherwise.} \end{cases}$$
7:     Increment $n$ by the number of new function evaluations in Step 6.
8:     Update $y_{on}$ if a lower function value was found in Step 6.
9:     $D \leftarrow D \backslash R^* \cup \{R_1^*, R_2^*\}$.
10:    if $V(R^*) = v$ then $v \leftarrow v/2$.
11:    end if
12: end while
```

4. $\mathbf{p}^i = (p_0^i, p_1^i, \dots, p_{d-1}^i)$, $i = 0, \dots, 2^d - 1$, a vertex of a hyper-rectangle. Vertices are ordered such that $\mathbf{p}^i < \mathbf{p}^j \implies \exists l : 0 \leqslant l \leqslant d-1, p_k^i \leqslant p_k^j, \forall k = 0, \dots, l$ and $p_l^i < p_l^j$;

5. $f_i = f(\mathbf{p}^i), i = 0, \dots, 2^d - 1$, the corresponding objective function value at the hyper-rectangle vertex;

6. $verts(R)$, the set of hyper-rectangle R vertices, i. e. $verts(R) = \{\mathbf{p}^i : i = 0, \dots, 2^d - 1\}$;

7. $y_{on} = \min_{i=1,\dots,n} y_i$, the minimum objective function value found after n function evaluations;

8. $L : [0,1]^d \to \mathbb{R}$, a function defined in the following way. In the interior of some hyper-rectangle R, $L(\mathbf{x})$ coincides with the multi-linear interpolant of the function values at the vertices of R; on the boundaries of hyper-rectangles $L(\mathbf{x})$ is set equal to y_{on}, i. e. the global minimum of $L(\mathbf{x})$.

   The multi-linear interpolant of the objective function values for an arbitrary dimension d is defined for $\mathbf{x} = (x_0, x_1, \ldots, x_{d-1}), x_k \in (a_k, b_k)$, $a_k = p_k^0, b_k = p_k^{2^d-1}$, $k = 0, \ldots, d-1$, as follows

$$L(\mathbf{x}) = \sum_{i=0}^{2^d-1} f_i \prod_{k=0}^{d-1} \frac{|x_k - p_k^{2^d-1-i}|}{b_k - a_k}. \tag{4.3}$$

9. $g(x)$, a function defined as:

$$g(x) = \begin{cases} (\lambda x \log(1/x))^{2/d}, & 0 < x \leqslant 1/2, \\ qd^2/4, & x = 1 \end{cases}, \text{ where} \tag{4.4}$$

$$q = \frac{3 \cdot 2^{2/3} e^{-1}}{2\log(2)} \approx 1.27, \lambda = \left(\frac{qd^2}{4}\right)^{d/2}. \tag{4.5}$$

10. $V(R)$, the volume of hyper-rectangle R;

11. $v = \min_{R \in D} V(R)$, the minimum hyper-rectangle volume in the current decomposition;

12. $\eta(R)$, a statistically-justified criterion, assessing the potential of hyper-rectangle R to contain the global minimizer:

$$\eta(R) = \int_R \frac{ds}{(L(s) - y_{on} + g(v))^{d/2}}. \tag{4.6}$$

13. $\hat{\eta}(R)$, a computationally simpler version of criterion (4.6), avoiding the integration:

$$\hat{\eta}(R) = \frac{V(R)}{(L(\mathbf{c}) - y_{on} + g(v))^{d/2}},$$
$$\mathbf{c} = (c_0, c_1, \ldots, c_{d-1}), c_k = (a_k + b_k)/2, k = 0, \ldots, d-1. \tag{4.7}$$

14. $R^* \in D$, a hyper-rectangle maximizing the statistical criterion (4.6) or (4.7),

depending on the specific implementation, see Section 4.1.2.3.

4.1.2.2 Description of the Pseudo-Code

Algorithm 1 starts with basic initialization. First, the original domain of the problem is scaled to a unit hyper-cube $A = [0, 1]^d$, and the objective function is evaluated at its vertices. The hyper-rectangular decomposition of the feasible region D is initialized with A itself, and the minimum hyper-rectangle volume v is set to $V(A) = 1$.

Further, the algorithm proceeds iteratively (Steps 3-12). At each iteration, a criterion (which could either be (4.6), or (4.7) depending on the implementation, see Section 4.1.2.3) is evaluated for each of the hyper-rectangles $R \in D$ (Step 3) and the one with the largest value is selected, denoting it R^*. The best hyper-rectangle is divided into two equal parts in Step 5, cutting the edges along the longest of its dimensions γ in half. The objective function is then evaluated at the no more than 2^{d-1}cutting points, which are defined in Step 6, since it is assumed that a history of already performed evaluations is maintained. Accordingly, the number of objective function evaluations n is incremented and the current record y_{on} is updated, if necessary. The new hyper-rectangles replace R^* in D. The minimum volume v is updated, if the smallest hyper-rectangle has just been divided.

4.1.2.3 Implementations

At the core of the *Rect* algorithm is the computation of the hyper-rectangle selection criterion (4.6). It reduces to the numerical integration over a hyper-rectangle. Since this aspect is highly implementation-specific, the implementation versions differing in a strategy used to compute this criterion are described here. In the two-dimensional case, the integral can be computed combining analytical and numerical techniques. Using the notation and ordering of the hyper-rectangle vertices, introduced in Section 4.1.2.1, in the 2-dimensional case the general expression (4.3) reduces to the bi-linear interpolant of the objective function value at an arbitrary point $\mathbf{x} = (x_0, x_1)$, $x_k \in (a_k, b_k), a_k = p_k^0, b_k = p_k^3, k = 0, 1,$ of the form

$$\begin{aligned} L(\mathbf{x}) &= L(x_0, x_1) = A(x_1)x_0 + B(x_1), \\ A(x_1) &= \frac{\frac{x_1-a_1}{b_1-a_1}(f_0 - f_1 - f_2 + f_3) + (f_2 - f_0)}{b_0 - a_0}, \\ B(x_1) &= \frac{\frac{x_1-a_1}{b_1-a_1}(-f_0p_0^3 + f_1p_0^2 + f_2p_0^1 - f_3p_0^0) + (f_0p_0^3 - f_2p_0^1)}{b_0 - a_0}. \end{aligned} \tag{4.8}$$

Then

$$\eta(R) = \int_R \frac{d\mathbf{x}}{L(\mathbf{x}) - y_{on} + g(v)} = \int_{a_1}^{b_1}\int_{a_0}^{b_0} \frac{dx_0}{L(x_0, x_1) - y_{on} + g(v)}dx_1 = \\ \int_{a_1}^{b_1}\int_{a_0}^{b_0} \frac{dx_0}{A(x_1)x_0 + C(x_1)}dx_1 = \int_{a_1}^{b_1} u(x_1)dx_1, \tag{4.9}$$

where

$$\begin{aligned} u(x_1) &= \begin{cases} \frac{1}{A(x_1)}(\ln(A(x_1)b_0 + C(x_1)) - \ln(A(x_1)a_0 + C(x_1))), & \text{if } A(x_1) \neq 0, \\ \frac{1}{C(x_1)}(b_0 - a_0), & \text{if } A(x_1) = 0, \end{cases} \\ C(x_1) &= B(x_1) - y_{on} + g(v). \end{aligned} \tag{4.10}$$

Note that the function $u(x_1)$ is continuous at the point $\hat{x}_1 = \frac{f_0 - f_2}{f_0 - f_1 - f_2 + f_3}, \hat{x}_1 \in [a_1, b_1]$, that gives $A(\hat{x}_1) = 0$.

As a result, in the 2-dimensional case we can compute $\eta(R)$ as a 1-dimensional integral using equations (4.8)-(4.10). The 1-dimensional integral is computed numerically using the GNU Scientific Library [4] routine *gsl_integration_cquad* requiring a relative error smaller than 10^{-7}. We call the resulting implementation *Rect-A*.

For comparison, we also include a version of $\eta(R)$ that is computed as an integral over a hyper-rectangle R using the classical Monte Carlo method (GNU Scientific Library routine *gsl_monte_plain_integrate*) with $M = 10000$ evaluations of the 2-dimensional integrand. This implementation is referred to as *Rect-MC* below. It is also possible to extend this implementation for an arbitrary dimension.

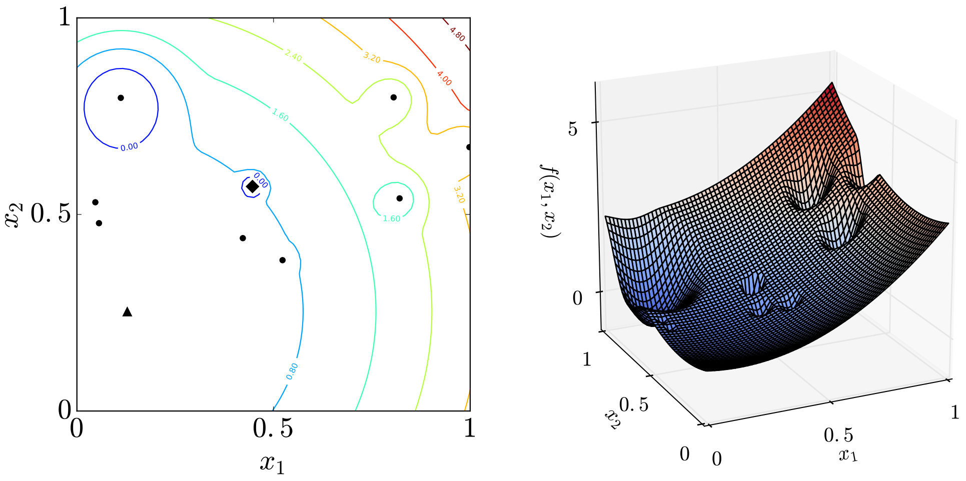

Figure 4.1: The level plot and the surface of a 2-dimensional test function generated by the GKLS generator (function #27 from class #2, see Section 4.4.2 for the description of the testing function classes).

Finally, the simplified version $\hat{\eta}(R)$ of the criterion for dimension d, computed according to Equation (4.7), results in the implementation *Rect-1*. This implementation is the fastest for any dimension.

All of the above presented versions of the algorithm have been implemented in the C++ programming language as part of this thesis. In all implementations the function evaluation database is employed, so that the objective is never evaluated twice at the same location.

4.1.3 Illustrated Example

This section presents an illustrated example of the operation of the *Rect* algorithm.

Figure 4.1 shows the level plot and the surface of a 2-dimensional test function, generated by the GKLS function generator [29], with the domain scaled to the unit square $A = [0, 1]^2$. The generator itself and the testing function classes used in this work are introduced more thoroughly in Section 4.4.2. It now suffices to say that this function is obtained by defining a convex quadratic function, distorted by polynomials to produce local minima.

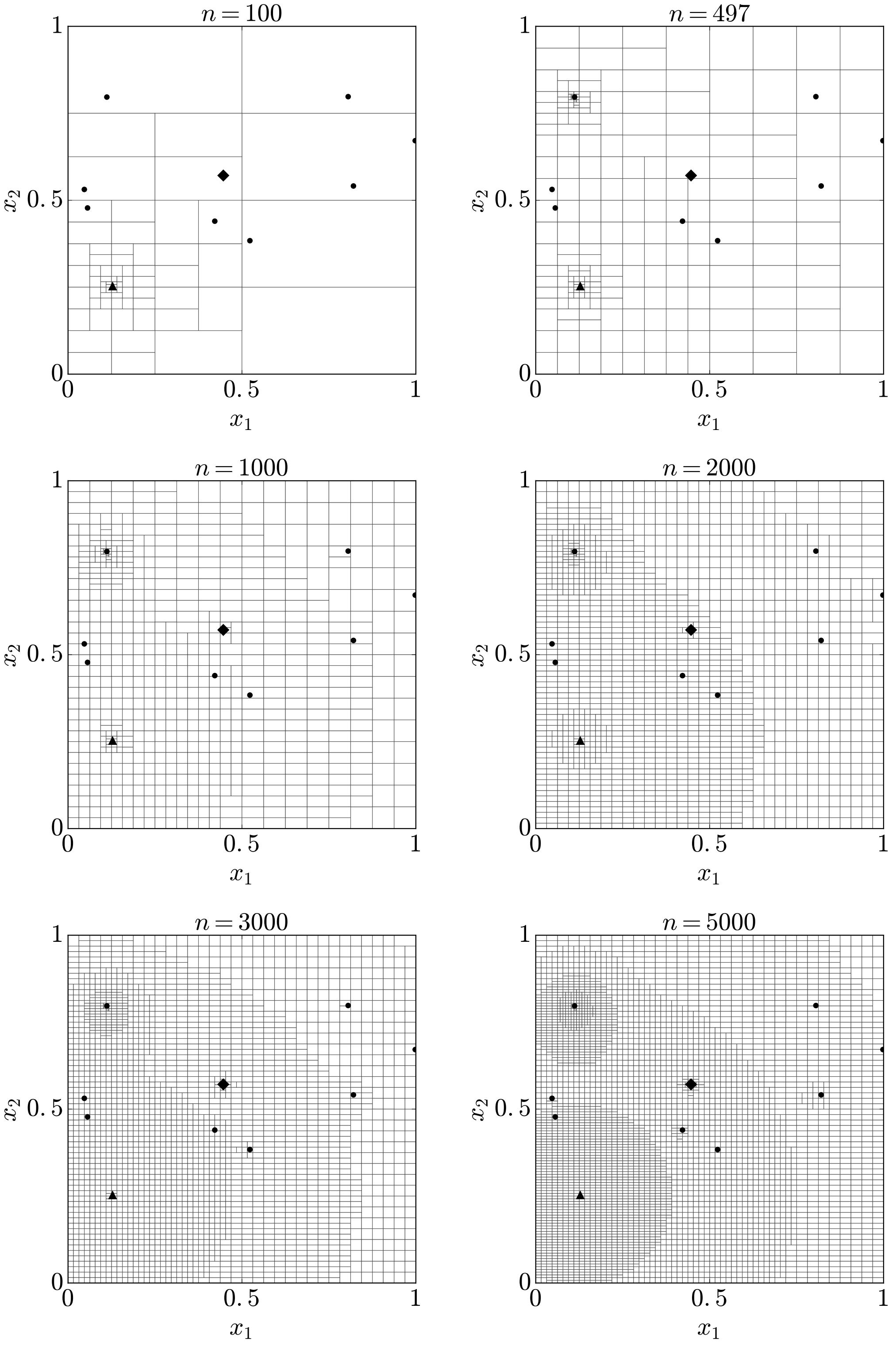

Figure 4.2: An example of the operation of the *Rect* algorithm with a 2-dimensional test function, generated by the GKLS test function generator (function #27 from class #2, see Section 4.2.3 for description and Section 4.4.2 for the introduction to the generator test function classes).

This test function has 10 local minimizers, including the global one, located at $x^* = (0.4464, 0.5711)$ (in the scaled domain) with the value $f(x^*) = -1$. The local minimizers are marked as follows: the black diamond denotes the global minimizer, the black triangle denotes the vertex of the convex quadratic function and black dots mark other local minimizers. The performance of two other algorithms discussed in the persent chapter will be illustrated on the same test function in Sections 4.2.3 and 4.3.3.

The operation of the *Rect* algorithm is illustrated in Figure 4.2. The local minimizers are marked analogously to the Figure 4.1. The function evaluations are the vertices of the displayed rectangles. The rectangular decompositions of the feasible region produced after $n = 100, 497, 1000, 2000, 3000, 5000$ objective function evaluations are shown. The choice $n = 497$ corresponds to the moment when the algorithm termination condition (4.16), used in the experiments in Section 4.4.2, is met. The figure shows that the rectangular decomposition is gradually refined, forming marked agglomerations of function evaluations around the discovered local minimizers.

4.2 Globally-Biased Statistical Global Optimization Algorithm

4.2.1 Introduction

In the ideal case, the *Rect* algorithm is supposed to balance the global and local search efficiently, so that the promising regions could be spotted quickly, ensuring at the same time a rapid convergence towards the global minimum. Such an overly optimistic expectation would mean that the global algorithm is at the same time an efficient local optimizer. However, that is not the case, and for difficult multi-modal problems an excessively local behavior occurs. More precisely, the algorithm concentrates its efforts on refining the hyper-rectangular decomposition around discovered local minimizers, neglecting other areas that might contain deeper local minima. The discovery of the global minimum is thus delayed.

Other decomposition-based methods, predominantly in the Lipschitz optimization context, are also known to suffer from related problems. For example, *DIRECT* [48] is admitted to quickly discover promising regions and obtain a rough approximation to the global minimum, however, further descrease in residual error of the global minimum is slow. One of the possible reasons is, as in the case of the *Rect* algorithm, an excessive refinement of the hyper-rectangles around the suboptimal local minimizers. Similar problems are also present in the methods based on simplicial decomposition of the feasible region, e. g. *DISIMPL-C* and *DISIMPL-V* [68]. Due to structural similarity, these methods could be jointly referred to as *DIRECT*-type methods.

The ramifications of the above-stated problem include a large number of objective function evaluations, which might be expensive, and a potential exhaustion of the available computatational resources such as memory before a sufficiently accurate estimate of the global minimum is obtained.

Alternative solutions to this problem have been proposed in the literature.

A two-phase approach first introduced in [78] in the context of global optimization of expensive Lipschitz continuous functions using hyper-rectangular decompositions of the feasible region was generalized to the case of simplicial decompositions in [69]. The approach targets multivariate multi-extremal *black-box* problems. Balancing the explicitly defined local and global search phases is at the core of the approach. The algorithms execute the local phase refining the current best known function value until no significant improvement has been obtained for some time. At this point the explored neighborhood contains only small subregions, and the large ones are most likely located at a distance. Then a switch to the global phase is triggered to explore large subregions, possibly containing better local minimizers. Thus excessive refinement is prevented, and resources are saved. The phases are essentially defined by filtering the sizes of the subregions eligible for partitioning.

Likewise, bi-level (and multi-level) approaches were introduced in [57, 58], driving inspiration from the multi-grid algorithms for solving linear systems of discretized partial differential equations. The goal is to achieve solutions with a high degree of accuracy at a reasonable computational cost. A decomposition-based global optimization algorithm (e.g. *DIRECT*) is applied to a problem of the same structure on a set of different scales. Namely, on the coarsest scale, the algorithm is allowed to partition the original feasible region for a number of iterations. Then a fixed percentage of the produced hyper-rectangles is selected to be processed by the same algorithm on a finer scale until the finest scale is reached, followed by propagation of the results back to the coarsest scale. The greatest advantage over the original algorithm is achieved when a high precision of approximation is requested.

A set of *DIRECT*-based methods taking advantage of local optimization algorithm runs is developed in [60, 61]. A sufficiently accurate approximation of the global minimum is chosen as a primary goal, to be pursued at the expense of all the available computer memory and without regard to the number of required objective function evaluations. The rationale behind the resulting methods is exploiting the ability of *DIRECT* to detect the promising regions and compensating for the slow convergence towards the global minimum. A number of the suggested methods is derived from two major ideas. First, a local optimization algorithm is applied using the centroids of the potentially optimal hyper-rectangles as starting points at every iteration of *DIRECT*. Second, the problem is repeatedly solved after a piece-wise linear transformation has been applied to the feasible region, centering it on one of the discovered promising points, to benefit from obtaining a possibly different decomposition of the feasible region. The first modification is applied by varying the local optimization procedure, i. e. a non-monotone Newton method used in *DIRMIN*, a derivative-free optimizer in *DFO-DIRMIN* and an economical management of a pool of local optimization executions is utilized in *BDF-DIRMIN*. The second modification encloses one of the modified versions of *DIRECT*, by repeatedly launching it on a transformed problem. This results in algorithms *DIRMIN-TL*, *DFO-DIRMIN-TL* and *BDF-DIRMIN-TL*, respectively. The proposed algorithms are very expensive from the point of view of the number of function evaluations, e. g. the number of local searches might be on the order of several thousand in the "enclosed" version of the algorithm.

Driving the inspiration from [69, 78], an extension to the *Rect* algorithm addressing the excessive local refinement problem is introduced below. The resulting algorithm, referred to as *GB* in the discussion that follows, is an adaptation of a two-phase approach to the context of hyper-rectangular decomposition-based stastistical global optimization.

4.2.2 Description

The present section describes the *GB* algorithm, the steps of which are given in Algorithm 2.

Algorithm 2 The pseudo-code of the *GB* algorithm.

```
Initialization.
$phase \leftarrow STANDARD.$
while the stopping condition is not satisfied do
    Select and divide hyper-rectangle(s).
    if $phase = STANDARD$ then                        ▷ Entering the $STANDARD$ phase.
        if condition (4.12) is satisfied then
            Reset the phase-start record: $s_{best} \leftarrow y_{on}.$
        else if $v < v_{small}$ then
            $phase \leftarrow GLOBAL.$
            Reset the phase-start record: $s_{best} \leftarrow y_{on}.$
            $\bar{D} \leftarrow \{R \in D : V(R) \geqslant v_{large}\}.$
            Initialize the $GLOBAL$ phase iterations counter: $i_{glob} \leftarrow 0.$
        end if
    else                                               ▷ Entering the $GLOBAL$ phase.
        if condition (4.12) is satisfied then
            $phase \leftarrow STANDARD.$
            Reset the phase-start record: $s_{best} \leftarrow y_{on}.$
            $\bar{D} \leftarrow \{R \in D : V(R) \geqslant v_{best}\}.$
        else
            $i_{glob} \leftarrow i_{glob} + 1.$
            if $i_{glob} \bmod i_{period} = 0$ then       ▷ Perform one $SECURITY$ iteration.
                $\bar{D} \leftarrow \{R \in D : V(R) \geqslant v_{best}\}.$
                Select and divide hyper-rectangle(s).
                if condition (4.12) is satisfied then
                    $phase \leftarrow STANDARD.$
                    Reset the phase-start record: $s_{best} \leftarrow y_{on}.$
                    $\bar{D} \leftarrow \{R \in D : V(R) \geqslant v_{best}\}.$
                else
                    $\bar{D} \leftarrow \{R \in D : V(R) \geqslant v_{large}\}.$
                end if
            end if
        end if
    end if
end while
```

4.2.2.1 Notation

Let us start with the following notation that is necessary to describe the operation of the algorithm.

1. $A = [0, 1]^d$, the feasible region;
2. $\bar{D} \subset D$, the subset of the current hyper-rectangular decomposition of the feasible region, holding the active hyper-rectangles, i. e. those eligible for partitioning;
3. s_{best}, the phase-start record objective function value;
4. v, the volume of the smallest hyper-rectangle in $\bar{D}$;
5. v_{small}, the volume threshold parameter, used as a determiner that hyper-rectangles small enough exist for the $GLOBAL$ phase of the algorithm to be triggered;
6. v_{max}, the volume of the largest hyper-rectangle in $\bar{D}$;
7. v_{best}, the volume of the largest hyper-rectangle with a value y_{on} at one of its vertices;
8. v_{large}, the volume threshold, satisfying $v_{best} \leqslant v_{large} \leqslant v_{max}$:

   $$v_{large} = \min\left\{v_{max}, \frac{v_{max}}{2^{\tau}}\right\}, \tau = 0.4\left(\log_2 \frac{v_{max}}{v_{best}} + 1\right). \tag{4.11}$$

9. Sufficient decrease condition:

   $$y_{on} \leqslant s_{best} - 0.01|s_{best}|. \tag{4.12}$$

   The same condition is also used in [69] to determine the moment of switching between the phases.

10. i_{glob}, the $GLOBAL$ phase iterations counter;
11. i_{period}, the algorithm parameter, specifying that every i_{period} $GLOBAL$ phase iterations one $SECURITY$ iteration is performed.

4.2.2.2 Description of the Pseudo-Code

The algorithm starts with a basic initialization at Step 1, consisting of the following actions. First, as in the case of the *Rect* algorithm, the original domain of the problem is scaled to a unit hyper-cube $A = [0, 1]^d$ and the objective function is evaluated at its vertices. A set of active hyper-rectangles $\bar{D}$ is initialized to contain A, i. e. $\bar{D} \leftarrow \{A\}$. Volumes v, v_{max} and v_{best} are all initialized to $V(A) = 1$.

The algorithm is enclosed in a while loop (Step 3) that checks the stopping condition at the beginning of each iteration. Every iteration of the algorithm is performed in either of the two phases, namely $STANDARD$ and $GLOBAL$, starting from the $STANDARD$. The purpose of the $STANDARD$ phase is to explore all but the smallest hyper-rectangles, i. e. those satisfying $V(R) < v_{best}$ are omitted from processing. On the other hand, the search during the $GLOBAL$ phase is biased towards the largest hyper-rectangles, i. e. only those satisfying $v_{large} \leqslant V(R) \leqslant v_{max}$ are considered.

Each iteration of the algorithm starts by selecting and dividing one or more hyper-rectangles (Step 4). At the beginning of each phase the sufficient decrease condition (4.12) is checked. The intuition behind the fact that it is satisfied means that the algorithm found a potential region of attraction of some new local minimizer, and as a result, the vicinity of $x_{on} : f(x_{on}) = y_{on}$ should be explored more thoroughly in the course of the following iteration. This leads to the following hyper-rectangle selection procedure. If condition (4.12) was satisfied in the previous iteration, 2^d hyper-rectangles with lowest center distances to x_{on} are selected. Otherwise, a single hyper-rectangle with the highest simplified score of the *Rect* algorithm $\hat{\eta}$ (4.7) is selected. The same division method as in the *Rect* algorithm (cutting the edges along the longest hyper-rectangle dimension in half, see Step 5 of Algorithm 1), is used to divide the selected hyper-rectangles.

Now let us examine the specifics of each phase in detail. If the sufficient decrease condition (4.12) is satisfied in the $STANDARD$ phase (Step 6), s_{best} is updated, and the next iteration divides 2^d hyper-rectangles. Otherwise, the algorithm operates identically to the *Rect-1* version of the *Rect* algorithm (see Section 4.1.2.3) until very small hyper-rectangles are generated, i. e. the condition $v < v_{small}$ is satisfied (Step 8). This is an indication that enough effort has been spent exploring the vicinity of a certain local minimizer and it is time to globalize the search in order to spot a better one. At this point a switch to the $GLOBAL$ phase occurs, ensuring the necessary preconditions (Steps 9-12), i. e. the phase-start record s_{best} is updated, the active hyper-rectangles set $\bar{D}$ is initialized to contain only large hyper-rectangles and the global iterations counter i_{glob} is reset.

The $GLOBAL$ phase considers a subset of the feasible region, covered by the relatively large hyper-rectangles, as determined at the beginning of the phase. They are gradually partitioned until the sufficient decrease condition (4.12) has been satisfied (Step 15). At this point a switch to the $STANDARD$ phase occurs. The phase-start record s_{best} is updated, and all but the smallest hyper-rectangles become eligible for partitioning again so that the vicinity of a new potentially better local minimizer could be explored appropriately.

On the other hand, in case the condition (4.12) is not satisfied in i_{period} iterations, a better local minimizer might have been missed by not including a hyper-rectangle containing it in $\bar{D}$. It is also possible that the algorithm once again generated many small hyper-rectangles, making the search too local. To guard against such situations, every i_{period} $GLOBAL$ phase iterations a $SECURITY$ iteration is included (Step 21). Namely, a single hyper-rectangle from a set of all but the smallest hyper-rectangles is divided (Steps 22-23). In case the condition (4.12) is satisfied (Step 24), a switch to the $STANDARD$ phase is performed. Otherwise, the smallest hyper-rectangles are filtered away in preparation for the next $GLOBAL$ phase iteration (Step 29).

4.2.3 Illustrated Example

This section presents an illustrated example of the operation of the *GB* algorithm using the test function introduced in Section 4.1.3.

The operation of the *GB* algorithm is illustrated in Figure 4.3. The local minimizers are marked analogously to those in Figure 4.1. The function evaluations are the vertices of the displayed rectangles. The rectangles excluded from further processing are shaded in grey, while the rectangles about to be divided are shaded in red.

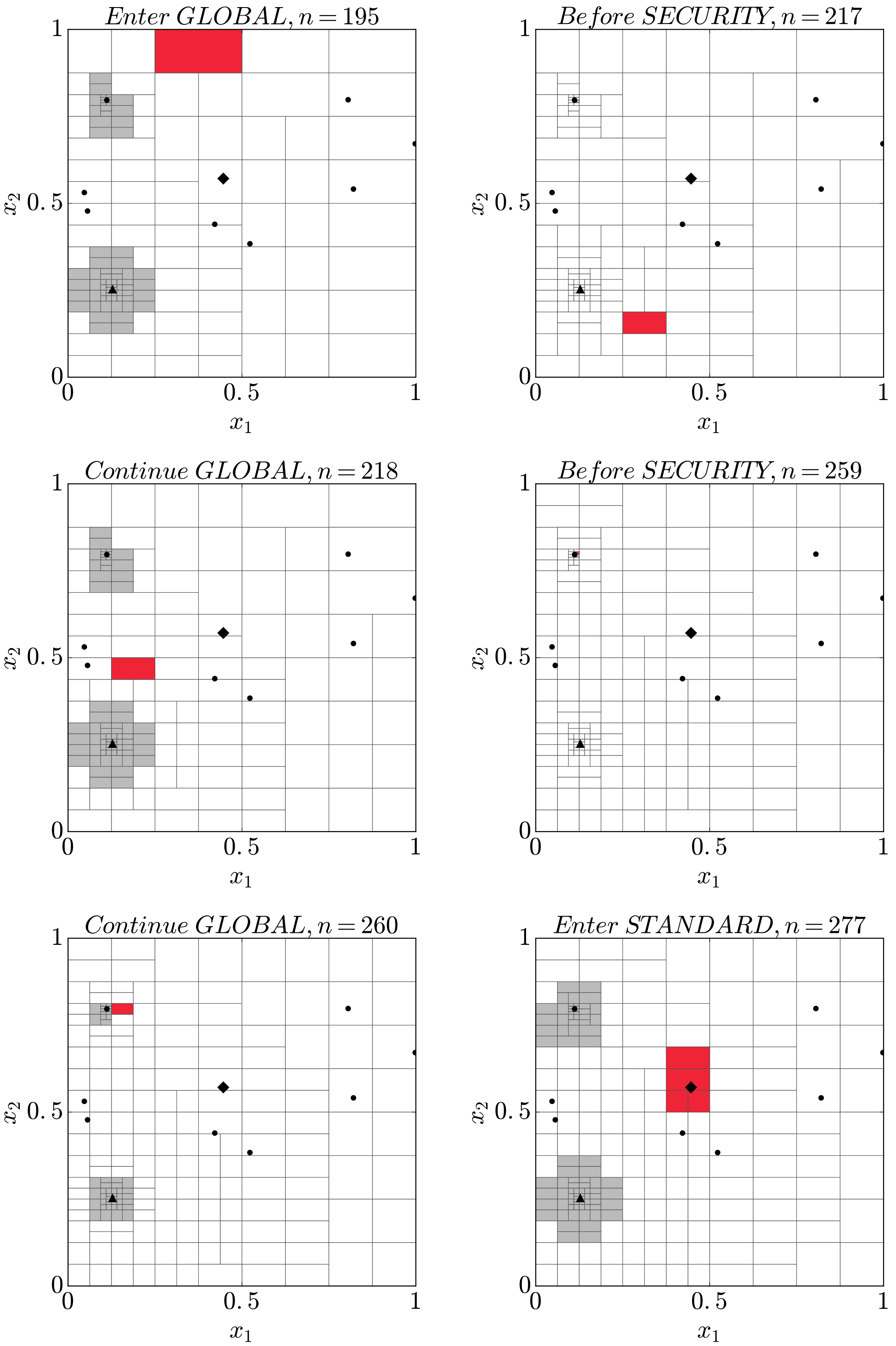

Figure 4.3: An example of the operation of the *GB* algorithm with a 2-dimensional test function, generated by the GKLS test function generator (function #27 from class #2, see Section 4.2.3 for description and Section 4.4.2 for the introduction to the generator test function classes).

The top left subfigure ("Enter $GLOBAL$, $n = 195$") shows the rectangular decomposition of the feasible region when the $GLOBAL$ phase is entered for the first time. A high density of function evaluations in two subregions, corresponding to certain local minimizers, is visible. Since the sufficient decrease was not achieved and the smallest rectangle volume satisfied $v < v_{small}$, no further significant progress was possible in these well-explored regions. These regions are successfully excluded to boost the importance of the larger rectangles.

The top right subfigure ("Before $SECURITY$, $n = 217$") shows the moment of interrupting the $GLOBAL$ phase to perform the $SECURITY$ iteration for the first time. There are no excluded rectangles, as no rectangles are filtered away in Step 22 of the algorithm. Further, since no sufficient decrease happened during the $SECURITY$ iteration, the algorithm returned to the $GLOBAL$ phase, as shown in the middle left subfigure ("Continue $GLOBAL$, $n = 218$"). Analogously, the $SECURITY$ iteration is performed a couple of times more by the algorithm, with subfigures "Before $SECURITY$, $n = 259$" and "Continue $GLOBAL$, $n = 260$" illustrating the last such time before algorithm termination condition (4.16) was met. Finally, an objective function evaluation satisfying the sufficient decrease condition is found by the algorithm operating in the $GLOBAL$ phase and a transition to the $STANDARD$ phase is made (bottom right subfigure, "Enter $STANDARD$, $n = 277$"). As the figure shows, that was the basin of the global minimizer. Algorithm termination according to the condition (4.16) happened after $n = 277$ objective function evaluations.

4.3 Statistical Global Optimization Algorithm with Clustering-Based Local Refinement

4.3.1 Introduction

The same problem of irrationally dense distribution of the function evaluation locations in the vicinity of the currently best point produced by the *Rect* algorithm, as discussed in detail in Section 4.2.1, is addressed in the present section.

However, a different line of reasoning is pursued this time. The points where the function values are computed by the *Rect* global search algorithm tend to cluster in the promising regions, potentially containing the global minimizer. However, it might take a significant number of further function evaluations to explore those regions thoroughly enough in terms of the statistical model. Assuming that a basin of a local minimum is found, the corresponding minimizer can be found with the prescribed termination condition faster by a local search algorithm than continuing the global search. The cluster analysis of the produced hyper-rectangles might reveal the areas worth considering for local refinement using an external local optimizer; moreover, the identified regions might afterwards be masked away, thus directing the global search towards less explored areas.

A hybrid algorithm is presented below, combining the global optimization algorithm *Rect* with the clustering algorithm in order to identify the sub-regions potentially containing the global minimum, and a local optimization algorithm. The resuling hybrid algorithm is referred to as *Cluster* in the discussion that follows.

4.3.2 Description

The present section describes the *Cluster* algorithm, the steps of which are given in Algorithm 3. The algorithm operates by repeatedly switching between two modes of operation: the global search according to the *Rect-1* algorithm (see Section 4.1.2) and the local refinement. The description of the pseudo-code of the algorithm is given below, reflecting the structure of the algorithm. Since the global search part was covered in detail in Section 4.1.2, the focus is mainly on the notation and steps pertaining to the local refinement part of the algorithm.

Remark 4.3.1. The *Cluster* algorithm is coincident with the *Rect-1* algorithm, if the clustering phase is never entered, namely, when the stopping condition is triggered before the clustering phase. □

Algorithm 3 The pseudo-code of the *Cluster* algorithm.

```
Evaluate $f(\mathbf{x})$ at the vertices of $A = [0,1]^d$, $n \leftarrow 2^d$, $D \leftarrow \{A\}$.
while The stopping condition is not satisfied do
    Determine the best hyper-rectangle: $R^* \leftarrow \max_{R \in D} \hat{\eta}(R)$.
    Replace $R^* \in D$ by two hyper-rectangles resulting from dividing $R^*$.
    Increase $n$ by the number of new function evaluations made.
    if $v < v_{small}(d)$ then
        Cluster the hyper-rectangles in $D$, represented by their center locations.
        Run the local optimization algorithm on the cluster containing the hyper-rectangle with the best (lowest) value.
        Eliminate part of the hyper-rectangles from further search.
    end if
end while
```

4.3.2.1 Notation

Some additional notation relevant to the local refinement part of the *Cluster* algorithm is introduced. For the notation relating to the global search part of the algorithm, please, refer to Section 4.1.2.

1. $v_{small}(d)$, a threshold on the minimum hyper-rectangle volume used to trigger the local refinement phase;
2. P, the input point set to the clustering algorithm;
3. $|P|$, the number of points P contains;
4. C, the current set of clusters produced by the clustering algorithm;
5. $c \in C$, a cluster;
6. m, the number of representative points per cluster;
7. $rep(c)$, the set of representative points of cluster c;
8. $\boldsymbol{\mu}_c$, the mean of cluster c;
9. α, the shrinking factor for the representative points;
10. k, the number of clusters to be formed;

11. $\boldsymbol{\omega}_R = \left(\frac{a_0+b_0}{2}, \ldots, \frac{a_{d-1}+b_{d-1}}{2}\right)$, the center of the hyper-rectangle $R \in D, R = \prod_{i=0}^{d-1}[a_i, b_i]$;

12. $verts(c) = \bigcup_{R\in c} verts(R)$, vertices of the hyper-rectangles in cluster c;

13. BB_c, the bounding box of the hyper-rectangles in cluster c;

14. δ_c, the length of the diagonal of BB_c;

15. Δ, a precision parameter;

16. N_{local}^{max}, a budget of function evaluations per local optimization run.

4.3.2.2 Global Search

The algorithm is initialized and operates identically to the *Rect* algorithm implementation *Rect-1*, as described in Section 4.1.2. In short, a hyper-rectangular decomposition of the feasible region is refined at each iteration of the algorithm (Steps 3-5) until a small enough hyper-rectangle is generated, satisfying the condition $v < v_{small}$ (Step 6).

4.3.2.3 Local Refinement

The appropriate moment to enter the local refinement phase is determined automatically. Namely, each time the volume v of the smallest hyper-rectangle in the current decomposition D falls below a threshold $v_{small}(d)$, depending on the dimension, it is assumed that the global search discovered a basin of a certain local minimizer. As a result, a clustering algorithm is now likely to form a cluster in this region.

The threshold $v_{small}(d)$ was set according to the following argument. First, the value $v_{small}(2) = 2^{-15}$ was fixed. Then, assuming that $v_{small}(2)$ is the volume of a 2-dimensional hyper-cube with a side length $\sqrt{v_{small}(2)}$, $v_{small}(d)$ was set to be equal to the volume of a d-dimensional hyper-cube with the same side length. Namely,

$$v_{small}(d) = v_{small}(2)^{d/2}. \tag{4.13}$$

The local refinement phase consists of running a clustering algorithm (Step 7), selecting the best cluster and performing local optimization over it (Step 8) and removing the explored area from further consideration by the global search (Step 9). These steps are discussed in order.

Clustering. The local refinement phase starts by clustering the hyper-rectangles in the current decomposition D. More precisely, for each hyper-rectangle $R \in D$, its center $\boldsymbol{\omega}_R$ point is included in the input points set of the clustering algorithm. Thus a direct correspondence between the clustered points and hyper-rectangles exists.

A hierarchical clustering algorithm *CURE* [33] is employed. It has attractive features, including the possibility to form clusters of arbitrary shape, robustness in the presence of outliers and usage of only a subset of the original point set to produce clusters. The latter feature could be applied to the problems of higher dimensionality, when generally more points are generated by the global search before the clustering starts, and the clustering algorithm takes more time to process them. However, the current implementation produced as part of this thesis has not yet taken advantage of this possibility.

Since it is characteristic of the hierarchical clustering approach to initially treat all input points as separate clusters which are merged starting from those closest to each other, the areas where function evaluations are densely positioned naturally generate large clusters quickly. As a result, it is possible to stop clustering when some large clusters appear, which happens far sooner than all the points in the input set have been merged at least with one other cluster, and interpret them as local minima neighborhoods. This is the reason why hierarchical clustering was chosen over the divisive clustering.

The *CURE* algorithm operates as follows. Initially, the points in the input set P are treated as separate clusters $C = \{\mathbf{x} : \mathbf{x} \in P\}$. Each cluster $c \in C$ is represented by a collection of m (or less for smaller clusters) representative points $rep(c)$, obtained by shrinking the well-scattered points of the cluster towards its mean $\boldsymbol{\mu}_c$ by a factor $\alpha \in [0, 1]$. Specifically, a representative point $\mathbf{q} \in rep(c)$ is obtained as $\mathbf{q} = \mathbf{q}_0 + \alpha(\boldsymbol{\mu}_c - \mathbf{q}_0)$, where $\mathbf{q}_0$ is one of the cluster points positioned relatively far from the rest of the points. This mechanism allows for mitigating the effect of the outliers. The inter-cluster distance is defined in terms of the representative points. Namely, for clusters c and $\hat{c}$ the inter-cluster distance is $dist(c, \hat{c}) = \min_{\mathbf{p} \in rep(c), \mathbf{q} \in rep(\hat{c})} \|\mathbf{p} - \mathbf{q}\|_2$, where $\| \cdot \|_2$ is the Euclidean norm.

Distance to the other closest cluster is used to arrange all clusters in the heap data structure. At each step, the pair with the smallest inter-cluster distance is merged until only k clusters remain. The resulting merged cluster combines the points of its constituent clusters, while the representative points and its closest cluster are redetermined.

The values for the algorithm parameters were chosen as follows. Under the assumption that the neighborhoods of the minimizers are circular, the number of representative points per cluster was $m = 1$, and the cluster was represented by its mean location resulting from $\alpha = 1$. Choosing the number of clusters is complicated, the main goal is to keep the clusters dense, as well as avoid to prematurely stop merging the sub-clusters. The choice $k = |P|/10$, where $| \cdot |$ denotes the number of elements in a set, seemed satisfactory for the experiments performed.

The intuition behind entering the clustering phase is that the global search spotted a new local minimizer and started converging towards it. Moreover, it is assumed that this minimizer is the best point found in the subset of the feasible region, covered by the hyper-rectangles in D, i. e. a subset not excluded in the previous executions of the local refinement phase. It is assumed that the vicinity of such a minimizer contains a great number of small hyper-rectangles, which the clustering algorithm combines into a prominent cluster. Moreover, the majority of the clusters formed are meaningless, as they appear in the regions where the function evaluation locations are not densely positioned and contain small numbers of hyper-rectangles. As a result, the best cluster is selected to be processesd further, while the rest are ignored. More precisely, to determine the best cluster, let a set of vertices of the hyper-rectangles in cluster c be denoted

$verts(c) = \cup_{R \in c} verts(R)$. Then the best cluster is characterized by the lowest value of expression $\min_{\mathbf{x} \in verts(c)} f(\mathbf{x})$.

Local search. The best cluster is further explored using a local derivative-free optimization algorithm by Hooke and Jeeves [37, 50], starting from the cluster vertex with the smallest objective function value, i. e. $\arg\min_{\mathbf{x} \in verts(c)} f(\mathbf{x})$. The algorithm performs the search over a set of decreasing scales h_i, taking probes in a set of directions, parallel to the coordinate axes, trying to find an improvement over the base point. The improvement results in the base point change, otherwise the scale is decreased.

It is important to confine the local search to a box, corresponding to the boundaries of the cluster. This way the search does not deviate from the starting point unpredictably and is likely to stay within the attraction region of the local minimizer that generated the cluster of the global search points in the first place. This helps to ensure that removing the hyper-rectangles spanned by the local search steps does not remove attraction regions of other local minimizers, possibly, the global one. The bound constraints are easily incorporated in the local search procedure by disallowing objective function evaluations outside the bounded region.

The bound constraints for the search inside the cluster c coincide with the bounding box of the hyper-rectangles of the cluster:

$$BB_c = \prod_{i=0}^{d-1} [\min_{\mathbf{p} \in verts(c)} p_i, \max_{\mathbf{p} \in verts(c)} p_i]. \tag{4.14}$$

Let us denote the diagonal of BB_c as δ_c. Then the search scales are set to $h_j = \frac{1}{2^j}$, $j = \lfloor \log_{1/2} 0.2\delta_c \rfloor, \ldots, \lceil \log_{1/2} \Delta \rceil$, where Δ is a parameter corresponding to the required precision. This way the first step size is not greater than 0.5 and the solution is refined up to the required precision. The local search stops when no more improvement over the remaining scales is achieved or the budget of function evaluations $N_{max}^{local} = 1000$ is exhausted. All function evaluations performed at this stage are stored separately from those of the global optimization algorithm, corresponding to the vertices of hyper-rectangles.

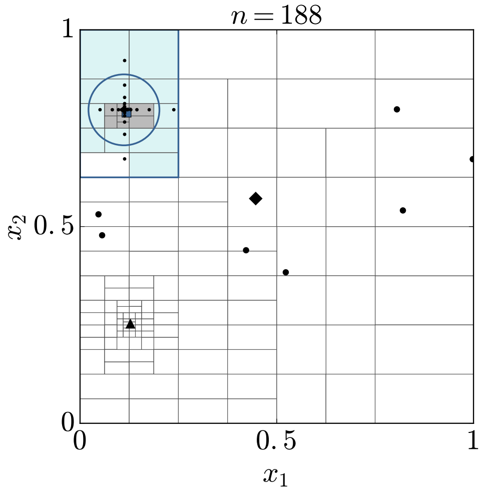

Figure 4.4: An example of the operation of the *Cluster* algorithm with a 2-dimensional test function, generated by the GKLS test function generator (function #27 from class #2, see Section 4.3.3 for description and Section 4.4.2 for the introduction to the generator test function classes).

Excluding explored regions. After the local optimization was performed on the cluster, the region might be considered well-explored and removed from further consideration by the global search.

A hyper-sphere with the radius equal to $0.2\delta_c$, centered at the best point found by the local search over the cluster, is used to determine the hyper-rectangles that need to be removed from D. The hyper-rectangles that have all vertices in this hyper-sphere are removed. The global search continues with the reduced set of hyper-rectangles.

The smallest hyper-rectangles are usually among the removed hyper-rectangles, therefore v increases and the further search becomes more global.

4.3.3 Illustrated Example

This section presents an illustrated example of the operation of the *Cluster* algorithm using the test function introduced in Section 4.1.3.

The operation of the *Cluster* algorithm is illustrated in Figure 4.4. The local minimizers are marked analogously to the Figure 4.1. The function evaluations produced by the global search part of the algorithm are the vertices of the displayed rectangles.

For this objective function the local refinement phase was performed once before the stopping condition (4.16) was met, namely, after $n = 188$ objective function evaluations produced by the global search. The figure corresponds to the moment when a new smallest rectangle was generated, having the area $v < v_{small}(d) = 2^{-15}$, triggering the local refinement phase.

The clustering algorithm produced $\lfloor |D|/10 \rfloor$ hyper-rectangle clusters, out of which a single cluster containing the rectangle with the lowest value was selected for further processing. The rectangles of this cluster are shaded in light blue. The representative point of the cluster is denoted by a square marker. It is evident that the cluster corresponds to the vicinity of a local minimizer. Moreover, the bounding box of the cluster rectangles BB_c, used to restrict the local optimization algorithm, is shown. The steps taken by the local optimization algorithm exploring this cluster are denoted as small black dots. A sphere with the radius $0.2\delta_c$ centered at the best point found by the local search is displayed. All rectangles that are fully enclosed by this sphere are colored in gray and removed from further processing. The rest of the rectangles constitute the set eligible for partitioning by the global search, that continues until either a stopping condition is satisfied or a small enough rectangle for the local refinement phase to be entered again is generated. The stopping condition (4.16) was satisfied in $n = 333$ (270 global and 63 local) objective function evaluations.

4.4 Numerical Experiments

This section presents the numerical experiments demonstrating and comparing the performance of the algorithms described in this chapter. Namely, different implementations of the *Rect* algorithm (*Rect-1*, *Rect-A* and *Rect-MC*) are compared as well as the algorithms derived from the *Rect-1* version of the *Rect* algorithm, i. e. the *GB* and *Cluster*. Additionally, some well-known algorithms of similar purpose are included for comparison. The parameter settings used for various algorithms are given in Table 4.1.

Table 4.1: Parameter settings of the algorithms.

Algorithm	Parameter	Value
Rect-1	-	-
Rect-A	Relative integration error	10^{-7}
Rect-MC	Number of evaluations of the d-dimensional integrand	10^4, if $d = 2$ 10^5, if $d = 3$
GB	Volume threshold, v_{small}	10^{-4}, if $d = 2$ 10^{-6}, if $d = 3$
	Period of repeating the *SECURITY* iteration, i_{period}	20
Cluster		
Simpl[16]	-	-
DIRECT[1]	Jones factor, ϵ	10^{-4}
	Maximum number of iterations	10^6
	Maximum number of hyper-rectangle divisions	10^4
Sobol[86]	-	-

The algorithms of this chapter have been implemented in C++ as part of this thesis. In all implementations the function evaluations database - a nested collection of self-balancing AVL binary trees [52] - is employed, so that the objective is never evaluated twice at the same location.

Experiments with two different test suites are presented, namely the 2- and 3-dimensional test suite from [34] (see Section 4.4.1) and a test suite produced by the GKLS generator [2, 29] including test functions up to the 4-th dimension (see Section 4.4.2).

4.4.1 Experiments with the 2- and 3-dimensional Test Suite

Experimental Setup. The testing functions from [34] used in numerous papers on Lipschitz optimization algorithms and defined in dimensions 2 and 3 were minimized in the first part of the experimental performance demonstration.

As a performance metric, the residual error on a logarithmic scale, computed after n objective function evaluatios, was used:

$$\delta_n = -\log_{10}(y_{on} - f^*), \tag{4.15}$$

where $y_{on} = \min_{i=1,\ldots,n} y_i$ and f^* is the global minimum value. Higher values of δ_n imply higher precision achieved.

The performance metric (4.15) can reasonably be applied to global optimization algorithms that, in theory, return to the vicinity of the global minimizer an infinite number of times. Since the global search part of the hybrid *Cluster* algorithm is not allowed to explore the local minimizers to an infinite precision, and the local search precision is parametrically controlled, the results according to metric (4.15) could hardly be properly interpreted, and therefore *Cluster* is not included in the comparison here.

In addition to the algorithms described in the present chapter, this subsection covers the results of two sequential global optimization algorithms intended for solving problems with similar formulation and one passive global optimization algorithm.

First, a recent algorithm [16], based on the Delaunay triangulation of the optimization region and referred to as *Simpl* in the discussion that follows, was included for comparison. The reason for choosing this algorithm is that both algorithms *Rect* and *Simpl* stem from the statistical models-based approach to global optimization; however, there are reasons to expect that the *Simpl* algorithm might be more efficient. This is because the interior of a hyper-rectangle is relatively worse represented by the information at its vertices, as compared to a simplex, and a single *Rect* algorithm iteration takes at most 2^{d-1} new function evaluations, as compared to only one in algorithm *Simpl*. The experimentation might show if these properties contribute to any noticeable performance difference. The *Simpl* algorithm results for the 2-dimensional case in Table 4.3 were copied from Table 1 of the original source, while the results for the 3-dimensional case in Table 4.5 were obtained using a C++ implementation produced as part of this thesis and using the 3-dimensional Delaunay triangulation code from the *CGAL* library [3].

The second sequential global optimization algorithm included is the well-known

Table 4.2: Comparison of different implementations of the *Rect* algorithm using the 2-dimensional test suite from [34]. Optimization stops after a predefined number of function evaluations N_{max}.

	Rect-1		**Rect-A**		**Rect-MC**	
	N_{max}		N_{max}		N_{max}	
f	1000	3000	1000	3000	1000	3000
1	5.30	5.80	5.80	6.84	5.80	6.84
2	6.93	6.93	6.93	11.00	6.93	11.00
3	13.80	14.30	11.88	11.88	12.91	14.30
3.1	14.16	14.16	11.88	11.88	13.12	14.30
3.2	14.16	14.16	11.88	11.88	13.12	14.30
3.3	13.43	14.30	11.87	11.87	13.37	14.30
4	∞	∞	∞	∞	∞	∞
5	4.14	6.01	6.01	6.01	6.01	6.01
6	4.49	4.49	4.49	6.21	4.49	6.21
7	11.83	12.38	2.81	12.38	2.88	12.38
8	6.12	6.23	8.53	8.53	8.53	8.53
9	∞	∞	∞	∞	∞	∞
9.1	-1.91	-1.91	∞	∞	∞	∞
9.2	-1.91	-1.91	10.61	10.61	∞	∞
9.3	∞	∞	∞	∞	∞	∞
10	∞	∞	12.27	12.27	∞	∞
11	12.34	∞	11.37	14.10	11.37	
12	7.42	11.78	7.42	11.78	7.42	11.78
13	1.51	1.56	1.51	1.59	1.51	1.59

DIRECT [48], which also relies on the rectangular partition of the feasible region. The open-source implementation [1] was used to produce the results for the 2- and 3-dimensional test suite.

Finally, a non-adaptive optimization algorithm, referred to as *Sobol*, was implemented using the so-called LP_τ sequences [86], and is known to be the worst case quasi-optimal since it generates the objective function evaluation locations nearly uniformly over the feasible region.

Results. Different implementations of the *Rect* algorithm are compared in Tables 4.2 and 4.4 by the value of (4.15) after a predefined number of objective function evaluations. Since an analytical expression of the integral in (4.6) is

Table 4.3: Comparison of different algorithms using the 2-dimensional test suite from [34]. Optimization stops after a predefined number of function evaluations N_{max}.

	Rect-1		**GB**		**Simpl**		**DIRECT**		**Sobol**	
	N_{max}		N_{max}		N_{max}		N_{max}		N_{max}	
f	3000	5000	3000	5000	3000	5000	3000	5000	3000	5000
1	5.30	5.80	5.30	5.80	5.30	6.86	4.14	∞	0.97	1.95
2	6.93	6.93	6.93	6.93	6.95	∞	3.71	3.71	1.82	2.01
3	13.80	14.30	3.96	4.73	∞	∞	5.58	5.58	1.36	4.54
3.1	14.16	14.16	3.44	4.36	∞	∞	5.58	5.58	3.44	3.44
3.2	14.16	14.16	3.44	4.36	∞	∞	5.58	5.58	1.80	1.80
3.3	13.43	14.30	3.31	4.55	∞	∞	5.58	5.58	1.55	2.33
4	∞	∞	∞	∞	2.97	3.29	∞	∞	∞	∞
5	4.14	6.01	3.97	6.01	6.01	6.01	4.48	4.48	0.76	1.37
6	4.49	4.49	2.66	4.49	4.25	4.25	1.78	1.78	-0.48	-0.48
7	11.83	12.38	1.82	3.03	25.56	29.61	3.74	6.67	1.31	1.31
8	6.12	6.23	3.83	6.12	6.23	8.58	4.70	4.70	0.99	0.99
9	∞	∞	∞	∞	∞	∞	∞	∞	0.50	1.28
9.1	-1.91	-1.91	∞	∞	∞	∞	∞	∞	-1.33	0.47
9.2	-1.91	-1.91	4.59	12.41	∞	∞	∞	∞	-0.23	-0.23
9.3	∞	∞	∞	∞	∞	∞	∞	∞	2.15	2.15
10	∞	∞	4.85	9.22	∞	∞	4.64	4.64	3.39	3.39
11	12.34	∞	5.30	6.06	8.04	8.04	∞	∞	2.86	3.00
12	7.42	11.78	4.65	6.02	2.38	2.38	4.32	4.32	3.22	4.24
13	1.51	1.56	1.51	1.51	1.37	1.58	1.29	1.86	1.15	1.38

Table 4.4: Comparison of different implementations of the *Rect* algorithm using the 3-dimensional test suite from [34]. Optimization stops after a predefined number of function evaluations N_{max}.

	Rect-1		**Rect-MC**	
	N_{max}		N_{max}	
f	3000	5000	3000	5000
20	∞	∞	∞	∞
21	10.58	12.92	10.58	12.89
22	9.66	13.32	12.19	13.32
23	0.84	1.20	0.88	1.19
24	12.21	13.57	10.53	13.57
25	4.00	4.00	4.00	4.00
26	∞	∞	∞	∞

Table 4.5: Comparison of different algorithms using the 3-dimensional test suite from [34]. Optimization stops after a predefined number of function evaluations N_{max}.

	Rect-1		**GB**		**Simpl**		**DIRECT**		**Sobol**	
	N_{max}		N_{max}		N_{max}		N_{max}		N_{max}	
f	3000	5000	3000	5000	3000	5000	3000	5000	3000	5000
20	∞	∞	∞	∞	∞	∞	3.15	3.37	1.86	1.86
21	10.58	12.92	3.91	3.91	4.45	4.61	∞	∞	1.82	1.82
22	9.66	13.32	3.12	3.67	4.11	4.52	15.91	17.82	1.52	2.13
23	0.84	1.20	0.84	1.20	0.94	0.94	1.18	1.18	1.44	1.44
24	12.21	13.57	6.45	6.45	3.88	3.88	∞	∞	1.61	1.61
25	4.00	4.00	4.00	4.00	1.18	1.28	4.47	4.47	1.20	1.20
26	∞	∞	∞	∞	-0.69	-0.60	2.52	2.52	-0.81	-0.81

not available in the 3-dimensional case, only versions *Rect-1* and *Rect-MC* of the *Rect* algorithm are included in Table 4.4. The original numbering of the objective functions, as on p. $468 - 469$ in [34], is given in the first column of the tables. Symbol ∞ means that the exact global minimum was found.

Analogous results for various other algorithms compared are presented in Tables 4.3 and 4.5.

Discussion. The results in Table 4.2 show that different versions of the *Rect* algorithm perform similarly in the 2-dimensional case with the exception of two problems (9.1 and 9.2), where *Rect-1* was unable to converge to the global minimum. There appears to be no significant performance difference in the 3-dimensional case as well, as Table 4.4 implies. This suggests that an efficient implementation *Rect-1* could be the right choice for higher dimensional problems as well.

The comparison of the algorithms in Tables 4.3 and 4.5 shows that the *Rect* algorithm for the test suite in question seems to perform the best, as it reaches the highest precision both in the 2- and 3-dimensional experiments. As could be expected, the results of the non-adaptive *Sobol* algorithm are the worst.

It can be seen that the various versions of the *Rect* algorithm perform com-

Table 4.6: The parameters of the GKLS test function classes.

Class	**d**	$\mathbf{f^*}$	$\mathbf{n_{local}}$	$\mathbf{r^*}$	ρ^*	Δ
1	2	-1	10	0.9	0.2	10^{-4}
2	2	-1	10	0.9	0.1	10^{-4}
3	3	-1	10	0.66	0.2	10^{-6}
4	3	-1	10	0.9	0.2	10^{-6}
5	4	-1	10	0.66	0.2	10^{-6}
6	4	-1	10	0.9	0.2	10^{-6}

parably with algorithm *Simpl* in the 2-dimensional case, however, contrary to the expectations discussed, *Simpl* performs worse in the 3-dimensional case. The performance of *GB* and *DIRECT* is similar in the 2-dimensional case and is slightly worse than that of *Rect* algorithm implementations. In the 3-dimensional case, *Rect-1* and *DIRECT* produced the best results.

4.4.2 Experiments with the GKLS Function Generator

Experimental Setup. In the second part of the experimental investigation the test functions produced by the free implementation [2] of the GKLS test function generator [29] were minimized. A methodology similar to that of [78] was followed.

The GKLS generator produces multi-extremal and multivariate testing functions of desirable differentiability for testing box-constrained global optimization algorithms. Each testing function is obtained by systematically distorting a convex quadratic function by polynomials forming the surface spikes. One hundred functions are available per function class that is fully reproducible by supplying five parameters:

1. d, problem dimension;
2. n_{local}, the number of local minima;
3. f^*, the value of the global minimum;
4. r^*, the distance between the paraboloid vertex and the global minimizer;

5. ρ^*, the radius of the attraction region of the global minimizer.

In this work 6 continuously differentiable GKLS function classes from [78], defined over $A = [-1, 1]^d$, were used. The parameters of these classes are given in Table 4.6. It is noteworthy that the complexity of the objective functions grows when parameter r^* grows or parameter ρ^* decreases. As a result, for each dimension d, the second class in Table 4.6 is more complicated than the first one.

The global optimization problems in this experiment are considered difficult. The performance of the base global search algorithm *Rect-1*, as well as algorithms derived from it, i. e. *GB* and *Cluster*, was assessed to determine the ability of the algorithms to handle complicated global optimization problems. It is important to check how the suggested modifications compare to the global search algorithm *Rect-1* they are based on as well as to each other. Moreover, a comparison with the popular algorithms is also important. Therefore the results of the *DIRECT* [48] and its locally-biased modification *DIRECTl* [26, 27], as they originally appeared in [78], are included. According to the source [78], both algorithms were implemented according to the description in [25].

The algorithms were run with a budget of $N_{max} = 1000000$ function evaluations until a trial point $\mathbf{x}_i = (x_{i1}, \dots, x_{id}), i = 1, \dots, n,$ close to the global minimizer $\mathbf{x}^* = (x_1^*, \dots, x_d^*)$ was generated such that

$$|x_{ij} - x_j^*| \leqslant \sqrt[d]{\Delta}|b_j - a_j|, j = 1, \dots, d, \tag{4.16}$$

where the feasible region is $A = \prod_{i=1}^{d}[a_i, b_i]$ and Δ is a precision parameter. The values for Δ are also given in Table 4.6.

Results. Tables 4.7 and 4.8 show $\max_{i=1,\dots,100} n_i$ and $\frac{1}{100}\sum_{i=1}^{100} n_i$, respectively, for each of the test classes, where n_i denotes the number of function evaluations performed before the stopping condition (4.16) was met. The notation $10^6(k)$ means that for k functions in a given class the global minimizer was not found after conducting N_{max} function evaluations. The best result in each row is highlighted. The results are displayed graphically in Figure 4.5.

Table 4.7: The maximum number of trials for the GKLS classes: $\max_{i=1,\ldots,100} n_i$.

Class	Rect-1	GB	Cluster	DIRECT	DIRECTl
1	2121	**449**	463	1159	2318
2	9833	1401	**1130**	3201	3414
3	12595	5626	**5582**	12507	13309
4	46024	8355	**7589**	$> 10^6(4)$	29233
5	131341	41210	**26394**	$> 10^6(4)$	118744
6	416556	47597	**44272**	$> 10^6(4)$	287857

Table 4.8: The average number of trials for the GKLS classes: $\frac{1}{100}\sum_{i=1}^{100} n_i$.

Class	Rect-1	GB	Cluster	DIRECT	DIRECTl
1	342.70	218.08	248.71	**198.89**	292.79
2	1485.15	**607.75**	662.33	1063.78	1267.07
3	3958.98	2759.15	3200.74	**1117.70**	1785.73
4	8027.53	**3765.39**	4364.15	>42322.65	4858.93
5	19181.31	13176.30	**10660.82**	>47282.89	18983.55
6	39642.24	18097.49	**16953.56**	>95708.25	68754.02

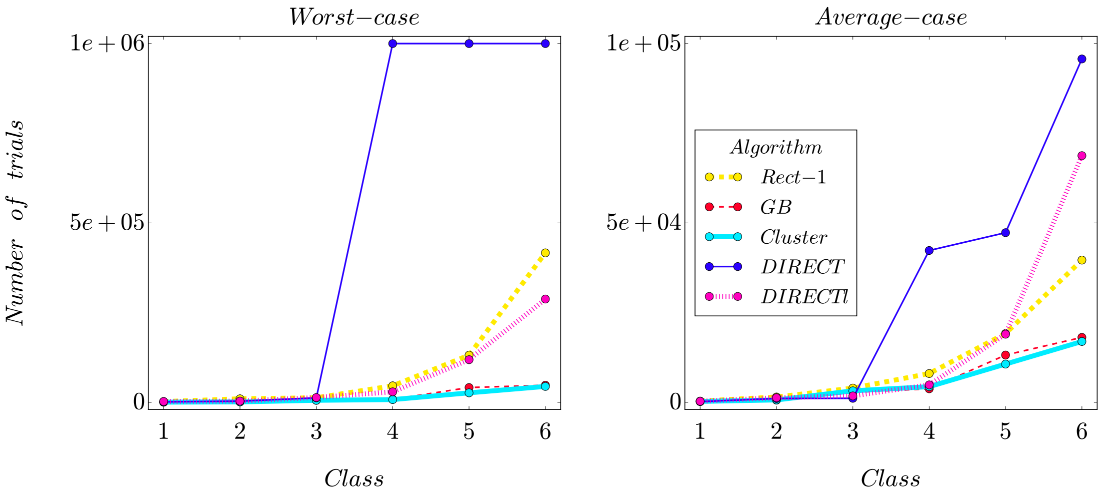

Figure 4.5: The number of trials for the GKLS test classes. **Left:** The worst-case: $\max_{i=1,\ldots,100} n_i$. **Right:** The average-case: $\frac{1}{100}\sum_{i=1}^{100} n_i$.

Discussion. The results in Tables 4.7 and 4.8 show that algorithms *Cluster* and *GB* exhibit similar performance and produce the best results for the given family of test functions. These extensions of the *Rect-1* global search strategy consume considerably fewer function evaluations in the average and worst cases than the original algorithm. *Cluster* worst-case performance is the best in general.

It can be seen that *Rect-1* performs worse than *DIRECT* for the first three classes, but better for the rest of the classes, as it manages at least to converge for all objective functions. *Rect-1* performs worse than *DIRECTl* in general.

4.5 Chapter Summary and Conclusions

1. Different implementations of a statistical hyper-rectangular decomposition-based global optimization algorithm *Rect* were described. Implementation *Rect-1* is efficiently computable in any dimension and results in no significant difference as compared to other implementations, involving integral approximation techniques. *Rect-1* is therefore suitable to serve as a basis for derived global optimization algorithms.

2. The performance of the *Rect* algorithm is similar to that of *DIRECT* and the simplicial decomposition-based global search algorithm *Simpl*.

3. Due to the fact that the *Rect-1* algorithm does not enter the fast convergence mode in a moderate number of iterations, two heuristic global search acceleration techniques were proposed, preventing *Rect-1* from excessively exploring the vicinity of the suboptimal local minimizers. Specifically, algorithm *GB* operates by switching between explicitly defined global and local search phases based on improvement achieved in the current phase. On the other hand, algorithm *Cluster* employs a clustering procedure to identify the well-explored regions.

4. The extensions *GB* and *Cluster* consume considerably fewer function evaluations than the original algorithm *Rect-1* optimizing difficult multi-modal global optimization problems. For such problems *Cluster* worst-case performance is the best among the considered algorithms.

Chapter 5

Asymptotic Properties in Simplicial Statistical Global Optimization

In the present chapter a theoretical investigation of certain properties of a stationary isotropic Gaussian random field is carried out in order to support a feature of a recent global optimization algorithm. More precisely, a stationary isotropic Gaussian random field is considered as a statistical model of an objective function in the context of global optimization. An asymptotic expression of an improvement probability-related criterion for this assumed statistical model in the setting of simplicial global optimization is sought. To this end, the asymptotic behavior of the conditional mean and variance of the random field with respect to the random field values known at the vertices of a simplex is investigated, assuming that the simplex is contracting. The obtained result allows for driving a correspondence between the probability of improvement related criterion and a simply-computable, but heuristically justified simplex selection criterion in a recent simplicial global optimization algorithm, primarily targeting bivariate problems. Furthermore, the results in this chapter enable to simplify the probability of improvement in a higher-dimensional case.

The first section of this chapter provides the context of the research. The second section introduces the problem of the correspondence between the probability of improvement and the computationally efficient simplex selection criterion in question. A detailed statement of a theoretical link being sought is given in the third section. Finally, the derivation of relevant asymptotic properties of the random field and the resulting theorem, formalizing the targeted correspondence, are presented in the fourth section.

5.1 Introduction

A global optimization of an expensive *black-box* objective function is considered. A well-established approach in this situation is to resort to the rational decisions theory and statistical models of uncertainty, organizing the global optimization process as a sequence of rational decisions under uncertainty.

A global optimization approach based on the statistical models of the objective function originally suffered from intensive auxiliary computations, involved in the optimization of a statistical criterion value, as the probability of improvement in the *P-algorithm* (2.18) or the expected improvement in the *Maximum expected improvement algorithm* (2.17) to determine the next function evaluation location. This limited the applicability of these methods to expensive objective functions to counterbalance the associated cost of auxiliary computations. To extend the class of functions that could be targeted by these methods, computational simplifications are required.

Similarly, in the past, Lipschitzian optimization methods were known to involve heavy computations and therefore apply to expensive objective functions only [40]. However, the introduced advanced modifications, involving various decompositions [9, 19, 43, 55, 59, 73, 78, 83] of the feasible region, reduced the computational burden and expanded the applicability of these methods. Similar simplifications could also be applied to the methods based on the statistical models of the objectives.

The idea of adapting a statistical model of the objective for the decomposed feasible region was implemented in a recent algorithm [16], where the subsets in the decomposition are simplices, obtained by means of the Delaunay triangulation. The algorithm is interesting as the authors provide an established convergence rate for it in the two-dimensional case, although the algorithm itself can be applied in higher dimensions as well. In the present chapter asymptotic properties of the statistical model are proved which, in a two-dimensional case, provide a theoretical basis for the computational simplification employed by this global optimization algorithm as well as its certain generalization to higher dimensions.

5.2 Motivation of the Research

This section introduces the problem of establishing the theoretical correspondence between the probability of improvement in the *P-algorithm* and a relatively simple heuristic selection criterion in a recent simplicial decomposition-based global optimization algorithm [16]. First, certain specialized forms of the *P-algorithm*, defined by (2.18) in the most general form, are given. Then, the selection criterion for a simplex in algorithm [16] is given. The specific correspondence being sought is introduced here and is further detailed in Section 5.3.

A global optimization problem $\min_{\mathbf{x}\in A} f(\mathbf{x})$ is considered, where $f(\mathbf{x})$ is expensive and *black-box*.

Let us recall the general form of the *P-algorithm*, this time with an altered definition of the minimal target value at the current optimization step, i. e. $y_{on} < \min_{i=1,\dots,n} y_i$:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in A} \mathbb{P}(\xi(\mathbf{x}) \leqslant y_{on} | \xi(\mathbf{x}_i) = y_i, i = 1, \dots, n), \tag{5.1}$$

where the probability in question is defined by the chosen statistical model $\xi(\mathbf{x}), \mathbf{x} \in A$.

Normally, the random variables $\xi(\mathbf{x})$ are assumed to be distributed according to the Gaussian probability density. In that case, the probability in (5.1) is defined by the formula

$$\mathbb{P}(\xi(\mathbf{x}) \leqslant y_{on} \,|\, \xi(\mathbf{x}_i) = y_i, i = 1, \dots, n) = \Phi\left(\frac{y_{on} - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \dots, n)}\right), \tag{5.2}$$

where $\Phi(\cdot)$ denotes the Gaussian cumulative distribution function, while $m(\cdot)$ and $s^2(\cdot)$ denote the conditional mean and conditional variance of $\xi(\mathbf{x})$, respectively.

Since $\Phi(t)$ is monotonically increasing, maximizing the probability of improvement (5.2) is equivalent to maximizing

$$\frac{y_{on} - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}. \tag{5.3}$$

It was shown in [16, 111, 112] that an algorithm maximizing (5.3) to select a new function evaluation location can be justified without the Gaussian assumption as well.

The expressions for $m(\cdot)$ and $s(\cdot)$ in (5.2) and (5.3) are complicated and require time-consuming computations. Fortunately, the prospect of computational simplifications appeared with optimization algorithms based on simplicial decompositions of the feasible region and simplicial statistical models, as in such case only trials closest to $\mathbf{x} \in A$ could be used in the definitions of $m(\cdot)$ and $s(\cdot)$, as opposed to the full set of the trials performed.

A simplicial statistical model originally appeared in the two-dimensional *Select and clone* algorithm by [110]. The auxiliary maximization over the complete feasible region at every optimization step was replaced by the maintainance of the priority queue comprised of triangular sub-regions of A, where the priority of the triangle S_i is equal to the maximum of an equivalent to probability of improvement (5.3) over the triangle:

$$\pi_i = \arg\max_{\mathbf{x} \in S_i} \left(\frac{y_{on} - m(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)}{s(\mathbf{x}|\xi(\mathbf{x}_i) = y_i, i = 1, \ldots, n)} \right). \tag{5.4}$$

The point $\mathbf{x}_{max}$, maximizing expression (5.4), was given explicitly for a regular triangle. The algorithm was later generalized for higher dimensions in [113].

Another algorithm [16] bases the global optimization process on a statistical model, adjusted for a simplicial decomposition of the feasible region by means of the Delaunay triangulation. The algorithm has an established convergence rate in the bivariate case, although is applicable to higher dimensions as well. An initial simplicial covering is iteratively refined, selecting simplices based on a computationally efficient criterion, further reducing the computational burden, as compared to *Select and clone*. The criterion is expressed as follows:

$$\eta_i = \frac{V(S_i)}{\frac{1}{d+1}\sum_{j=1}^{d+1} z_j - y_{on}}, \tag{5.5}$$

where $V(S_i)$ is the volume of simplex S_i, $\mathbf{z} = (z_1, z_2, \ldots, z_{d+1})$ are the values of the objective function at the vertices of the simplex, and $y_{on} < \min_{j=1,\ldots,n} y_j$ is the target value at step $n+1$.

The priority criterion (5.5) was justified only heuristically in [16]. Therefore it seemed reasonable to determine its mathematical relationship with the probability of improvement (5.2) or, alternatively, with its equivalent in terms of maximization (5.3). Let us note that this relationship is clear in the case of *Select and clone* algorithm from (5.4). It will be shown in the following section that under some assumptions the criterion (5.5) is equivalent to (5.3) at the center of a regular simplex.

5.3 Statement of the Problem

Let $\xi(\mathbf{x})$, $\mathbf{x} \in A \subseteq \mathbb{R}^d$, be a stationary isotropic Gaussian random field with the mean value μ, variance σ^2, and the correlation function $\rho(\tau) = \exp(-c\tau^2)$, $\tau \geqslant 0$. Let the values of $\xi(\mathbf{x})$ be known at the vertices $\mathbf{a}_i, i = 1, \ldots, n+1,$ of a regular n-simplex with edge length equal to δ, such that $\xi(\mathbf{a}_i) = z_i$, $i = 1, \ldots, n+1$. Let $\mathbf{a}$ be the weight center of the simplex.

The behavior of the conditional mean and conditional variance of $\xi(\mathbf{a})$ with respect to $\mathbf{a}_i, z_i, i = 1, \ldots, n+1,$ when the edge length of the simplex vanishes, i. e. $\delta \to 0$, constitutes an important part of the discussion that follows. To compute the conditional mean and variance, the correlation coefficients between $\xi(\mathbf{a}_j)$ and $\xi(\mathbf{a}_k)$ as well as between $\xi(\mathbf{a}_j)$ and $\xi(\mathbf{a})$, are needed. The former is obviously equal to $\rho_{jk} = \exp(-c\delta^2)$, and the latter is equal to $\rho_j = \exp(-c\delta^2 \frac{d}{2(d+1)})$, as the distance between $\mathbf{a}$ and $\mathbf{a}_j$ is equal to $\delta\sqrt{\frac{d}{2(d+1)}}$. In the upcoming discussion, the following notation and expressions of conditional mean and conditional variance will be used:

$$\begin{aligned}
&\mathbb{E}(\xi(\mathbf{a})|\xi(\mathbf{a}_i) = z_i, i = 1, \ldots, d+1) = m(\mathbf{a}|\mathbf{a}_i, z_i, i = 1, \ldots, d+1) = \\
&= M(\delta, \mathbf{z}) = \mu + \exp\left(-c\delta^2 \frac{d}{2(d+1)}\right) \cdot \mathbf{I}^T \cdot C^{-1} \cdot (\mathbf{z} - \mu\mathbf{I}), \qquad (5.6) \\
&\mathrm{Var}(\xi(\mathbf{a})|\xi(\mathbf{a}_i) = z_i, i = 1, \ldots, d+1) = s^2(\mathbf{a}|\mathbf{a}_i, z_i, i = 1, \ldots, d+1) = \\
&= S^2(\delta) = \sigma^2 \left(1 - \exp\left(-c\delta^2 \frac{d}{d+1}\right) \cdot \mathbf{I}^T \cdot C^{-1} \cdot \mathbf{I}\right), \qquad (5.7)
\end{aligned}$$

where **I** denotes the $(d+1)$-dimensional unit vector, $\mathbf{z} = (z_1, z_2, \ldots, z_{d+1})^T$, and C is a $(d+1) \times (d+1)$ matrix with all the elements equal to $\exp(-c\delta^2)$ except the diagonal elements which are equal to 1.

Using the notation just introduced, the equivalent of the improvement probability (5.3) at the center of the considered simplex can be expressed as follows:

$$p(\delta) = \frac{y_{on} - M(\delta, \mathbf{z})}{S(\delta)}, \qquad (5.8)$$

where the essential variables are presented explicitly. To avoid computations involving fractions with denominators close to zero, instead of $p(\delta)$ its reciprocal value with a reverse sign $\hat{p}(\delta)$ should be used for the implementation of the algorithm as well as in a further analysis:

$$\hat{p}(\delta) = \frac{S(\delta)}{M(\delta, \mathbf{z}) - y_{on}}. \qquad (5.9)$$

On the other hand, the heuristically justified criterion (5.5) adapted for a regular simplex can be written as follows:

$$\eta(\delta) = \frac{Q}{\tilde{z} - y_{on}}, \qquad (5.10)$$

where $\tilde{z} = \frac{1}{d+1} \sum_{i=1}^{d+1} z_i$, Q is the volume of the considered simplex and $y_{on} < \min_{i=1,\ldots,d+1} z_i$.

The goal is to show that in the 2-dimensional case ($d = 2$) criterion $\eta(\delta)$ (5.10), i. e. a version of η_i (5.5) for a regular simplex, well approximates the equivalent of the improvement probability at the center of a regular simplex $\hat{p}(\delta)$ (5.9) for small simplices, i. e. as $\delta \to 0$. It is expected that for $d > 2$ a similar expression to η_i (5.5) can approximate $\hat{p}(\delta)$ (5.9) as well.

5.4 Assessment of Approximation

Both criteria $\hat{p}(\delta)$ and $\eta(\delta)$, compared in this chapter, obviously converge to 0, as $\delta \to 0$. For the case $d = 2$, the asymptotic behavior of $\eta(\delta)$, as $\delta \to 0$, becomes clear after replacing Q in (5.10) by its expression via δ:

$$\eta(\delta) = \frac{\sqrt{3}\delta^2}{4(\tilde{z} - y_{on})}. \tag{5.11}$$

The rest of this section is devoted to the investigation of the asymptotic behavior of $\hat{p}(\delta)$ (5.9).

To begin with, it is necessary to derive an explicit form of C^{-1}, as it is included into an expression of $\hat{p}(\delta)$ via $M(\delta, \mathbf{z})$ and $S^2(\delta)$. Matrix C has the structure analogous to the $m \times m$ matrix

$$U = \begin{pmatrix} 1 & a & \dots & a \\ a & 1 & \dots & a \\ \dots & \dots & \dots & \dots \\ a & a & \dots & 1 \end{pmatrix}, \text{ where } a \in \mathbb{R}. \tag{5.12}$$

Lemma 5.4.1. *The following equality holds*

$$V = U^{-1} = \frac{1}{u} \begin{pmatrix} t & a & \dots & a \\ a & t & \dots & a \\ \dots & \dots & \dots & \dots \\ a & a & \dots & t \end{pmatrix}, \tag{5.13}$$

where $t = -(m-2)a - 1$*, and* $u = (m-1)a^2 - (m-2)a - 1$.

Proof. The statement is proved simply by computing the elements of $W = U \cdot V$:

$$\begin{aligned} w_{ii} &= \big((m-1)a^2 + t\big)/u = \big((m-1)a^2 - (m-2)a - 1\big)/u = 1, \\ w_{ij} &= \big(a + at + (m-2)a^2\big)/u = \\ &= \big(a + a(-(m-2)a - 1) + (m-2)a^2\big)/u = 0,\ i \neq j. \end{aligned} \tag{5.14}$$

Thus, we have proved that W is a unit matrix. □

Corollary 5.4.2. *The inverse correlation matrix C^{-1} is a $(d+1) \times (d+1)$ matrix of the structure presented by formula (5.12), where $t = -(d-1)\exp(-c\delta^2) - 1$, and $u = d\exp(-2c\delta^2) - (d-1)\exp(-c\delta^2) - 1 = (d\exp(-c\delta^2) + 1)(\exp(-c\delta^2) - 1)$.*

In order to investigate the convergence of $\tilde{p}(\delta)$ to 0, as $\delta \to 0$, we start from the convergence of its constituent parts, $M(\delta, \mathbf{z})$ and $S^2(\delta)$.

The substitution of C^{-1} in (5.6) by its expression defined in Corollary 5.4.2 yields

$$\begin{aligned} M(\delta, \mathbf{z}) &= \mu + \exp\left(-c\delta^2 \frac{d}{2(d+1)}\right) \cdot \mathbf{I}^T \cdot \frac{1}{u} \begin{pmatrix} t & a & \dots & a \\ a & t & \dots & a \\ \dots & \dots & \dots & \dots \\ a & a & \dots & t \end{pmatrix} \cdot (\mathbf{z} - \mu\mathbf{I}) = \\ &= \mu + \frac{1}{u}\exp\left(-c\delta^2 \frac{d}{2(d+1)}\right)\big(\exp(-c\delta^2) - 1\big)\,\mathbf{I}^T \cdot (\mathbf{z} - \mu\mathbf{I}) = \\ &= \mu + \frac{\exp\left(-c\delta^2 \frac{d}{2(d+1)}\right)(\exp(-c\delta^2) - 1)\sum_{i=1}^{d+1}(z_i - \mu)}{(d\exp(-c\delta^2) + 1)(\exp(-c\delta^2) - 1)} = \\ &= \mu + \frac{\exp\left(-c\delta^2 \frac{d}{2(d+1)}\right)\sum_{i=1}^{d+1}(z_i - \mu)}{d\exp(-c\delta^2) + 1}. \end{aligned} \tag{5.15}$$

The expansion

$$\exp(-cx^2) = 1 - cx^2 + o(x^2), \tag{5.16}$$

applied to all the exponential terms in (5.15), gives the following asymptotic expression of $M(\delta, \mathbf{z})$:

$$
\begin{aligned}
M(\delta, \mathbf{z}) &= \mu + \frac{(1 - c\delta^2 \frac{d}{2(d+1)} + o(\delta^2)) \sum_{i=1}^{d+1} (z_i - \mu)}{d + 1 - dc\delta^2 + o(\delta^2)} = \\
&= \tilde{z} + o(\delta). \qquad (5.17)
\end{aligned}
$$

Similarly, the substitution of C^{-1} in (5.7) by its expression defined in Corollary 5.4.2 yields

$$
\begin{aligned}
S^2(\delta) &= \sigma^2 \left(1 - \exp\left(-c\delta^2 \frac{d}{d+1}\right) \cdot \mathbf{I}^T \cdot \frac{1}{u} \begin{pmatrix} t & a & \dots & a \\ a & t & \dots & a \\ \dots & \dots & \dots & \dots \\ a & a & \dots & t \end{pmatrix} \cdot \mathbf{I} \right) = \\
&= \sigma^2 (1 - \frac{1}{u} \exp\left(-c\delta^2 \frac{d}{d+1}\right) \cdot \\
&\cdot (-(d+1)((d-1)\exp(-c\delta^2) + 1) + (d^2 + d)\exp(-c\delta^2))) = \\
&= \sigma^2 \left(1 - \frac{\exp\left(-c\delta^2 \frac{d}{d+1}\right)(d+1)\left(\exp(-c\delta^2) - 1\right)}{(d\exp(-c\delta^2) + 1)(\exp(-c\delta^2) - 1)} \right) = \\
&= \sigma^2 \left(1 - \frac{(d+1)\exp(-\frac{cd}{d+1}\delta^2)}{d\exp(-c\delta^2) + 1} \right). \qquad (5.18)
\end{aligned}
$$

The expansion

$$
\exp(-cx^2) = 1 - cx^2 + \frac{1}{2}c^2x^4 + o(x^4), \qquad (5.19)
$$

applied to all the exponential terms in (5.18), yields the following asymptotic expression of $S^2(\delta)$:

$$
\begin{aligned}
S^2(\delta) &= \sigma^2 \frac{d\exp(-c\delta^2) - (d+1)\exp(-\frac{cd}{d+1}\delta^2) + 1}{d\exp(-c\delta^2) + 1} = \\
&= \sigma^2 \frac{d(1 - c\delta^2 + \frac{1}{2}c^2\delta^4 + o(\delta^4)) - (d+1)(1 - \frac{cd}{d+1}\delta^2 + \frac{1}{2}\left(\frac{cd}{d+1}\right)^2\delta^4 + o(\delta^4)) + 1}{d\exp(-c\delta^2) + 1} = \\
&= \sigma^2 \frac{d(-c\delta^2 + \frac{1}{2}c^2\delta^4 + o(\delta^4)) - (d+1)(-\frac{cd}{d+1}\delta^2 + \frac{1}{2}\left(\frac{cd}{d+1}\right)^2\delta^4 + o(\delta^4))}{d\exp(-c\delta^2) + 1} = \\
&= \sigma^2 \frac{-cd\delta^2 + \frac{1}{2}c^2d\delta^4 + o(\delta^4) + \frac{cd(d+1)}{d+1}\delta^2 - \frac{1}{2}\left(\frac{cd}{d+1}\right)^2(d+1)\delta^4 + o(\delta^4)}{d\exp(-c\delta^2) + 1} = \\
&= \sigma^2 \frac{(-cd + cd)\,\delta^2 + \frac{1}{2}\left(c^2d - \frac{c^2d^2}{d+1}\right)\delta^4 + o(\delta^4)}{d\exp(-c\delta^2) + 1} = \\
&= \sigma^2 \frac{\frac{c^2d}{2(d+1)}\delta^4 + o(\delta^4)}{d\exp(-c\delta^2) + 1} = \sigma^2 \frac{\frac{c^2d}{2(d+1)}\delta^4 + o(\delta^4)}{d(1 - c\delta^2 + o(\delta^2)) + 1} = \\
&= \sigma^2 \frac{\frac{c^2d}{2(d+1)}\delta^4 + o(\delta^4)}{d + 1 + d(-c\delta^2 + o(\delta^2))} = \sigma^2 \frac{\frac{c^2d}{2(d+1)}\delta^4 + o(\delta^4)}{d + 1 + o(\delta)} = \\
&= \sigma^2 c^2 \delta^4 \frac{d}{2(d+1)^2} + o(\delta^4). \qquad (5.20)
\end{aligned}
$$

The obtained assessments of asymptotic expressions (5.17) and (5.20) can be summarized as the following theorem.

Theorem 5.4.3. *The following equation is valid*

$$\tilde{p}(\delta) = \frac{\sigma c \delta^2}{(d+1)(\tilde{z} - y_{on})}\sqrt{\frac{d}{2}} + o(\delta^2). \qquad (5.21)$$

Corollary 5.4.4. *In the case $d = 2$, the following relation of asymptotic equivalence is valid*

$$\tilde{p}(\delta) \sim \frac{4\sqrt{3}}{9}\sigma c \eta(\delta). \qquad (5.22)$$

Thus it has been demonstrated that in the 2-dimensional case for a regular triangle, the heuristically justified and efficiently computable criterion (5.5) is proportional to the analogue of the improvement probability at the center of such triangle (5.9), i. e. the theoretical justification of the criterion (5.5) has been established.

In a higher dimensional case, an expression proportional to δ^2 and depending on the dimension can be used to approximate the analogue of the probability of improvement (5.9).

5.5 Chapter Summary and Conclusions

1. Two criteria used in the construction of global optimization algorithms based on simplicial statistical objective function models are considered. The first criterion is an analogue to the probability of improvement at the current optimization step. The other one is defined by a computationally simpler formula, but lacks a strict theoretical justification. The asymptotic equivalence of both criteria is demonstrated for contracting simplices in the bivariate case. The obtained result theoretically supports the application of the computationally simpler criterion in the bivariate case.

2. Insight into how the first criterion could be approximated in higher dimensions is provided. This result could be used in new optimization algorithms.

Chapter 6

Worst-Case Optimality in Univariate Bi-objective Lipschitz Optimization

This chapter deals with the problem of univariate bi-objective optimization, where each of the objectives satisfies the Lipschitz condition. The optimality of an algorithm iteratively refining a division of the feasible interval into smaller subintervals by trisecting a selected one is investigated. An interest in optimal algorithms can be attributed to their sharing certain properties with other, practically applicable algorithms. In this case, trisection was previously used in a number of single-objective multivariate algorithms.

An introduction in the first section of this chapter is followed by an interpretation of the well-known *Pijavskij-Shubert* algorithm [70, 84] with respect to the concept of optimality in the second section. A presentation of an optimal trisection in the univariate single-objective Lipschitz optimization is also included. The third section details the main results of this chapter, dealing with the bi-objective case. Specifically, the tolerance of the lower Lipschitz bound over an interval is generalized to arbitrary subintervals of the feasible interval. The one-step worst-case optimality of trisecting an interval with respect to the resulting tolerance is established. The trisection-based algorithm is introduced. Finally, some numerical examples illustrating the performance of the algorithm are provided in the fourth section.

6.1 Introduction

It has recently become popular to approach multi-objective non-convex optimization problems by meta-heuristic methods, despite the unexploited potential of theoretically justified methods based on mathematical models. An interesting task from the theoretical point of view is construction and investigation of the properties of algorithms, optimal with respect to certain mathematical models of non-convex problems. Two classical types of optimality - the average case [6] and the worst-case [74] - have been thoroughly investigated in the single-objective non-convex optimization case. Some of these ideas can be carried over to the field of multi-objective optimization.

For example, it was shown in [97] that for the objective functions satisfying the Lipschitz condition a worst-case optimal bi-objective algorithm amounts to covering the feasible region by balls of minimum radius, analogously to the single-objective case, as determined considerably earlier in [91]. Similarly, the well-known one-step optimal *Pijavskij-Shubert* algorithm [70, 84] was generalized to the bi-objective case in [98].

It is true, however, that the applicability of optimal algorithms is narrow, either due to difficulties in practically checking the relevant assumptions or prohibitive implementation complexity. Nevertheless, the design of practically useful algorithms benefits from an understanding of the properties of the optimal algorithms, as some of them might be shared [67, 73, 88, 106].

The present chapter investigates the optimality of trisecting an interval in an iterative univariate bi-objective optimization algorithm with the assumption of both objectives satisfying the Lipschitz condition. At each step of optimization an algorithm selects an interval for division into three parts by placing new trials in a way, that is shown to be worst-case optimal with respect to a specific criterion.

The favourable properties of using trisection in the univariate single-objective optimization were demonstrated in [54]. An optimal univariate algorithm for bi-objective problems is interesting in the view of developing multivariate algorithms, using the adaptive diagonal partitions of hyper-rectangles. The single-objective multivariate algorithms utilizing this scheme are detailed in

[55, 79–82]. The results in the present chapter support the type of trisection used in [115].

This chapter deals with optimality of partitioning an interval. Other methods have also considered optimal interval partitioning, however, differently defined. The latter circumstance is illustrated by starting with an optimal partitioning in the single-objective case, including a specific interpretation of the well-known *Pijavskij-Shubert* algorithm [70, 84]. The main results, i. e. the treatment of the bi-objective case, follow immediately afterwards together with some illustrative experiments.

6.2 One-Step Worst-Case Optimal Methods for Single-Objective Univariate Optimization

6.2.1 Bisection in the Single-Objective Case

Let us consider a univariate single-objective Lipschitz optimization problem

$$\min_{x \in A} f(x), \tag{6.1}$$

$$|f(u) - f(t)| \leqslant L|u - t|, \; u, t \in A, \; L > 0, \tag{6.2}$$

where A is a closed interval, and L is the Lipschitz constant.

A well-known, theoretically justified and intuitively perceptible *Pijavskij-Shubert* [70, 84] algorithm, identified by the names of its authors, solves the problem (6.1) by iteratively updating the lower Lipschitz bound of the objective function values $F_n(x)$ and performing the next function evaluation at the global minimizer of $F_n(x)$:

$$F_n(x) = \max_{i=1,\ldots,n} (y_i - L|x - x_i|), \tag{6.3}$$

$$x_{n+1} = \arg \min_{x \in A} F_n(x), \tag{6.4}$$

where the objective function evaluations previously performed are $x_i, y_i = f(x_i)$, $i = 1, \ldots, n$. This algorithm is presented here as apparently the first one-step worst-case optimal global optimization algorithm. The notion of optimality, with respect to which this algorithm is interpreted, is explained below to facilitate a unified analysis with the other cases of interest to this work, e. g. trisection based algorithms.

In the algorithm in question an estimate of the global minimum after n objective function evaluations is the best value found so far $y_{on} = \min_{i=1,\dots,n} y_i$. Therefore the worst-case error of the estimate with respect to the available information is equal to

$$\Delta_n(\mathbf{X}_n, \mathbf{Y}_n) = \max_{f(\cdot)\in\Phi(L,n)} (y_{on} - \min_{x\in A} f(x)) = \tag{6.5}$$

$$= y_{on} - \min_{x\in A} F_n(x), \tag{6.6}$$

where $\Phi(L, n)$ is the subclass of Lipschitz functions satisfying (6.2) and

$$\mathbf{X}_n = (x_1, \dots, x_n), \mathbf{Y}_n = (f(x_1), \dots, f(x_n)). \tag{6.7}$$

The notion of optimality is explained here by selecting the next function evaluation location x_{n+1} to minimize the potential error, resulting from this choice:

$$x_{n+1} = \arg\min_{x\in A} \max_{f(\cdot)\in\Phi(L,n)} \Delta_{n+1}((\mathbf{X}_n, x), (\mathbf{Y}_n, f(x))). \tag{6.8}$$

The equivalence of (6.8) to the following expression can be easily demonstrated:

$$x_{n+1} = \arg\min_{x\in A} F_n(x). \tag{6.9}$$

The function $F_n(x)$ is piecewise linear, as is obvious from (6.3), and its global minimizer is computable using a simple analytical formula. The definition of $x_{n+1} \in [x_j, x_{j+1}]$, is illustrated on the left side of Figure 6.1, where x_{n+1} is denoted by a star. The shaded subinterval indicates the part of the search region where objective function values below the already known record y_{on} are still possible. The point x_{n+1} is the center of this subinterval. Moreover, its location coincides with that defined by the optimal algorithm [91] with the budget of one objective function evaluation.

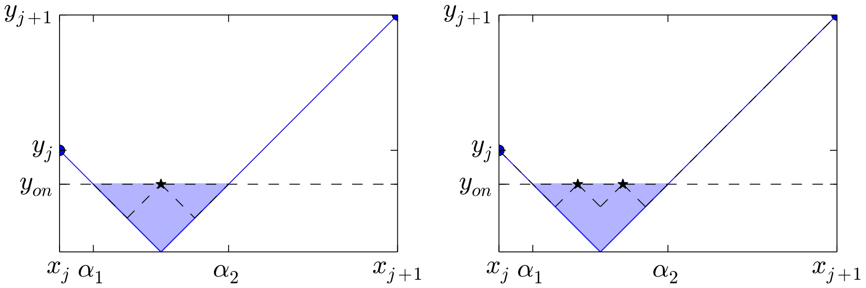

Figure 6.1: One-step worst-case optimal bisection (left) and trisection (right) of an interval $[x_j, x_{j+1}]$ in the single-objective Lipschitz optimization. The function can fall below y_{on} in the shaded area. The stars denote the worst-case function values at the points that minimize the resulting maximum tolerance.

As proved in [91], the worst-case optimal algorithm searching for the global minimum of Lipschitz continuous functions, given a predifined budget of objective function evaluations N_{max}, is equivalent to selecting N_{max} centers of balls of minimum radius to cover the feasible region. Similarly, a center of one ball is chosen in the one-step optimal algorithm. This can be seen considering the left side of Figure 6.1, when an interval with endpoints α_1 and α_2, defined by an intersection of the lower Lipschitz bound with a horizontal line at the level y_{on}, is regarded as the feasible region. The locations of the centers of three minimal-radius balls in the optimal cover of $[\alpha_1, \alpha_2]$ would be α_1, x_{n+1} and α_2.

Note that besides the considered subinterval $[x_j, x_{j+1}] \subset A$ containing the presumably single global minimizer of $F_n(x)$ there might be other subintervals of A, that include regions of possible improvement over y_{on}. However, if the next function evaluation location were positioned in such a subregion, the reduction of the worst-case error would not be guaranteed.

6.2.2 Trisection in the Single-Objective Case

Most of the computational effort in the algorithm considered in the previous section is devoted to updating the Lipschitz lower bounds and selecting a subinterval for partitioning. It might therefore be beneficial to place more function evaluations in the selected subinterval. The discussion that follows concentrates on the trisection of the selected interval. In addition to an interest in a stand-alone trisection-based algorithm, an optimal trisection of a search interval is important from the applicability to the diagonal Lipschitz optimization algorithms [55, 79–82] point of view.

Let us consider the minimization problem (6.1) and a branch-and-bound type algorithm, similar to that considered in the previous section. However, this time the selected search interval will be trisected. To simplify the formulas, it is assumed that $L = 1$.

Let the interval $[x_j, x_{j+1}]$ be given with known $f(x_j) = y_j$, $f(x_{j+1}) = y_{j+1} > y_j$. The following problem needs to be solved:

$$(\tilde{a}, \tilde{b}) = \arg \min_{a,b \in (x_j, x_{j+1})} \max_{f(\cdot) \in \Phi(L,n)} \Delta_{n+2}((\mathbf{X}_n, a, b), (\mathbf{Y}_n, f(a), f(b))), \tag{6.10}$$

here $\Delta_{n+2}(\cdot)$ is defined as in (6.6).

Let us first obtain an expression for $\Delta_{n+2}(\cdot)$. When two new function evaluations are made inside the interval (x_j, x_{j+1}), three intervals result, and hence three new local minima of the lower Lipschitz bound (6.3). If the minimizers are denoted by $t_i, i = 1, \ldots, 3$, the following relations hold:

$$\begin{aligned}
&f(x_j) - (t_1 - x_j) = f(a) - (a - t_1) &&\Longrightarrow\quad t_1 = \frac{y_j - f(a) + x_j + a}{2},\\
&f(a) - (t_2 - a) = f(b) - (b - t_2) &&\Longrightarrow\quad t_2 = \frac{f(a) - f(b) + a + b}{2},\\
&f(b) - (t_3 - b) = f(y_{j+1}) - (x_{j+1} - t_3) &&\Longrightarrow\quad t_3 = \frac{f(b) - y_{j+1} + b + x_{j+1}}{2}.
\end{aligned} \tag{6.11}$$

Correspondingly, the minima values at $t_i, i = 1, \ldots, 3$, are:

$$\begin{aligned}
M_1 &= y_j - (t_1 - x_j) = \frac{y_j + f(a) + x_j - a}{2},\\
M_2 &= f(a) - (t_2 - a) = \frac{f(a) + f(b) + a - b}{2},\\
M_3 &= f(b) - (t_3 - b) = \frac{f(b) + y_{j+1} + b - x_{j+1}}{2}.
\end{aligned} \tag{6.12}$$

Now the following expression for $\Delta_{n+2}(\cdot)$ results:

$$\Delta_{n+2}(\cdot) = \min(y_{on}, f(a), f(b)) - \\ \frac{1}{2}\min\left(y_j + f(a) + x_j - a, f(a) + f(b) + a - b, f(b) + y_{j+1} + b - x_{j+1}\right), \quad (6.13)$$

which is maximized when $f(a) = f(b) = y_{on}$, and becomes

$$\Delta^*_{n+2}((\mathbf{X}_n, a, b), (\mathbf{Y}_n, y_{on}, y_{on})) = \\ = \frac{1}{2}\max\left(y_{on} - y_j - x_j + a, -a + b, y_{on} - y_{j+1} - b + x_{j+1}\right). \quad (6.14)$$

To minimize the worst-case tolerance (6.14) and find the solution to (6.10), the following terms should be equal:

$$y_{on} - y_j - x_j + a = -a + b = y_{on} - y_{j+1} - b + x_{j+1} \implies \\ \tilde{a} = \alpha_1 + \frac{1}{3}(\alpha_2 - \alpha_1), \tilde{b} = \alpha_1 + \frac{2}{3}(\alpha_2 - \alpha_1), \\ \alpha_1 = x_j + (y_j - y_{on}), \alpha_2 = x_{j+1} - (y_{j+1} - y_{on}). \quad (6.15)$$

Thus it has been demonstrated that the worst-case optimal way to trisect an interval is to split the subinterval $[\alpha_1, \alpha_2]$, where improvement is still possible, into three equal parts (see the right side of Figure 6.1).

Similarly to the bisection case considered in Section 6.2.1, an optimal covering of $[\alpha_1, \alpha_2]$ by four balls/subintervals would consist of the balls/subintervals centered at the points α_1, $\tilde{a}$, $\tilde{b}$ and α_2. The radius of the optimal covering is equal to $1/6(\alpha_2 - \alpha_1)$.

6.3 One-Step Worst-Case Optimal Trisection for Bi-objective Univariate Optimization

A univariate bi-objective Lipschitz function $f(t) = (f_1(t), f_2(t)), t \in [x_j, x_{j+1}]$ is considered, where

$$|f_k(u) - f_k(t)| \leqslant L_k|u - t|, k = 1, 2, \tag{6.16}$$

for $\forall u, t \in [x_j, x_{j+1}]$, $L = (L_1, L_2)^T, L_k > 0, k = 1, 2.$

This section generalizes the concept of tolerance of the local Lipschitz lower bound for the Pareto front in the univariate bi-objective optimization. The tolerance was first introduced in [98]. The generalizations serve to establish the one-step worst-case optimal way of trisecting a subinterval and to enable a consistent implementation of the algorithm in Section 6.3.3.

This section contains three subsections. First, a set of definitions and lemmas are presented that lead to the tolerance with respect to which the optimality of trisection is defined. Second, the problem of establishing a one-step worst-case optimal trisection of an interval is formally stated along with a corresponding theorem formalizing the solution. The proofs of the lemmas and the theorem are given in Appendix C. The last subsection gives a pseudo-code of a trisection-based univariate bi-objective optimization algorithm.

6.3.1 Definitions

To begin with, the original concept of tolerance from [98] is presented (see Definition 6.3.1 of $\Delta^{nd}(\cdot)$), which applies to intervals $[x_j, x_{j+1}]$ with known, mutually non-dominating objective values at the endpoints. Further the definition is extended to the case where the Lipschitz constants of the objectives can differ from each other. In Definition 6.3.3 of $\Delta^{d}(\cdot)$ the tolerance is extended to cover the case where one of the endpoints dominates the other. It is shown that definitions 6.3.1 and 6.3.3 are actually coincident (Definition 6.3.5, $\Delta(\cdot)$). Finally, the notion of tolerance is generalized to an arbitrary subinterval $[r_1, r_2]$ of $[x_j, x_{j+1}]$, at the endpoints of which only the objective value ranges are known (Definition 6.3.6,

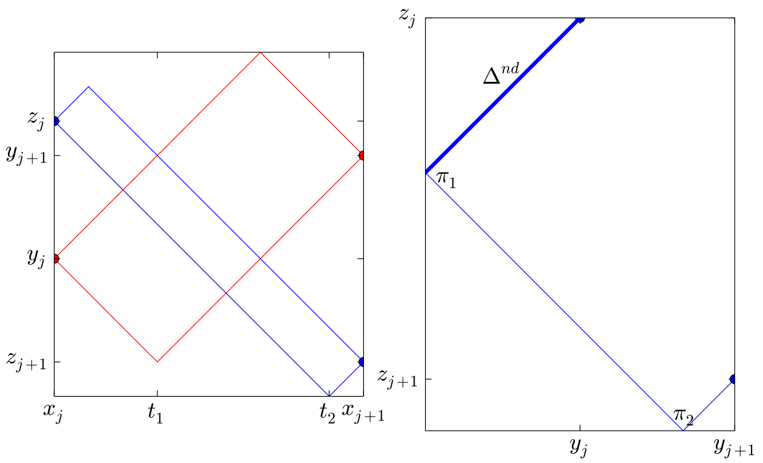

Figure 6.2: Illustration of the definition of $\Delta^{nd}(\cdot)$: $f(x_j) = (y_j, z_j)$ and $f(x_{j+1}) = (y_{j+1}, z_{j+1})$ do not dominate each other. **Left**: the lines show the Lipschitz bounds over the interval $[x_j, x_{j+1}]$. **Right**: $\Delta^{nd}(\cdot)$ is shown as the length of the highlighted line segment.

$\bar{\Delta}(\cdot)$). The scope of these definitions is limited to a single interval, therefore they are not to be confused with (6.6).

The following notation will be used:

$$f_1(t) = y_t, f_2(t) = z_t, f_1(x_i) = y_i, f_2(x_i) = z_i, i = j, j+1, \tag{6.17}$$

$$\delta y = |y_j - y_{j+1}|, \delta z = |z_j - z_{j+1}|, \tag{6.18}$$

$$C = \frac{1}{2}\sqrt{L_1^2 + L_2^2}, \tag{6.19}$$

$$\Psi = \{f(\cdot) : \text{function } f(\cdot) \text{ satisfies (6.16) and } f(x_i) = (y_i, z_i), i = j, j+1\}. \tag{6.20}$$

Let the values at the endpoints of the interval $[x_j, x_{j+1}]$ be known, i. e. $f(x_j) = (y_j, z_j)$, $f(x_{j+1}) = (y_{j+1}, z_{j+1})$.

Definition 6.3.1. When $f(x_j)$ and $f(x_{j+1})$ do not strongly dominate each other ($y_j \leqslant y_{j+1}, z_j \geqslant z_{j+1}$), the tolerance of the local Lipschitz lower bound for the actual Pareto front over $[x_j, x_{j+1}]$ is defined as

$$\begin{aligned}&\Delta^{nd}((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ &= \max(||\pi_1 - (y_j, z_j)^T||, ||\pi_2 - (y_{j+1}, z_{j+1})^T||),\end{aligned} \tag{6.21}$$

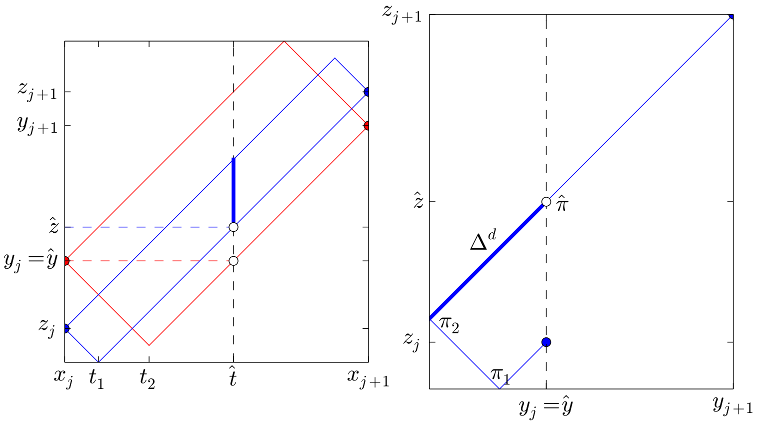

Figure 6.3: Illustration of the definition of $\Delta^d(\cdot)$: $f(x_j) = (y_j, z_j)$ dominates $f(x_{j+1}) = (y_{j+1}, z_{j+1})$. **Left**: the thin lines show the Lipschitz bounds over the interval $[x_j, x_{j+1}]$. The worst-case objective values at $\hat{t}$ are $y_{\hat{t}} = y_j$ and any value of z in the highlighted range; empty circles denote one of the possibilities. **Right**: $\Delta^d(\cdot)$ is shown as the length of the highlighted line segment. The empty circle corresponds to the example value of $f(\hat{t})$ on the left: $\hat{\pi} = f(\hat{t})$.

here $\pi_1 = (y_j - L_1(t_1 - x_j), z_j - L_2(t_1 - x_j))^T$, $\pi_2 = (y_{j+1} - L_1(x_{j+1} - t_2), z_{j+1} - L_2(x_{j+1} - t_2))^T$, $t_1 = \frac{x_j + x_{j+1}}{2} - \frac{\delta y}{2L_1}$, $t_2 = \frac{x_j + x_{j+1}}{2} + \frac{\delta z}{2L_2}$. In Figure 6.2, $\Delta^{nd}(\cdot)$ corresponds to the longer of the two line segments, connecting either $(y_j, z_j)^T$ and π_1, or $(y_{j+1}, z_{j+1})^T$ and π_2. □

Lemma 6.3.2.

$$\Delta^{nd}((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ = C \max\left((x_{j+1} - x_j) - \frac{\delta y}{L_1}, (x_{j+1} - x_j) - \frac{\delta z}{L_2}\right). \tag{6.22}$$

The case where $f(x_j)$ strongly dominates $f(x_{j+1})$ ($y_j < y_{j+1}, z_j < z_{j+1}$) is shown in Figure 6.3. Indeed, $f(t)$ is strongly dominated by $f(x_j)$ for all $t \in (\hat{t}, x_{j+1}]$, $\hat{t} = x_{j+1} - \min\left(\frac{\delta y}{L_1}, \frac{\delta z}{L_2}\right)$. In the figure, the condition

$$\frac{\delta y}{L_1} \leqslant \frac{\delta z}{L_2} \tag{6.23}$$

holds. Thus the following equalities hold: $\hat{t} = x_{j+1} - \frac{\delta y}{L_1}$, $t_1 = \frac{x_j + x_{j+1}}{2} - \frac{\delta z}{2L_2}$, $t_2 = \frac{x_j + x_{j+1}}{2} - \frac{\delta y}{2L_1}$. The dominated subinterval $(\hat{t}, x_{j+1}]$ should be excluded from

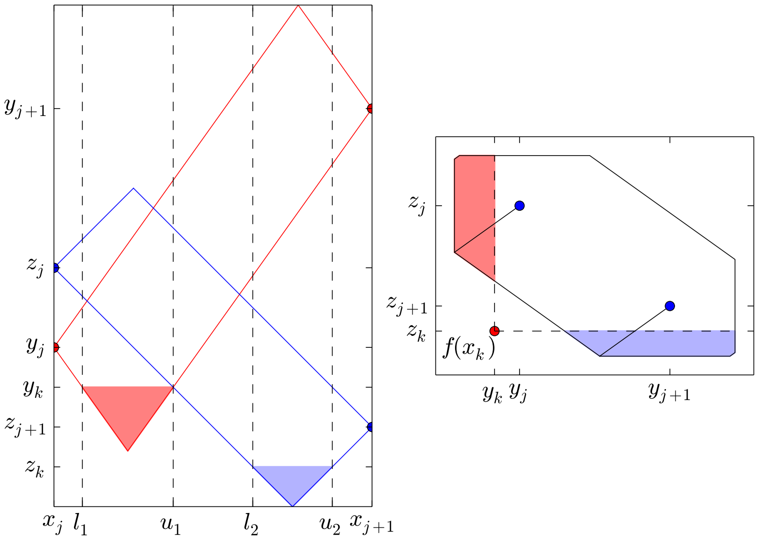

Figure 6.4: Subintervals generated from $[x_j, x_{j+1}]$ when a dominating $f(x_k)$ is known for some trial point $x_k \neq x_j, x_{j+1}$. Here $L_1 = 1.4, L_2 = 1$. On the right, the range of the objectives over the interval $[x_j, x_{j+1}]$ is displayed as a closed octagon. A point $f(x_k) = (y_k, z_k)$ dominates a part of it. The shaded regions denote the part where non-dominated solutions are still possible: $\{(y, z) : y \leqslant y_k \text{ or } z \leqslant z_k\}$. On the left, the non-dominated values are still possible in the subintervals $S_1 = [l_1, u_1]$ and $S_2 = [l_2, u_2]$. Note that for endpoints l_1, u_1, l_2, u_2, only the objective value ranges are known.

the further optimization process. Since the possible values at the endpoints of the remaining subinterval $[x_j, \hat{t}]$ do not strongly dominate each other, the tolerance for $[x_j, x_{j+1}]$ is defined as equal to that of $[x_j, \hat{t}]$ based on the Definition 6.3.1 as the largest possible with respect to the objective value ranges at $\hat{t}$.

This is formalized in the following definition.

Definition 6.3.3. When $f(x_j)$ strongly dominates $f(x_{j+1})$ $(y_j < y_{j+1}, z_j < z_{j+1})$ and $\hat{t} = x_{j+1} - \min\left(\frac{\delta y}{L_1}, \frac{\delta z}{L_2}\right)$, then the tolerance for the interval $[x_j, x_{j+1}]$ is defined as

$$\Delta^d((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \max_{f(\cdot)\in\Psi} \Delta^{nd}((x_j, \hat{t}), (f(x_j), f(\hat{t}))). \tag{6.24}$$

□

Lemma 6.3.4.

$$\begin{aligned}&\Delta^d((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ &= C \max\left((x_{j+1} - x_j) - \frac{\delta y}{L_1}, (x_{j+1} - x_j) - \frac{\delta z}{L_2}\right).\end{aligned} \tag{6.25}$$

Definitions 6.3.1 and 6.3.3 cover the cases where function values are known at both endpoints of an interval. Irrespective of the mutual domination of these endpoint values, the tolerance is computed according to the identical formula ((6.22) and (6.25)). The tolerance depends on the length of the interval and the smaller of the absolute differences of objective values at the interval endpoints. Both cases are aggregated into one as follows.

Definition 6.3.5. For any interval $[x_j, x_{j+1}]$ with known $f(x_j), f(x_{j+1})$

$$\begin{aligned}&\Delta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ &= C \max\left((x_{j+1} - x_j) - \frac{\delta y}{L_1}, (x_{j+1} - x_j) - \frac{\delta z}{L_2}\right).\end{aligned} \tag{6.26}$$

□

During the operation of the algorithm, described in Section 6.3.3, the subintervals, where only dominated function values are possible, are excluded from further processing. Therefore it is possible to generate a subinterval $[r_1, r_2]$, the endpoints of which satisfy $x_j \leqslant r_1 \leqslant r_2 \leqslant x_{j+1}$, where only $f(x_j), f(x_{j+1})$ are known. Thus at r_1 and r_2 only the objective value ranges are known (e. g. see subintervals $[l_1, u_1]$ and $[l_2, u_2]$ in Figure 6.4).

Definition 6.3.6. For an interval $[r_1, r_2], r_1 \leqslant r_2$, such that $[r_1, r_2] \subseteq [x_j, x_{j+1}]$, and known $f(x_j), f(x_{j+1})$, the tolerance is defined as:

$$\bar{\Delta}((r_1, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \max_{f(\cdot)\in\Psi} \Delta((r_1, r_2), (f(r_1), f(r_2))). \tag{6.27}$$

□

Lemma 6.3.7. *Denote*

$$\beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = (x_{j+1} - x_j) - \min\left(\frac{\delta y}{L_1}, \frac{\delta z}{L_2}\right). \tag{6.28}$$

Then

$$\bar{\Delta}((r_1, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) =$$
$$= C \times \begin{cases} r_2 - r_1, & \textit{if } r_2 - r_1 \leqslant \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \\ \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), & \textit{otherwise.} \end{cases} \tag{6.29}$$

Definition 6.3.6 is the most general and reduces to Definition 6.3.5 when $r_1 = x_j$, $r_2 = x_{j+1}$. Moreover, this is the tolerance used in the definition of the optimality of trisecting an interval, as described in the following subsection.

6.3.2 The One-Step Worst-Case Optimal Trisection Problem

The trisection of the interval $[r_1, r_2]$, $x_j \leqslant r_1 < r_2 \leqslant x_{j+1}$, using points $\tilde{a}, \tilde{b} \in (r_1, r_2)$, $\tilde{a} < \tilde{b}$, is of interest. The worst-case optimality criterion for the choice of the points is defined as follows:

$$(\tilde{a}, \tilde{b}) = \arg\min_{a,b} \bar{\Delta}^*((a, b, r_1, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \tag{6.30}$$

where

$$\bar{\Delta}^*(\cdot) = \max_{f(\cdot)\in\Psi} \max \{ \bar{\Delta}((r_1, a, x_j, b), (f(x_j), f(b))), \bar{\Delta}((a, b, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \bar{\Delta}((b, r_2, a, x_{j+1}), (f(a), f(x_{j+1}))) \}. \tag{6.31}$$

Thus, assuming the most unfavorable function values at the division points, $\tilde{a}$ and $\tilde{b}$ minimize the maximum of the resulting tolerances of the three subintervals.

The following lemma serves for proving the main theorem of this chapter.

Lemma 6.3.8. *Let $x_j \leqslant t_1 < u < v < t_2 \leqslant x_{j+1}$ and $f(x_j), f(x_{j+1})$ be known. Then*

$$\max_{f(\cdot)\in\Psi} \bar{\Delta}((u, v, t_1, t_2), (f(t_1), f(t_2))) = \bar{\Delta}((u, v, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))). \quad (6.32)$$

Theorem 6.3.9. *Let $x_j \leqslant r_1 < r_2 \leqslant x_{j+1}$ and $\beta = \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1})))$ be defined by (6.28). The worst-case optimal division points $\tilde{a}$ and $\tilde{b}$ of (r_1, r_2), solving (6.30), are defined as follows:*

1) $\tilde{a} = r_1 + \frac{1}{3}(r_2 - r_1), \tilde{b} = r_1 + \frac{2}{3}(r_2 - r_1)$, if $\frac{1}{3}(r_2 - r_1) < \beta$;

2) an arbitrary choice $\tilde{a} \in (r_1, r_2), \tilde{b} \in (r_1, r_2)$, otherwise.

Then $\bar{\Delta}^*(\cdot)$ equals $\frac{C}{3}(r_2 - r_1)$ in the first case, and $C\beta$ in the second.

Theorem 6.3.9 states that the one-step worst-case optimal division points $\tilde{a}$ and $\tilde{b}$ for an arbitrary interval of the feasible region divide it into three equal parts. Note that for very short intervals, all possible configurations of division points are equivalent, thus division into three equal parts is always optimal.

6.3.3 The Trisection Algorithm

The pseudo-code of an algorithm, based on the discussion in Section 6.3, is presented in Algorithm 4 as procedure *Trisection*. The functions $f_1(x)$ and $f_2(x)$ to be minimized are defined over the interval $[a, b]$. The estimates of the Lipschitz constants $L_k, k = 1, 2,$ are provided as input parameters. Initially, the first two function evaluations at the interval endpoints are made, and the set of generated intervals I is initialized to $\{[a, b]\}$.

Algorithm 4 The pseudo-code of the one-step worst-case optimal trisection algorithm.

```
procedure TRISECTION($f_1$, $f_2$, $a$, $b$, $N_{max}$, $\epsilon$, $L_1$, $L_2$)
    $n \leftarrow 0, I \leftarrow \{[a,b]\}$
    $x_1 \leftarrow a$, $x_2 \leftarrow b$, compute $f_k(x_i), i = 1,2, k = 1,2$
    SPLIT($a$, $b$, $x_i$, $I$), $i = 1,2$
    $n \leftarrow 2$
    while $n < N_{max}$ and ($\epsilon$ is undefined or condition (6.33) does not hold) do
        Select the interval $\bar{R} = [\bar{r_1}, \bar{r_2}]$, which has the highest $\bar{\Delta}$ in the current interval set $I$
        If an interval with $\bar{\Delta} < 0$ is encountered, print a warning that either $L_1$ or $L_2$ is too small
        $x_{n+1} \leftarrow \bar{r}_1 + \frac{1}{3}(\bar{r}_2 - \bar{r}_1)$, $x_{n+2} \leftarrow \bar{r}_1 + \frac{2}{3}(\bar{r}_2 - \bar{r}_1)$, compute $f(x_i)$, $i = n+1, n+2$
        $I \leftarrow (I \backslash \bar{R}) \cup \{[\bar{r}_1, x_{n+1}], [x_{n+1}, x_{n+2}], [x_{n+2}, \bar{r}_2]\}$
        SPLIT($\bar{r}_1$, $\bar{r}_2$, $x_i$, $I$), $i = 1, \ldots, n$
        SPLIT($a$, $b$, $x_i$, $I$), $i = n+1, n+2$
        $n \leftarrow n + 2$
    end while
    return $I$, $x_i$, $f(x_i)$, $i = 1, \ldots, N_{max}$
    procedure SPLIT($u$, $v$, $x_k$, $I$)
        $J \leftarrow \varnothing$
        for all $R \in I : R \subseteq [u,v]$ do
            $I \leftarrow I \backslash R$
            Find $j \in \{1, \ldots, n+2\}$ such that $R \subseteq [x_{o_j}, x_{o_{j+1}}]$, where $x_{o_i}, i = 1, \ldots, n+2$, are ordered trial points
            Compute $S_i(x_{o_j}, x_{o_{j+1}}, x_k), i = 1,2$
            $R_1 \leftarrow R \cap S_1$, $R_2 \leftarrow R \cap S_2$
            if $R_1 \cap R_2 = \varnothing$ then
                if $R_1 \neq \varnothing$ then $J \leftarrow J \cup R_1$
                end if
                if $R_2 \neq \varnothing$ then $J \leftarrow J \cup R_2$
                end if
            else $J \leftarrow J \cup (R_1 \cup R_2)$
            end if
        end for
        $I \leftarrow J$
    end procedure
end procedure
```

The algorithm operates iteratively until either the maximum number of function evaluations N_{max} is reached or the maximum tolerance, computed according to (6.29), of the generated intervals falls below the chosen threshold $\epsilon > 0$, i. e.

$$\max_{[r_1,r_2]\in I} \bar{\Delta}((r_1, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) \leqslant \epsilon. \tag{6.33}$$

At each iteration tolerances $\bar{\Delta}(\cdot)$ are computed for every interval $R = [r_1, r_2] \in I$, according to (6.29). The interval with the largest $\bar{\Delta}(\cdot)$ value, denoted $\bar{R} = [\bar{r}_1, \bar{r}_2]$, is divided into three equal parts by the new trial points x_{n+1} and x_{n+2}, considering Theorem 6.3.9. $\bar{R}$ is replaced by the three resulting subintervals in I. If for some interval a negative $\bar{\Delta}(\cdot)$ has been determined, a warning is printed to inform the user that one of the provided Lipschitz constants $L_k, k = 1, 2$, is too small. The operation of the algorithm can continue, effectively excluding this interval from further search, since (6.33) implies that only intervals with $\bar{\Delta}(\cdot) > \epsilon > 0$ can be selected for partitioning. However, the user might prefer to run the algorithm providing a larger Lipschitz constant.

After the new trials have been performed, the restructuring of I follows to exclude the dominated subintervals. With the indexing of the trial points $x_i, i = 1, \ldots, n+2$, in an ascending order, i. e. $x_{o_j} \leqslant x_{o_l}$, if $j \leqslant l$, given some trial $f(x_k)$, a subinterval of $[x_{o_j}, x_{o_{j+1}}]$ can be computed, where values of objective $i = 1, 2$, non-dominated by $f(x_k)$, are still possible. It will be denoted by $S_i(x_{o_j}, x_{o_{j+1}}, x_k) = [l_i, u_i] \subseteq [x_{o_j}, x_{o_{j+1}}]$ and computed as

$$\begin{aligned} l_i &= \begin{cases} \frac{1}{L}(f_i(x_{o_j}) + L_i x_{o_j} - f_i(x_k)), & \text{if } f_i(x_k) < f_i(x_{o_j}), \\ x_{o_j}, & \text{otherwise,} \end{cases} \\ u_i &= \begin{cases} -\frac{1}{L}(f_i(x_{o_{j+1}}) - L_i x_{o_{j+1}} - f_i(x_k)), & \text{if } f_i(x_k) < f_i(x_{o_{j+1}}), \\ x_{o_{j+1}}, & \text{otherwise.} \end{cases} \end{aligned} \tag{6.34}$$

This situation is illustrated in Figure 6.4, where, for brevity, $x_{o_j} = x_j$, $x_{o_{j+1}} = x_{j+1}$.

The procedure *Split* uses (6.34) to update all intervals $R \in I : R \subseteq [u, v]$ with respect to a trial point x_k. Given the ordered trial points $x_{o_i}, i = 1, \ldots, n+2$, the index $j \in \{1, \ldots, n+2\}$ is found such that $R \subseteq [x_{o_j}, x_{o_{j+1}}]$. The non-dominated part of R, consisting of intersections $R \cap S_i(x_{o_j}, x_{o_{j+1}}, x_k), i = 1, 2$, replaces R

in I. Thus Step 4 excludes subintervals of $[a, b]$, dominated by either x_1 or x_2. Step 11 updates the subintervals of $\bar{R}$ in case of domination by any of the previous points $x_i, i = 1, \ldots, n$. Finally, Step 12 updates all subintervals, if they are dominated by the new trials at x_{n+1} and x_{n+2}.

The algorithm returns the performed trials $x_i, f(x_i), i = 1, \ldots, N_{max}$, and the set of non-dominated intervals I.

6.4 Numerical Experiments

This section illustrates the performance of the proposed *Trisection* algorithm. In order to have comparable results, an experimental methodology analogous to that of the original study [98] on the univariate bi-objective optimization using the concept of the Lipschitz local lower bound tolerance is followed.

Table 6.1: The test problems used.

Test problem	**Objectives**	**Domain**	**Lipschitz constants**	**Pareto set**
Rastr [98]	$f_1(x) = (x - 0.5)^2 - \cos(18(x - 0.5))$ $f_2(x) = (x + 0.5)^2 - \cos(18(x + 0.5))$	$[-1, 1]$	$L_1 = 21$ $L_2 = 21$	$[-0.527622, -0.5] \cup$ $[0.5 - \frac{2\pi}{9}, -0.5 + \frac{\pi}{9}] \cup$ $[0.5 - \frac{\pi}{9}, -0.5 + \frac{2\pi}{9}] \cup$ $[0.5, 0.527622]$
Fo & Fle [23], [22, pp. 339-340]	$f_1(x) = 1 - \exp(-(x - 1)^2)$ $f_2(x) = 1 - \exp(-(x + 1)^2)$	$[-4, 4]$	$L_1 = 1$ $L_2 = 1$	$[-1, 1]$
Schaf [22, pp. 339-340]	$f_1(x) = \begin{cases} -x, & \text{if } x \leqslant 1, \\ x - 2, & \text{if } 1 < x \leqslant 3, \\ 4 - x, & \text{if } 3 < x \leqslant 4, \\ x - 4, & \text{if } x > 4, \end{cases}$ $f_2(x) = (x - 5)^2$	$[-1, 8]$	$L_1 = 1$ $L_2 = 12$	$[1, 2] \cup [4, 5]$

The implementation of Algorithm 4 is referred to as *Trisection* in the tables. For comparison, the analogous algorithm from [98], is included. The main difference of [98] from the *Trisection* algorithm is the partitioning of the selected subinterval, which in [98] is bisected into two equal parts; the name *Bisection* is used for this algorithm throughout the section. The two algorithms are easy to compare as they target the same type of optimization problems, moreover, the same stopping criterion (6.33) is applicable to both of them. All the results for algorithm *Bisection* are taken from the original study [98].

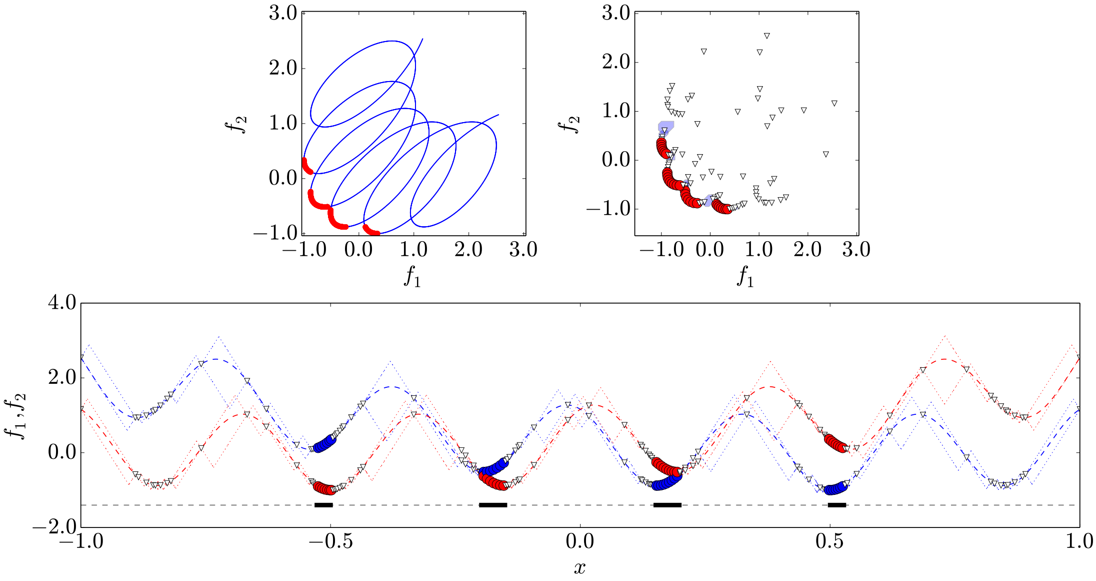

Figure 6.5: The first experiment results of the *Trisection* algorithm for the problem *Rastr*. **Top left:** the range of the objectives and the actual Pareto front. **Top right:** the objective function values obtained by the algorithm *Trisection*. The non-dominated and dominated solutions found are marked by the filled points and the triangles, respectively. The shaded areas denote the objective ranges over subintervals where non-dominated objective values are still possible. **Bottom:** the same trials are displayed over the graphs of both objectives (dashed lines). The actual Pareto set is shown as the thick line segments beneath the graphs. The dotted lines denote the Lipschitz bounds over the unexplored subintervals.

Three univariate bi-objective optimization problems have been used (see Table 6.1): *Rastr*, *Fo* & *Fle* and *Schaf*. The first problem consists of two modified Rastrigin functions and is common in testing single-objective optimization algorithms. Problem *Fo* & *Fle* consists of two functions that are hard to optimize separately, as they have large flat regions. The last problem poses a challenge due to the discontinuous Pareto front and considerably different Lipschitz constants, as well as function value ranges. The Figures 6.5-6.7 show the graphs of the objectives and Pareto set of each problem (bottom), as well as function ranges and the Pareto front (top left).

In the first experiment algorithms *Trisection* and *Bisection* were allowed to run until the maximum tolerance of all the generated intervals fell below a certain

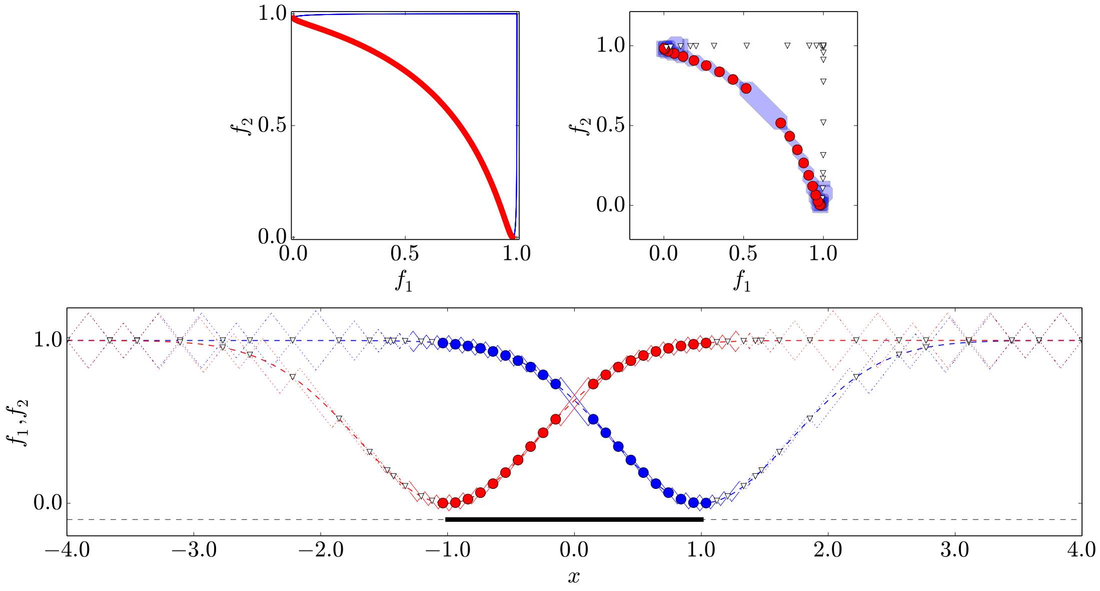

Figure 6.6: The first experiment results of the *Trisection* algorithm for the problem *Fo & Fle*. The figure structure is analogous to that of Figure 6.5.

Table 6.2: Comparison of the algorithms when the maximum tolerance of the intervals falls below $\epsilon = 0.1$.

Problem	*Rastr*		*Fo & Fle*		*Schaf*	
Algorithm	*Trisection*	*Bisection*	*Trisection*	*Bisection*	*Trisection*	*Bisection*
n	106	90	48	36	192	270
nd	30	29	20	16	124	186

threshold ϵ (condition (6.33)). The value $\epsilon = 0.1$ was used for all testing problems. Table 6.2 shows the number n of function evaluations required (one evaluation computes both objectives) and the number nd of non-dominated solutions generated. A solution is considered non-dominated if there is no other solution that weakly dominates it.

To reach the required tolerance for the test problems, the *Trisection* algorithm had to perform 106, 48 and 192 function evaluations and produced, respectively, 30, 20 and 124 non-dominated solutions. The *Bisection* algorithm required 90, 36 and 270 function evaluations to produce 29, 16 and 186 non-dominated solutions. The solutions generated by the *Trisection* algorithm are illustrated at the top

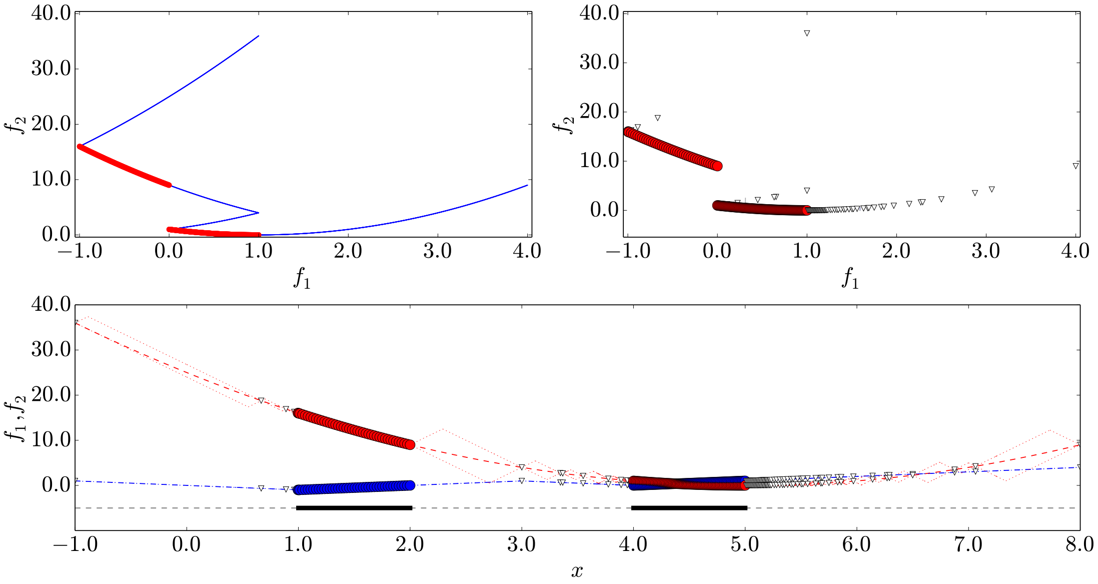

Figure 6.7: The first experiment results of the *Trisection* algorithm for the problem *Schaf*. The figure structure is analogous to that of Figure 6.5.

Table 6.3: Comparison of the optimization algorithms, using a fixed number of function evaluations $N_{max} = 100$.

Problem	*Rastr*		*Fo & Fle*		*Schaf*	
Algorithm	*Trisection*	*Bisection*	*Trisection*	*Bisection*	*Trisection*	*Bisection*
nd	30	34	58	66	51	54
np	28	26	56	65	50	51
Error Rate	0.067	0.235	0.034	0.015	0.02	0.056
E	0.022	0.014	0	0	0.1	0.01
H	0.057	0.064	0.051	0.027	0.293	0.254

right of Figures 6.5-6.7.

In the second experiment, the same algorithms were compared according to several metrics, but this time they were stopped after a fixed number ($N_{max} = 100$) of function evaluations. The number of non-dominated solutions found is denoted by nd, and the number of the true Pareto-optimal solutions among them is np. The formula $(nd - np)/nd$ produces the well-known *Error Rate* metric. Denoting the actual Pareto front in the objective space by P and by $\tilde{P}$ the set of non-dominated solutions found, the absolute error is defined:

$$E = \max_{\tilde{Y} \in \tilde{P}} \min_{Y \in P} \|\tilde{Y} - Y\|. \tag{6.35}$$

This value describes the proximity of the solutions found to the actual Pareto front. The smaller it is, the better the performance of the algorithm. The final metric

$$H = \max_{Y \in P} \min_{\bar{Y} \in \bar{P}} \|\bar{Y} - Y\|, \tag{6.36}$$

where $\bar{P}$ is the set of solutions found that belong to the actual Pareto front, serves to describe the extent of a part of the Pareto front, not represented by the solutions found. Similarly to (6.35), a better performance results in a smaller value. Since an exact solution of optimization problems (6.35) and (6.36) is too complex, an approximation was sought using the discretization of P by 10^6 uniformly distributed points over the Pareto set in the feasible region. The results of the experiment are presented in Table 6.3. The discretization-related error is smaller than 10^{-4}, therefore 3 digits after the decimal point are displayed, also, entries smaller than 10^{-4} are rounded to 0.

As could be expected, the results of the experiments illustrate a slightly worse performance of the trisection-based algorithm as compared to that of the bisection-based one, although the difference is not pronounced. However, the trisection of the interval results in more function evaluations at a single iteration of the algorithm, and the extension of a univariate algorithm to the higher-dimensional case will supposedly show the advantages of trisection, as indicated in [115].

Table 6.4: The sensitivity of the *Trisection* algorithm to the accuracy of estimation of the Lipschitz constants, when the maximum tolerance of the intervals falls below $\epsilon = 0.1$.

Problem	*Rastr*				*Fo & Fle*				*Schaf*			
L×	1/1.2	1	1.2	1.4	1/1.2	1	1.2	1.4	1/1.2	1	1.2	1.4
n	84	106	116	166	38	48	48	50	174	192	246	324
nd	27	30	33	60	14	20	20	22	113	124	159	230
np	26	28	32	59	12	18	18	20	112	123	159	230
Error Rate	0.037	0.067	0.03	0.017	0.143	0.1	0.1	0.091	0.009	0.008	0	0
$\max \bar{\Delta}(\cdot)$	0.097	0.098	0.096	0.097	0.092	0.068	0.098	0.096	0.097	0.092	0.099	0.01
$\sum \bar{\Delta}(\cdot)$	2.688	2.937	3.82	4.962	1.299	1.252	1.647	2.180	7.244	9.651	12.331	15.133

The last experiment is intended to analyze the sensitivity of the *Trisection* algorithm to the accuracy of the Lipschitz constant estimates used. Denoting by $L = (L_1, L_2)^T$ the actual Lipschitz constants for the problem, the following estimates are used: $\tilde{L} = L/1.2, L, 1.2L, 1.4L$. For each problem the algorithm is stopped, when condition (6.33) is satisfied with $\epsilon = 0.1$. The results are shown in Table 6.4. Here n denotes the number of function evaluations performed, nd, np and *Error Rate* have the same meaning as in Table 6.3. The maximum and total tolerance of the intervals immediately after the condition (6.33) has been satisfied is given on rows $\max \bar{\Delta}(\cdot)$ and $\sum \bar{\Delta}(\cdot)$, respectively.

It can be seen from Table 6.4 that the number of required function evaluations grows with the multiplier of the actual Lipschitz constants, as does the total tolerance of the intervals. This can be explained by the tolerance (6.29) being proportional to constant C (6.19). As a result, when higher Lipschitz constants are used, the intervals will have to become shorter for the maximum tolerance to drop below the level ϵ. The numbers of non-dominated and actual Pareto solutions increase with the multiplier as well. nd and np differ by 1 or 2 most of the time and the difference does not seem highly influenced by $\tilde{L}$. The maximum tolerance does not seem to be influenced, either. During the experiment no warnings regarding the use of too small Lipschitz constants have been printed, meaning that the variation of the objectives contradicting the Lipschitz condition with the given estimates of L has not been observed. So it can be concluded that the equal scaling of both Lipschitz constants does not affect the performance for the worse, as the increased number of function evaluations is balanced by the increased number of solutions in the actual Pareto front.

6.5 Chapter Summary and Conclusions

1. The problem of one-step worst-case optimal trisection of an arbitrary subinterval of the feasible region is defined with respect to the extended tolerance definition.

2. The trisection of a subinterval into three equal parts is proved to satisfy the optimality conditions.

3. The presented analysis enables to exclude the dominated regions from further search and thus efficiently distribute the trial points over the search region. A corresponding optimization algorithm is implemented.

4. The experimental investigation reveals a slight advantage of the bisection-based algorithm over the trisection-based one.

5. The presented optimal univariate algorithm for bi-objective problems supports the analogous trisection-based partitioning scheme used in bivariate bi-objective optimization.

6. The generalization of trisection to the higher-dimensional case could be advantageous in the case of diagonal algorithms.

Results and Conclusions

In this thesis guidelines for the choice of a statistical objective function model among the Gaussian stochastic functions constructing global optimization algorithms were formulated based on a proposed experimental methodology. It was suggested to base the choice of the stochastic function on a priori information about the objective function complexity that is either relatively high, low, or unknown.

A recent global optimization algorithm relying on a hyper-rectangular decomposition - adjusted statistical objective function model was implemented in several ways and experimentally compared to other similar contemporary algorithms. Two extensions of this algorithm, balancing the local and global search strategies, were proposed. First, algorithm *GB* operates by switching between explicitly defined global and local search phases, based on improvement achieved in the current phase. Second, algorithm *Cluster* employs a clustering procedure to identify the well-explored regions.

The asymptotic equivalence of two criteria, used in global optimization algorithms based on simplicial decomposition-adjusted statistical models of the objective function, was proved.

The definition of the one-step worst-case optimal interval trisection problem in the univariate bi-objective Lipschitz optimization was presented. The optimality was defined using the concept of tolerance of the local Lipschitz lower bound for the actual Pareto front over a subinterval of the feasible region. The trisection of an interval into three equal parts was proved to satisfy the optimality conditions, and a corresponding optimal algorithm was implemented.

The research reported in this thesis leads to the following conclusions:

1. The experimental results show that the *P-algorithm*, constructed assuming a stationary isotropic Gaussian stochastic function with an exponential correlation, performs the best for a variety of univariate and bivariate objective functions, and is therefore advised when no a priori information about the objective function complexity is available. Simple objective functions can successfully be optimized by the *Maximum expected improvement algorithm*, constructed using one of the stationary isotropic Gaussian stochastic functions with low short-range variability.

2. Two heuristic global search acceleration techniques were proposed, successfully preventing a recent global optimization algorithm *Rect-1* from excessively exploring the vicinity of the suboptimal local minimizers. Proposed extensions *GB* and *Cluster* consume considerably fewer function evaluations than the original algorithm for difficult multi-modal global optimization problems. For such problems *Cluster* worst-case performance is the best among the considered algorithms.

3. Two simplex selection criteria in global optimization with simplicial decomposition-adjusted statistical models are related by an asymptotic equivalence relation, as the size of a simplex decreases. The first criterion is defined by a simple expression, based on heuristic reasoning. The second criterion relates to the improvement probability at the current optimization step, but its expression is complex. The simple expression is naturally preferred over a complicated one, however, it lacked a theoretical justification. The demonstrated equivalence relation provides the required theoretical support for the simple expression in the bi-variate case. Moreover, a simplified expression approximating the second criterion in higher dimensions was obtained, that might be used in new optimization algorithms.

4. The trisection of an interval into three equal parts in the univariate bi-objective Lipschitz optimization is one-step worst-case optimal. The tolerance of the local Lipschitz lower bound for the actual Pareto front over an interval is used to define the optimality in question. This optimal univariate partitioning scheme supports an analogous trisection-based partitioning scheme used in bi-variate bi-objective optimization. The analysis presented enables to exclude the dominated regions from further search and thus efficiently distribute trial points over the search region.

The generalization of trisection to the higher-dimensional case could be advantageous in the case of diagonal algorithms.

Appendix A

Expressions of Conditional Characteristics of Stochastic Functions

This appendix presents the expressions of conditional mean $m(x|\xi(x_i) = y_i, i = 1, \dots, n)$ and conditional variance $s^2(x|\xi(x_i) = y_i, i = 1, \dots, n)$ for univariate Markovian Gaussian stochastic processes. The conditional characteristics of the random process are computed with respect to the already performed ordered trials $\xi(x_i^n) = y_i^n$, $i = 1, \dots, n$, where $x_i^n < x_{i+1}^n, i = 1, \dots, n-1$.

The *Wiener* process possesses the Markov property, therefore for $x \in [x_j^n, x_{j+1}^n]$, $j = 1, \dots, n-1$,

$$\begin{aligned} & m(x|\xi(x_i^n) = y_i, i = 1, \dots, n) = m(x|\xi(x_i^n) = y_i, i = j, j+1) = \\ & = \frac{y_j(x_{j+1}^n - x) + y_{j+1}(x - x_j^n)}{x_{j+1}^n - x_j^n}, \end{aligned} \tag{A.1}$$

$$\begin{aligned} & s^2(x|\xi(x_i^n) = y_i, i = 1, \dots, n) = s^2(x|\xi(x_i^n) = y_i, i = j, j+1) = \\ & = \sigma^2 \frac{(x_{j+1}^n - x)(x - x_j^n)}{x_{j+1}^n - x_j^n}, \end{aligned} \tag{A.2}$$

where σ is the *Wiener* process standard deviation parameter.

A special 1-dimensional case of a stationary Gaussian random process with mean μ, standard deviation σ and an *Exponential* correlation function $\rho(\tau) = \exp(-\tau/c), \tau \geqslant 0$, possesses the Markov property, therefore the conditional characteristics at $x \in [x_j^n, x_{j+1}^n]$, $j = 1, \ldots, n-1$, can be computed according to explicit expressions:

$$
\begin{aligned}
&m(x|\xi(x_i^n) = y_i, i = 1, \ldots, n) = m(x|\xi(x_i^n) = y_i, i = j, j+1) = \\
&= \mu + (y_j - \mu, y_{j+1} - \mu)(\mathbf{v}M)^T, \qquad \text{(A.3)} \\
&s^2(x|\xi(x_i^n) = y_i, i = 1, \ldots, n) = s(x|\xi(x_i^n) = y_i, i = j, j+1) = \\
&= \sigma^2 \left(1 - \left(\exp\left(-\frac{x - x_j^n}{c}\right), \exp\left(-\frac{x_{j+1}^n - x}{c}\right)\right) (\mathbf{v}M)^T\right), \qquad \text{(A.4)} \\
&\mathbf{v} = \left(\exp\left(-\frac{x - x_j^n}{c}\right), \exp\left(-\frac{x_{j+1}^n - x}{c}\right)\right)^T, \qquad \text{(A.5)} \\
&M = \frac{1}{1 - \exp(-\frac{2(x_{j+1}^n - x_j^n)}{c})} \begin{pmatrix} 1 & -\exp(-\frac{x_{j+1}^n - x_j}{c}) \\ -\exp(-\frac{x_{j+1}^n - x_j^n}{c}) & 1 \end{pmatrix}. \qquad \text{(A.6)}
\end{aligned}
$$

Appendix B

Statistical Model Parameter Estimation Statistics

Suppose that a global optimization algorithm is defined using an *assumed model* of the objective function, that is characterized by the parameters Θ. Further, suppose that the objective function is generated according to the *actual model*, that is either the same as the *assumed model*, or a different one.

The maximum likelihood estimates $\hat{\theta}_i, i = 1, \ldots, 1000$, of $\theta \in \Theta$ were obtained from the trials of the 1000 realizations of the *actual model*, as described in Section 3.5. Tables B.1 and B.2 provide the statistics of the obtained estimates. In particular, for each combination of the *assumed* and *actual* models, two values per parameter $\theta \in \Theta$ are given, denoted, respectively, by $E[\theta]$ and $\sqrt{V}[\theta]$:

1. $E[\theta]$ denotes the mean of the estimates:

$$\frac{1}{1000} \sum_{i=1}^{1000} \hat{\theta}_i; \tag{B.1}$$

2. $\sqrt{V}[\theta]$ denotes the standard deviation of the estimates:

$$\frac{1}{1001} \sum_{i=1}^{1000} (\hat{\theta}_i - E[\theta])^2. \tag{B.2}$$

Table B.1: Parameter estimation statistics for 1-dimensional models. The values at $k_i = i/M$, $i = 0, \ldots, M$, $M = 10$, were used to produce the estimates.

	Actual model	**Assumed model**		
		Wiener	*Exponential*	*Gaussian*
$E[\sigma]$	*Wiener*	0.93511	3.66874	3.84023
$\sqrt{V}[\sigma]$		0.20670	0.99573	0.97946
$E[\mu]$	*Exponential*	0.00604	0.00906	-0.00504
$\sqrt{V}[\mu]$		0.43485	0.36885	0.34311
$E[\sigma]$		0.43340	0.90704	0.92986
$\sqrt{V}[\sigma]$		0.20965	0.21994	0.21651
$E[c]$		0.39734	0.06359	0.05595
$\sqrt{V}[c]$		0.41824	0.05290	0.04210
$E[\mu]$	*Gaussian*	-0.31536	1.25238	-1.28916
$\sqrt{V}[\mu]$		18.84934	60.08774	35.85975
$E[\sigma]$		1.88043	3.79245	2.02258
$\sqrt{V}[\sigma]$		16.95115	50.28182	30.44435
$E[c]$		0.14726	0.07753	0.06853
$\sqrt{V}[c]$		0.22721	0.16468	0.11659

Table B.2: Parameter estimation statistics for 2-dimensional models. The values at $\mathbf{k}_{ij} = (i/M, j/M)$, $i, j = 0, \ldots, M$, $M = 4$, were used to produce the estimates.

	Assumed model	**Actual model**					
		Gaussian	*Exponential*	*Spheric*	*Gneiting-2*	*Gneiting-4*	*Stable*
$E[\mu]$	*Gaussian*	-0.01561	0.00879	-0.00370	-0.00682	-0.00591	-0.00719
$\sqrt{V}[\mu]$		0.26557	0.33574	0.26479	0.26428	0.23852	0.28018
$E[\sigma]$		0.96491	0.92431	0.95997	0.95779	0.96331	0.95313
$\sqrt{V}[\sigma]$		0.15871	0.14485	0.15054	0.14886	0.14391	0.15164
$E[c]$		0.18616	0.19003	0.18115	0.18125	0.15987	0.18943
$\sqrt{V}[c]$		0.04866	0.04937	0.04834	0.04761	0.04334	0.04833
$E[\mu]$	*Exponential*	-0.01691	0.00819	-0.00465	-0.00743	-0.00636	-0.00822
$\sqrt{V}[\mu]$		0.26766	0.33670	0.26901	0.26928	0.24161	0.28612
$E[\sigma]$		0.97507	0.94208	0.97033	0.96878	0.98019	0.96379
$\sqrt{V}[\sigma]$		0.15738	0.15643	0.14786	0.14743	0.14383	0.15144
$E[c]$		0.15978	0.18332	0.15415	0.15488	0.13637	0.16663
$\sqrt{V}[c]$		0.07237	0.10206	0.06882	0.06671	0.04488	0.07684
$E[\mu]$	*Spheric*	-0.01678	0.00895	-0.00319	-0.00608	-0.00559	-0.00724
$\sqrt{V}[\mu]$		0.26787	0.33771	0.26437	0.26504	0.23922	0.28244
$E[\sigma]$		0.96407	0.93294	0.95724	0.95664	0.96254	0.95409
$\sqrt{V}[\sigma]$		0.15775	0.15715	0.14883	0.14861	0.14420	0.15439
$E[c]$		0.39779	0.42498	0.37977	0.38290	0.32717	0.40857
$\sqrt{V}[c]$		0.13999	0.17716	0.13092	0.12968	0.11655	0.13661
$E[\mu]$	*Gneiting_2*	-0.01606	0.00869	-0.00367	-0.00658	-0.00591	-0.00672
$\sqrt{V}[\mu]$		0.26562	0.33586	0.26431	0.26451	0.23848	0.28065
$E[\sigma]$		0.96339	0.92530	0.95762	0.95618	0.96193	0.95213
$\sqrt{V}[\sigma]$		0.15719	0.14800	0.14805	0.14763	0.14320	0.15087
$E[c]$		0.47023	0.48307	0.45356	0.45561	0.38768	0.48008
$\sqrt{V}[c]$		0.14892	0.15477	0.14815	0.14549	0.14036	0.14762
$E[\mu]$	*Gneiting_4*	-0.01582	0.00868	-0.00370	-0.00674	-0.00590	-0.00709
$\sqrt{V}[\mu]$		0.26550	0.33603	0.26434	0.26425	0.23843	0.28020
$E[\sigma]$		0.96337	0.92362	0.95837	0.95650	0.96224	0.95183
$\sqrt{V}[\sigma]$		0.15713	0.14497	0.14900	0.14786	0.14348	0.15054
$E[c]$		0.52897	0.54340	0.51166	0.51381	0.43139	0.54183
$\sqrt{V}[c]$		0.17708	0.17580	0.17713	0.17493	0.17480	0.17252
$E[\mu]$	*Stable*	-0.01612	0.00872	-0.00368	-0.00671	-0.00593	-0.00707
$\sqrt{V}[\mu]$		0.26576	0.33509	0.26493	0.26480	0.23879	0.28082
$E[\sigma]$		0.96484	0.92617	0.95936	0.95756	0.96350	0.95351
$\sqrt{V}[\sigma]$		0.15764	0.14663	0.14897	0.14806	0.14335	0.15124
$E[c]$		0.18257	0.18903	0.17708	0.17744	0.15607	0.18676
$\sqrt{V}[c]$		0.05182	0.05451	0.05034	0.04950	0.04283	0.05157

Appendix C

Proofs of Chapter 6

In this appendix the proofs of the lemmas and the theorem of Chapter 6 are presented.

Proof of Lemma 6.3.2.

$$\begin{aligned}
&\Delta^{nd}((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\
&= \max(||\pi_1 - (y_j, z_j)^T||, ||\pi_2 - (y_{j+1}, z_{j+1})^T||) = \\
&= 2C \max(t_1 - x_j, x_{j+1} - t_2) = C \max\left((x_{j+1} - x_j) - \frac{\delta y}{L_1}, (x_{j+1} - x_j) - \frac{\delta z}{L_2}\right).
\end{aligned}$$

□

Proof of Lemma 6.3.4. Since the point $\hat{t}$ is such that it is possible that either $y_{\hat{t}} = y_j$, or $z_{\hat{t}} = z_j$, using (6.22) the following is obtained

$$\begin{aligned}
&\Delta^{d}((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\
&= C \max_{f(\cdot)\in\Psi} \max\left((\hat{t} - x_j) - \frac{|y_j - y_{\hat{t}}|}{L_1}, (\hat{t} - x_j) - \frac{|z_j - z_{\hat{t}}|}{L_2}\right) = \\
&= C(\hat{t} - x_j) = C\left((x_{j+1} - x_j) - \min\left(\frac{\delta y}{L_1}, \frac{\delta z}{L_2}\right)\right), \qquad \text{(C.1)}
\end{aligned}$$

from which (6.25) follows directly. □

Proof of Lemma 6.3.7. Expanding $\Delta((r_1, r_2), (f(r_1), f(r_2)))$ according to (6.26)

$$\begin{aligned}&\bar{\Delta}((r_1, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ &= C \max_{f(\cdot)\in\Psi} \max\left((r_2 - r_1) - \frac{|y_{r_1} - y_{r_2}|}{L_1}, (r_2 - r_1) - \frac{|z_{r_1} - z_{r_2}|}{L_2}\right).\end{aligned} \tag{C.2}$$

When the subinterval $[r_1, r_2]$ is not longer than $\beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1})))$, a possibility exists that the objective values at the endpoints are equal, i. e. $y_{r_1} = y_{r_2}$ or $z_{r_1} = z_{r_2}$. Therefore $\bar{\Delta}(\cdot)$ grows with the length of the subinterval, but never exceeds $\beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1})))$. □

Proof of Lemma 6.3.8. Using (6.28) for interval $[t_1, t_2]$, the following is obtained

$$\beta((t_1, t_2), (f(t_1), f(t_2))) = (t_2 - t_1) - \min\left(\frac{|y_{t_1} - y_{t_2}|}{L_1}, \frac{|z_{t_1} - z_{t_2}|}{L_2}\right). \tag{C.3}$$

Then since $[u, v] \subset [t_1, t_2]$, using (6.29)

$$\begin{aligned}&\max_{f(\cdot)\in\Psi} \bar{\Delta}((u, v, t_1, t_2), (f(t_1), f(t_2))) = \\ &= \max_{f(\cdot)\in\Psi} C \times \begin{cases} v - u, & \text{if } v - u \leqslant \beta((t_1, t_2), (f(t_1), f(t_2))), \\ \beta((t_1, t_2), (f(t_1), f(t_2))), & \text{otherwise,} \end{cases} = \\ &= C \times \begin{cases} v - u, & \text{if } v - u \leqslant \max_{f(\cdot)\in\Psi} \beta((t_1, t_2), (f(t_1), f(t_2))), \\ \max_{f(\cdot)\in\Psi} \beta((t_1, t_2), (f(t_1), f(t_2))), & \text{otherwise.} \end{cases}\end{aligned} \tag{C.4}$$

It is necessesary to show that (C.4) gives the same result as

$$\begin{aligned}&\bar{\Delta}((u, v, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) = \\ &= C \times \begin{cases} v - u, & \text{if } v - u \leqslant \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \\ \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), & \text{otherwise.} \end{cases}\end{aligned} \tag{C.5}$$

Let us first discuss the case when $t_2 - t_1 \leqslant \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1})))$. Since it is then possible that either $y_{t_1} = y_{t_2}$, or $z_{t_1} = z_{t_2}$, $\max_{f(\cdot)\in\Psi} \beta((t_1, t_2), (f(t_1), f(t_2))) =$

$t_2 - t_1 \geqslant v - u$. On the other hand, $\beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) \geqslant t_2 - t_1 \geqslant v - u$. Thus, both (C.4) and (C.5) result in $C(v - u)$.

Now let us analyze the case $t_2 - t_1 > \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1})))$. Then

$$\begin{aligned}
&\min_{f(\cdot)\in\Psi} \min \left(\frac{|y_{t_1} - y_{t_2}|}{L_1}, \frac{|z_{t_1} - z_{t_2}|}{L_2} \right) = \\
&= (t_2 - t_1) - \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))). \qquad \text{(C.6)}
\end{aligned}$$

Therefore

$$\max_{f(\cdot)\in\Psi} \beta((t_1, t_2), (f(t_1), f(t_2))) = \beta((x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) \qquad \text{(C.7)}$$

and expressions (C.4) and (C.5) coincide. □

Proof of Theorem 6.3.9. Let us first notice that (6.31) is equal to

$$\begin{aligned}
\bar{\Delta}^*(\cdot) \quad &= \max \quad \{ \\
&\qquad \max_{f(\cdot)\in\Psi} \bar{\Delta}((r_1, a, x_j, b), (f(x_j), f(b))), \\
&\qquad \max_{f(\cdot)\in\Psi} \bar{\Delta}((a, b, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \\
&\qquad \max_{f(\cdot)\in\Psi} \bar{\Delta}((b, r_2, a, x_{j+1}), (f(a), f(x_{j+1}))) \\
&\quad \} = \qquad \text{(C.8)} \\
&= \max \quad \{ \\
&\qquad \bar{\Delta}((r_1, a, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \\
&\qquad \bar{\Delta}((a, b, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))), \\
&\qquad \bar{\Delta}((b, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) \\
&\quad \}. \qquad \text{(C.9)}
\end{aligned}$$

The equality of (C.8) and (C.9) is based on the Lemma 6.3.8. Then, using (6.29),

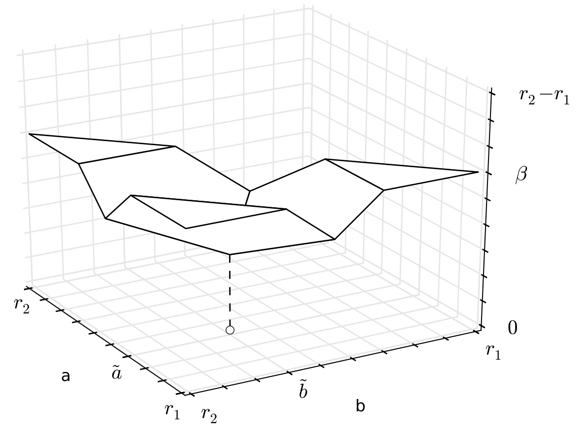

Figure C.1: The surface, formed by $\frac{1}{C}\bar{\Delta}^*(\cdot)$. The minimizer of this surface is $\tilde{a} = r_1 + \frac{1}{3}(r_2 - r_1), \tilde{b} = r_1 + \frac{2}{3}(r_2 - r_1)$.

$$\begin{aligned}
\frac{1}{C}\bar{\Delta}((r_1, a, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) &= \begin{cases} a - r_1, & \text{if } r_1 \leqslant a \leqslant r_1 + \beta, \\ \beta, & \text{if } r_1 + \beta \leqslant a \leqslant r_2; \end{cases} \\
\frac{1}{C}\bar{\Delta}((a, b, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) &= \begin{cases} b - a, & \text{if } a \leqslant b \leqslant a + \beta, \\ \beta, & \text{if } a + \beta \leqslant b \leqslant r_2; \end{cases} \\
\frac{1}{C}\bar{\Delta}((b, r_2, x_j, x_{j+1}), (f(x_j), f(x_{j+1}))) &= \begin{cases} \beta, & \text{if } r_1 \leqslant b \leqslant r_2 - \beta, \\ r_2 - b, & \text{if } r_2 - \beta \leqslant b \leqslant r_2. \end{cases}
\end{aligned} \tag{C.10}$$

The maximum of these terms produces the $\frac{1}{C}\bar{\Delta}^*(\cdot)$ surface, as shown in Figure C.1. The minimum of this surface corresponds to the $(\tilde{a}, \tilde{b})$ and is determined from the following system of equations, when $\frac{1}{3}(r_2 - r_1) \leqslant \beta$:

$$\tilde{a} - r_1 = \tilde{b} - \tilde{a} = r_2 - \tilde{b}. \tag{C.11}$$

This results in $\tilde{a} = r_1 + \frac{1}{3}(r_2 - r_1), \tilde{b} = r_1 + \frac{2}{3}(r_2 - r_1)$ and the minimum of $\frac{1}{C}\bar{\Delta}^*(\cdot)$ at $(\tilde{a}, \tilde{b})$ is $\frac{1}{3}(r_2 - r_1)$. In case $\frac{1}{3}(r_2 - r_1) > \beta$ the flat parts of the surface merge into a plane, and every choice of a and b leads to an equal $\frac{1}{C}\bar{\Delta}^*(\cdot) = \beta$. This completes the proof of the theorem. □

Bibliography

[1] DIRECT - a global optimization algorithm. `http://www4.ncsu.edu/~ctk/Finkel_Direct/`. Accessed on 2017-02-26.

[2] GKLS test functions generator implementation home page. `http://wwwinfo.deis.unical.it/~yaro/GKLS.html`.

[3] CGAL, Computational Geometry Algorithms Library, 1995-2016. `http://www.cgal.org`. Accessed on 2014-10-15.

[4] GSL, GNU Scientific Library, 1996-2016. `https://www.gnu.org/software/gsl/`. Accessed on 2016-08-01.

[5] R. J. Adler and J. E. Taylor. *Random fields and geometry*. Springer Science & Business Media, 2009.

[6] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[7] C. A. Baker, L. T. Watson, B. Grossman, W. H. Mason, and R. T. Haftka. Parallel global aircraft configuration design space exploration. 2000. (Preprint).

[8] G. R. Baoping, Z. Wood and W. P. Baritompa. Multidimensional bisection: the performance and the context. *Journal of Global Optimization*, 3(3):337–358, 1993.

[9] W. Baritompa. Customizing methods for global optimization: a geometric viewpoint. *Journal of Global Optimization*, 3(2):193–212, 1993.

[10] A. R. Butz. Space filling curves and mathematical programming. *Information and Control*, 12(4):314–330, 1968.

[11] J. Calvin. On a global optimization algorithm for multivariate smooth functions. (Private communication.).

[12] J. Calvin. An adaptive univariate global optimization algorithm and its convergence rate under the Wiener measure. *Informatica*, 22(4):471–488, 2011.

[13] J. Calvin and A. Žilinskas. One-dimensional P-algorithm with convergence rate O(n-3+δ) for smooth functions. *Journal of Optimization Theory and Applications*, 106(2):297–307, 2000. ISSN 0022-3239. doi: http://dx.doi.org/10.1023/A:1004699313526. URL `http://dx.doi.org/10.1023/A%3A1004699313526`.

[14] J. Calvin and A. Žilinskas. On convergence of a P-algorithm based on a statistical model of continuously differentiable functions. *Journal of Global Optimization*, 19(3):229–245, 2001.

[15] J. M. Calvin. A one-dimensional optimization algorithm and its convergence rate under the Wiener measure. *Journal of complexity*, 17(2):306–344, 2001.

[16] J. M. Calvin and A. Žilinskas. On a global optimization of bivariate smooth functions. *Journal of Optimization Theory and Applications*, 163: 528–547, 2014.

[17] J. M. Calvin and A. Žilinskas. One-dimensional global optimization for observations with noise. *Computers & Mathematics with Applications*, 50: 157–169, 2005.

[18] J. M. Calvin, Y. Chen, and A. Žilinskas. An adaptive univariate global optimization algorithm and its convergence rate for twice continuously differentiable functions. *Journal of Optimization Theory and Applications*, 155(2):628–636, 2012. ISSN 0022-3239. doi: http://dx.doi.org/10.1007/s10957-012-0060-3. URL `http://dx.doi.org/10.1007/s10957-012-0060-3`.

[19] J. Clausen and A. Žilinskas. *Global optimization by means of branch and bound with simplex based covering*. IMM, Department of Mathematical Modelling, Technical University of Denmark, 1998.

[20] Yu. M. Danilin. Estimation of the efficiency of an absolute-minimum-finding algorithm. *USSR Computational Mathematics and Mathematical Physics*, 11(4):261–267, 1971.

[21] Yu. M. Danilin and S. A. Pijavskij. An algorithm for finding the absolute minimum. *Theory of Optimal Decisions*, 2:25–37, 1967.

[22] K. Deb. *Multi-objective optimization using evolutionary algorithms*. J. Wiley, 2009.

[23] C. Fonseca and P. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In W. Ebeling, Rechenberg

I., H. P. Schwefel, and H. M. Voigt, editors, *Parallel Problem Solving from Nature*, volume 1141 of *Lecture notes in Computer Science*, pages 584–593, Berlin, 1996. Springer.

[24] A. Forrester and A. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79, 2009.

[25] J. M. Gablonsky. An implementation of the DIRECT algorithm. Technical report, North Carolina State University, Raleigh, N. C., USA, August 1998.

[26] J. M. Gablonsky. *Modifications of the DIRECT algorithm*. PhD thesis, North Caroline State University, Raleigh, North Carolina, 2001.

[27] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001.

[28] E. A. Galperin. The cubic algorithm. *Journal of Mathematical Analysis and Applications*, 112(2):635–640, 1985.

[29] M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software*, 29(4):469–480, 2003.

[30] G. Gimbutienė and A. Žilinskas. A two-phase global optimization algorithm for black-box functions. *Baltic Journal of Modern Computing*, 3(3): 214, 2015. ISSN 2255-8950.

[31] G. Gimbutienė and A. Žilinskas. Clustering-based statistical global optimization. In *AIP Conference Proceedings*, volume 1776, 2016. doi: http://dx.doi.org/10.1063/1.4965342. URL `http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.4965342`.

[32] E. Gourdin, P. Hansen, and B. Jaumard. Global optimization of multivariate Lipschitz functions: survey and computational comparison. *Les Cahiers du GERAD*, 1994.

[33] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 73–84, New York, NY, USA, 1998. ACM. ISBN 0-89791-995-5. doi: http://dx.doi.org/10.1145/276304.276312. URL `http://doi.acm.org/10.1145/276304.276312`.

[34] P. Hansen and B. Jaumard. Lipshitz optimization. In R. Horst and P. Pardalos, editors, *Handbook of Global Optimization*, pages 407 – 493.

Kluwer Academic Publisher, Dodrecht, 1995.

[35] P. Hansen, B. Jaumard, and S. H. Lu. On the number of iterations of Piyavskii's global optimization algorithm. *Mathematics of Operations Research*, 16(2):334–350, 1991.

[36] P. Hansen, B. Jaumard, and S. H. Lu. Global optimization of univariate Lipschitz functions: II. New algorithms and computational comparison. *Mathematical programming*, 55(1):273–292, 1992.

[37] R. Hooke and T. A. Jeeves. "Direct search" solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, 1961. ISSN 0004-5411. doi: http://dx.doi.org/10.1145/321062.321069. URL `http://doi.acm.org/10.1145/321062.321069`.

[38] R. Horst. A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications*, 51(2):271–291, 1986.

[39] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer Science & Business Media, 2013.

[40] R. Horst, P. Pardalos, and N. Van Thoai. *Introduction to global optimization*, volume 48 of *Nonconvex optimization and its applications*. Springer Science & Business Media, Dordrecht, 2000.

[41] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006.

[42] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.

[43] W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4):331–355, 1999.

[44] V. V. Ivanov. Optimal algorithms of minimization in the class of functions with the Lipschitz condition. *Information Processing*, 71:1324–1327, 1972.

[45] V. V. Ivanov. On optimal algorithms for the minimization of functions of certain classes. *Kibernetika*, 4:81–94, 1972.

[46] B. Jaumard, H. Ribault, and T. Herrmann. An on-line cone intersection algorithm for global optimization of multivariate Lipschitz functions. *Les Cahiers du GERAD*, 1995.

[47] D. R. Jones. A taxonomy of global optimization methods based on re-

sponse surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[48] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[49] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4): 455–492, 1998.

[50] C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, North Carolina State University, Raleigh, North Carolina, 1999.

[51] J. P. C. Kleijnen, W. van Beers, and I. van Nieuwenhuyse. Expected improvement in efficient global optimization through bootstrapped kriging. *Journal of Global Optimization*, 54(1):59–73, 2012.

[52] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Redwood City, CA, USA, 1973.

[53] H. Kushner. A versatile stochastic model of a function of unknown and time-varying form. *Journal of Mathematical Analysis and Applications*, 5: 150–167, 1962.

[54] D. E. Kvasov and Ya. D. Sergeyev. A univariate global search working with a set of Lipschitz constants for the first derivative. *Optimization Letters*, 3(2):303–318, 2009.

[55] D. E. Kvasov and Ya. D Sergeyev. Lipschitz gradients for global optimization in a one-point-based partitioning scheme. *Journal of Computational and Applied Mathematics*, 236(16):4042–4054, 2012.

[56] J. Lamperti. *Stochastic Processes: A Survey of the Mathematical Theory*, volume 23 of *Applied Mathematical Sciences*. Springer-Verlag New York, 1 edition, 1977.

[57] Q. Liu and W. Cheng. A modified DIRECT algorithm with bilevel partition. *Journal of Global Optimization*, 60(3):483–499, 2014.

[58] Q. Liu and J. Zeng. Global optimization by multilevel partition. *Journal of Global Optimization*, 61(1):47–69, 2015.

[59] G. Liuzzi, S. Lucidi, and V. Piccialli. A partition-based global optimization algorithm. *Journal of Global Optimization*, 48(1):113–128, 2010.

[60] G. Liuzzi, S. Lucidi, and V. Piccialli. A DIRECT-based approach exploiting local minimizations for the solution of large-scale global optimization

problems. *Computational Optimization and Applications*, 45(2):353–375, 2010.

[61] G. Liuzzi, S. Lucidi, and V. Piccialli. Exploiting derivative-free local searches in DIRECT-type algorithms for global optimization. *Computational Optimization and Applications*, 65(2):449–475, 2016.

[62] S. N. Løphaven, H. B. Nielsen, and J. Søndergaard. DACE: A Matlab Kriging toolbox. Technical report, Technical University of Denmark, Lyngby, 2002.

[63] D. Q. Mayne and E. Polak. Outer approximation algorithm for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications*, 42(1):19–30, 1984.

[64] C. C. Meewella and D. Q. Mayne. Efficient domain partitioning algorithms for global optimization of rational and Lipschitz continuous functions. *Journal of Optimization Theory and Applications*, 61(2):247–270, 1989.

[65] R. H. Mladineo. An algorithm for finding the global maximum of a multimodal, multivariate function. *Mathematical Programming*, 34(2):188–200, 1986.

[66] J. Mockus. On Bayes methods for seeking an extremum. *Avtomatika i Vychislitelnaja Technika*, 3:53–62, 1972. (In Russian.).

[67] J. Mockus. *Bayesian approach to global optimization*. Kluwer Academic Publishers, Dordrecht, 1988.

[68] R. Paulavičius and J. Žilinskas. Simplicial Lipschitz optimization without Lipschitz constant. In *Simplicial Global Optimization*, pages 61–86. Springer, 2014.

[69] R. Paulavičius, Y. D. Sergeyev, D. E. Kvasov, and J. Žilinskas. Globally-biased DISIMPL algorithm for expensive global optimization. *Journal of Global Optimization*, 59(2-3):545–567, 2014.

[70] S. Pijavskij. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972.

[71] J. Pintér. Extended univariate algorithms for n-dimensional global optimization. *Computing*, 36(1):91–103, 1986.

[72] J. D. Pintér. Branch-and bound algorithms for solving global optimization problems with Lipschitzian structure. *Optimization*, 19(1):101–110, 1988.

[73] J. D. Pintér. *Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications*, volume 6 of *Nonconvex*

Optimization and Its Applications. Kluwer Academic Publishers, 1996.

[74] K. Ritter. *Average-case analysis of numerical problems*, volume 1733 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2000.

[75] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[76] Martin Schlather, Alexander Malinowski, Marco Oesting, Daphne Boecker, Kirstin Strokorb, Sebastian Engelke, Johannes Martini, Felix Ballani, Olga Moreva, Peter J Menck, Sebastian Gross, Ulrike Ober, Christoph Berreth, Katharina Burmeister, Juliane Manitz, Olga Morena, Paulo Ribeiro, Richard Singleton, Ben Pfaff, and R Core Team. *RandomFields: Simulation and Analysis of Random Fields*, 2016. URL `http://CRAN.R-project.org/package=RandomFields`. R package version 3.1.8.

[77] F. Schoen. On a sequential search strategy in global optimization problems. *Calcolo*, 19(3):321–334, 1982.

[78] Y. D. Sergeyev and D. E. Kvasov. Global search based on efficient diagonal partitions and a set of Lipschitz constants. *SIAM Journal on Optimization*, 16(3):910–937, 2006. doi: http://dx.doi.org/10.1137/040621132.

[79] Ya. D. Sergeyev. Efficient strategy for adaptive partition of N-dimensional intervals in the framework of diagonal algorithms. *Journal of Optimization Theory and Applications*, 107(1):145–168, 2000.

[80] Ya. D. Sergeyev. Efficient partition of N-dimensional intervals in the framework of one-point-based algorithms. *Journal of Optimization Theory and Applications*, 124(2):503–510, 2005. ISSN 0022-3239. doi: http://dx.doi.org/10.1007/s10957-004-0948-7. URL `http://dx.doi.org/10.1007/s10957-004-0948-7`.

[81] Ya. D. Sergeyev and D. Kvasov. *Diagonal global optimization methods*. Fizmatlit, Moscow, 2008. (In Russian.).

[82] Ya. D. Sergeyev and D. Kvasov. A deterministic global optimization using smooth diagonal auxiliary functions. *Communications in Nonlinear Science and Numerical Simulation*, 21(1):99 – 111, 2015.

[83] Ya. D. Sergeyev, R. G. Strongin, and D. Lera. *Introduction to global optimization exploiting space-filling curves*. Springer Science & Business Media, 2013.

[84] B. O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[85] T. Simpson, V. Toropov, V. Balabanov, and F. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come-or not. In *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 5802, 2008.

[86] I. Sobol. On the systematic search in a hypercube. *SIAM Journal on Numerical Analysis*, 16(5):790–793, 1979.

[87] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, 2012.

[88] R. Strongin and Ya. D. Sergeyev. *Global optimization with non-convex constraints: Sequential and parallel algorithms*, volume 45. Springer Science & Business Media, 2013.

[89] R. G. Strongin. *Numerical Methods of Multiextremal Minimization*. Nauka, Moscow, 1978. (In Russian.).

[90] R. G. Strongin. Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. *Journal of Global Optimization*, 2(4):357–378, 1992.

[91] A. G. Sukharev. Optimal strategies of the search for an extremum. *USSR Computational Mathematics and Mathematical Physics*, 11(4):119–137, 1971.

[92] L. N. Timonov. Algorithm for search of a global extremum. *Engineering Cybernetics*, 15(3):38–44, 1977.

[93] A. Törn and A. Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[94] A. Žilinskas. On global one-dimensional optimization. *Izv. Acad. Nauk USSR, Eng. Cybern.*, 4:71–74, 1976. (In Russian.).

[95] A. Žilinskas. Axiomatic characterization of a global optimization algorithm and investigation of its search strategy. *Operations Research Letters*, 4(1):35–39, 1985.

[96] A. Žilinskas. Global optimization: axiomatic of statistical models, algorithms and their applications. *Mokslas*, 1986. (In Russian.).

[97] A. Žilinskas. On the worst-case optimal multi-objective global optimization. *Optimization Letters*, 7(8):1921–1928, 2013. ISSN 1862-4472. doi: http://dx.doi.org/10.1007/s11590-012-0547-8. URL `http://dx.doi.org/10.1007/s11590-012-0547-8`.

[98] A. Žilinskas. A one-step worst-case optimal algorithm for bi-objective univariate optimization. *Optimization Letters*, 8(7):1945–1960, 2014.

[99] A. Žilinskas and G. Gimbutienė. On an asymptotic property of a simplicial statistical model of global optimization. In A. Migdalas and A. Karakitsiou, editors, *Optimization, Control, and Applications in the Information Age: In Honor of Panos M. Pardalos's 60th Birthday*, volume 130 of *Springer Proceedings in Mathematics and Statistics*, pages 383–391. Springer International Publishing, Cham, 2015. ISBN 978-3-319-18567-5. doi: http://dx.doi.org/10.1007/978-3-319-18567-5_20.

[100] A. Žilinskas and G. Gimbutienė. On one-step worst-case optimal trisection in univariate bi-objective Lipschitz optimization. *Communications in Nonlinear Science and Numerical Simulation*, 35:123 – 136, 2016. ISSN 1007-5704. doi: http://dx.doi.org/10.1016/j.cnsns.2015.11.002.

[101] A. Žilinskas and G. Gimbutienė. Statistical models for global optimization: how to choose an appropriate one? In A. M. A. C. Rocha, M. Fernanda P. Costa, and E. M. G. P. Fernandes, editors, *Proceedings of the XIII global optimization workshop GOW'16*, University of Minho, Braga, Portugal, 2016. ISBN 978-989-20-6764-3.

[102] A. Žilinskas and E. Senkiene. On estimation of the parameter of a Wiener random field at random dependent points. *Cybernetics*, (6):107–109, 1979. (In Russian.).

[103] A. Žilinskas and A. Zhigljavsky. Stochastic global optimization: a review on the occasion of 25 years of Informatica. *Informatica*, 27(2):229–256, 2016.

[104] G. R. Wood. Multidimensional bisection applied to global optimisation. *Computers & Mathematics with Applications*, 21(6-7):161–172, 1991.

[105] A. Zhigljavsky. Mathematical theory of global random search. *Leningrad University Press*, 1985. (In Russian.).

[106] A. Zhigljavsky and Žilinskas. Stochastic global optimization. *Springer*, 2008.

[107] A. Žilinskas. Axiomatic approach to extrapolation problem under uncertainty. *Automatics and Remote Control*, 12:66–70, 1979.

[108] A. Žilinskas. Axiomatic approach to statistical models and their use in multimodal optimization theory. *Mathematical Programming*, 22(1):104–116, 1982.

[109] A. Žilinskas. A review of statistical models for global optimization. *Journal of Global Optimization*, 2(2):145–153, 1992.

[110] A. Žilinskas. A statistical model for global optimization by means of

select and clone. *Optimization: A Journal of Mathematical Programming and Operations Research*, 48(1):117–135, 2000. doi: http://dx.doi.org/10.1080/02331930008844497.

[111] A. Žilinskas. On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions. *Applied Mathematics and Computation*, 218(16):8131–8136, 2012.

[112] A. Žilinskas. *On the Statistical Models-Based Multi-objective Optimization*, pages 597–610. Springer, New York, 2014. ISBN 978-1-4939-0808-0. doi: http://dx.doi.org/10.1007/978-1-4939-0808-0_29.

[113] A. Žilinskas and J. Žilinskas. Global optimization based on a statistical model and simplicial partitioning. *Computers & Mathematics with Applications*, 44(7):957–967, 2002.

[114] A. Žilinskas and J. Žilinskas. P-algorithm based on a simplicial statistical model of multimodal functions. *Top*, 18(2):396–412, 2010.

[115] A. Žilinskas and J. Žilinskas. Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems. *Communications in Nonlinear Science and Numerical Simulation*, 21(1):89–98, 2015.

Publications by the Author

Periodicals

1. A. Žilinskas and G. Gimbutienė. On one-step worst-case optimal trisection in univariate bi-objective Lipschitz optimization. *Communications in Nonlinear Science and Numerical Simulation*, 35:123 – 136, 2016. ISSN 1007-5704. doi: http://dx.doi.org/10.1016/j.cnsns.2015.11.002.

2. G. Gimbutienė and A. Žilinskas. A two-phase global optimization algorithm for black-box functions. *Baltic Journal of Modern Computing*, 3(3): 214, 2015. ISSN 2255-8950.

Peer Reviewed Conference Proceedings

3. A. Žilinskas and G. Gimbutienė. Statistical models for global optimization: how to choose an appropriate one? In A. M. A. C. Rocha, M. Fernanda P. Costa, and E. M. G. P. Fernandes, editors, *Proceedings of the XIII global optimization workshop GOW'16*, University of Minho, Braga, Portugal, 2016. ISBN 978-989-20-6764-3.

4. G. Gimbutienė and A. Žilinskas. Clustering-based statistical global optimization. In *AIP Conference Proceedings*, volume 1776, 2016. doi: http://dx.doi.org/10.1063/1.4965342. URL `http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.4965342`.

5. A. Žilinskas and G. Gimbutienė. On an asymptotic property of a simplicial statistical model of global optimization. In A. Migdalas and A. Karakit-

siou, editors, *Optimization, Control, and Applications in the Information Age: In Honor of Panos M. Pardalos's 60th Birthday*, volume 130 of *Springer Proceedings in Mathematics and Statistics*, pages 383–391. Springer International Publishing, Cham, 2015. ISBN 978-3-319-18567-5. doi: http://dx.doi.org/10.1007/978-3-319-18567-5_20.

Conference Presentations

1. G. Gimbutienė and A. Žilinskas. Investigation of the effect of the assumed statistical model in global optimization. Poster presentation at the *8-th International Workshop "Data Analysis Methods for Software Systems"*, Druskininkai, 2016.

2. A. Žilinskas and G. Gimbutienė. Statistical models for global optimization: how to choose an appropriate one? Presentation at the XIII-th Global Optimization Workshop GOW'16, Braga, Portugal, 2016.

3. G. Gimbutienė and A. Žilinskas. Clustering-based statistical global optimization. Presentation at the 2-nd International Conference and Summer School *Numerical Computations: Theory and Algorithms*, Pizzo Calabro, Italy, 2016.

4. A. Žilinskas and G. Gimbutienė. On one-step worst-case optimal trisection in univariate bi-objective Lipschitz optimization. Poster presentation at the *7-th International Workshop "Data Analysis Methods for Software Systems"*, Druskininkai, 2015.

5. G. Gimbutienė and A. Žilinskas. Dviejų etapų globalaus optimizavimo algoritmas juodos dėžės funkcijoms. Presentation at *Kompiuterininkų dienos - 2015*, Panevėžys, 2015.

6. A. Žilinskas and G. Gimbutienė. On an asymptotic property of a simplicial statistical model of global optimization. Poster presentation at the *6-th International Workshop "Data Analysis Methods for Software Systems"*, Druskininkai, 2014.

Gražina Gimbutienė

ALGORITHMS FOR NON-CONVEX GLOBAL OPTIMIZATION BASED ON THE STATISTICAL AND LIPSCHITZ OBJECTIVE FUNCTION MODELS

Doctoral dissertation

Physical sciences (P000)

Informatics (09P)

Editor Nijolė Požėraitytė