Evolution strategies for robust optimization

Kruisselbrink, J.W.

Evolution Strategies for Robust Optimization

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof. mr. P.F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 10 mei 2012
klokke 11:15 uur

door

Johannes Willem Kruisselbrink
geboren te Nijmegen
in 1983

Promotiecommissie

Promotor:	Prof. Dr. T.H.W. Bäck
Copromotor:	Dr. M.T.M. Emmerich
Overige leden:	Prof. Dr. J. Branke (University of Warwick)
	Prof. Dr. B. Sendhoff (University of Darmstadt)
	Prof. Dr. J.N. Kok
	Dr. W.A. Kosters

Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Het onderzoek beschreven in dit proefschrift is uitgevoerd aan het Leiden Institute of Advanced Computer Science (LIACS), Universiteit Leiden. De totstandkoming van dit proefschrift is financieel ondersteund door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO), project 612.066.618 “RoDeO”.

Ontwerp omslag: Gilian Algra

ISBN: 978-94-6191-250-3

Contents

Chapter 1

Introduction

When solving real-world optimization problems a frequently encountered difficulty is the presence of uncertainties and noise within the system for which optima are sought. Due to various reasons, various types of uncertainties and noise can arise in optimization problems. Hence, for real-world scenarios, optimization methods are needed that can deal with these uncertainties and solutions ought to be found that are not only optimal in the theoretical sense, but that are also practical in real life. The practice of optimization that accounts for uncertainties and noise is often referred to as *robust optimization*.

Evolutionary Algorithms (EAs) form a class of optimization algorithms that use the principle of evolution to find good solutions to optimization problems. The paradigm of *evolutionary computation* is simple and effective; take a population of candidate solutions for the optimization problem and simulate the process of evolution to evolve the population toward better solutions. With applications in various real-world settings, Evolutionary Algorithms are well-established and have proven to be powerful for optimization, especially for complex problems that are difficult or impossible to solve analytically.

Evolutionary Algorithms are originally proposed for solving optimization problems that are not affected by uncertainties and noise. However, natural evolution seems to be inherently robust when viewing it as an optimization process, because uncertainty and noise are indispensable parts of nature. Being inspired by evolution in nature, one question is therefore to what extent the natural mechanisms adopted by Evolutionary Algorithms make them robust against uncertainties and noise. Additionally, when acknowledging the limited precision of natural evolution, the challenge is to make Evolutionary Algorithms even more effective in the scope of robust optimization.

In this work we will study the application of Evolutionary Algorithms, and in particular *Evolution Strategies*, in the context of robust optimization. The main questions that we will try to answer are: What is robust optimization and how does it relate to the traditional view on optimization? In what ways can noise and uncertainties emerge within an optimization model? To what extent are Evolutionary Algorithms indeed robust against uncertainties and noise in

the optimization model? What can be done to fix or improve the behavior of Evolutionary Algorithms when dealing with uncertainties and noise?

1.1 A Brief History

The concept of robust (design) optimization originated from the concept of *robust design* in industrial engineering. The introduction, or better, popularization of this concept is generally attributed to Taguchi [Tag78, Tag86, Tag89]. Taguchi stated that for product design engineering, a product should be designed not only for optimal performance, but also as to make the performance as insensitive as possible to variations that are beyond the designer's control. He proposed a method based on simple experimental designs and *loss functions* to incorporate the notion of quality or performance robustness into the process of design engineering. Taguchi bundled his ideas in a design method called the *Taguchi method* (see e.g., [Pha89]). Although the methodologies proposed by Taguchi have received much criticism [BST+88, AMB+92] and are considered to be outdated [CWZ99], the impact of the robust design philosophy cannot be neglected. The general aim of trying to take uncertainty and noise into account within the scope of product development has become a standard part of modern engineering [Par07].

In modern engineering, the increasing use of computer models to virtually test (parts of) designs has also led to increased interest in computational optimization techniques to aid the design process [EH02]. However, classical optimization methods focus mainly on finding optimal solutions for exact, noise-free systems. Due to the frequently noisy and uncertain nature of real life, the increased application of optimization techniques for real-world applications led to the awareness that there are a number of problems that require ways of incorporating the notion of robustness in the measures of solution quality (which is the leading principle of robust design) in optimization techniques [Tro97, TAW03].

As noted by Trosset [Tro97], robust optimization is only a part of the broader concept of robust design. The general aim of obtaining high quality products often involves *design of experiments* methods and *sensitivity analysis* where the optimization is performed manually by means of statistical analysis. Only in some cases automated optimization is used to aid the design process, and robust optimization methods can be applied to find robust solutions.

However, the inverse can also be said to be true; although the concept of robust optimization seems to borrow its right of existence directly from the concept of robust design, its context is much broader than only finding solutions which are stable under varying conditions. Beyer and Sendhoff [BS07] consider in their overview paper not only the robustness of the designs (or, in the scope of optimization, the robustness of optimal solutions), but also the robustness of the optimization process itself. Hence, robust optimization also deals with uncertainties of the optimization model, such as noisy output, and uncertainty within the optimization model, such as aiming to find robust optima. Also other studies, such as [SJ08] and [JB05], consider a broader view of uncertainties and noise in the complete optimization model. The work of

[SJ08] is particularly worthwhile mentioning as it presents a large number of applications of robust optimization techniques within the field of mechanical engineering.

In Operations Research, the problem of dealing with uncertainties and noise comprises a number of variants. Studies that consider uncertainty in the optimization model date back to the work of Dantzig [Dan55] in 1955 and Wets [Wet66] in 1966. Today, approaches dealing with uncertainties can be found in various settings in the scope of mathematical programming; in the form of *stochastic programming* (e.g., [KW94]), under the term *robust optimization* [MVZ95, BTGGN04, BBC11], and in the scope of *fuzzy programming* [BZ70, IR00] which knows the two types of *flexible programming* [TOA74, Zim76] and *possibilistic programming* [TA84]. In [Sah04], an overview can be found of the different mathematical programming classes that deal with uncertainty and noise, and the way in which these are modeled into optimization problems.

Also in the scope of *black-box optimization*, and particularly in the field of evolutionary computation, there is an increasing interest for methods that deal with uncertainty and noise. A summary of the way in which such optimization problems are approached within this field, and references to approaches proposed in literature can be found in [JB05].

Finally, some of the challenges and opportunities stated by Sahinidis [Sah04] in an overview of robust optimization using mathematical programming techniques also hold for the broader scope of optimization under uncertainty and noise. The following list of challenges, based on those stated by Sahinidis, forms the starting point of this thesis:

- An effort should be made to construct a unified framework which combines the different philosophies of modeling uncertainty and noise within optimization problems.

- Many approaches focus solely on one particular type of uncertainty and noise, whereas real-world optimization problems can exhibit multiple types. Hybrid approaches that can deal with various types of uncertainties and noise simultaneously can be the next step of robust optimization.

- Non-standard search spaces have received limited attention within the scope of optimization under uncertainty and noise. Extending the modeling framework and also the optimization approaches such that they also allow for optimization under uncertainty and noise in other domains (e.g., graph-like spaces) is a challenging next step.

- Accounting for uncertainty and noise will lead to higher computational demands in order to obtain solutions of reasonable quality. There is an ongoing need of smart sampling and search methods that further limit the computational effort (e.g., the number of function evaluations) for solving optimization problems that are subject to uncertainty and noise.

1.2 Aim and Objectives

The first objective of the current work is to obtain a clear formulation of robust optimization. The term *robustness* is in itself a very general term, obtaining a clear view on the scope and goals of robust optimization is therefore essential. An objective emerging from this is to provide guidelines for systematic classification of various types of robust optimization problems.

The second objective is to bridge the gap from the formulation of robust optimization to the practice of robust optimization. The aim is to exemplify how to approach robust optimization problems. Moreover, we aim to study the behavior of Evolution Strategies on such problems and find how they should be adapted in order to better deal with such problems. For this, approaches from the literature and newly proposed ideas will be compared conceptually and empirically using two Evolution Strategy variants as algorithmic cores.

The third objective is to provide a framework for empirical comparison for the two robust optimization scenarios considered in this work. This is done by providing a small, focused set of benchmark problems and empirical results that can, in term, be used as benchmarks.

1.3 Overview of this Thesis

This work consists of two parts. In the first part, the aim is to form an exact conceptual picture of what robust optimization is, how it relates to traditional non-robust optimization, and what the implications are for solving real-world optimization problems. The second part of this work focuses on the application of Evolution Strategies (ES) targeted on solving real-parameter robust optimization problems. In particular two main scenarios of robust optimization are considered: *optimization of noisy objective functions* and *finding robust optima*. These are considered as frequently occurring and representative scenarios of robust optimization. For these two scenarios, the performance and possible ways of improvement of two particular Evolution Strategy instances, namely the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, are studied.

Chapter 2 lays out the background by providing an overview on the classical view on optimization, together with frequently used terms and concepts. This chapter forms the backbone of this work.

Chapter 3 extends the classical view on optimization to a framework and definition of robust optimization. It provides a taxonomy of the different ways in which uncertainties and noise can arise within optimization problems and a unified and concise framework for their systematic classification. This chapter is partially based on insights and results previously published in [KBIvdH08, KAE+09, KEB+09b].

Chapter 4 introduces Evolutionary Algorithms and Evolution Strategies, therewith forming the bridge between the (robust) optimization problem specification and the practice of solving optimization problems.

Chapter 5 discusses the application of Evolution Strategies to noisy objective functions. In this chapter, the negative effects of noise on Evolution Strategies are studied and techniques to counter the effects of noise in the objective function are compared conceptually. In Section 5.4.6 an alternative type of uncertainty quantification is proposed and discussed. This chapter uses results and insights that have been published partly in [KEB09a, KRD+11].

Chapter 6 focuses on a particular class of noise handling techniques, namely adaptive averaging techniques. For these techniques the main question is how these compare to straightforward ways of dealing with noisy objective functions. This chapter presents new results on accuracy limits for adaptive noise handling in Evolution Strategies with the theoretical results presented in Section 6.1 and the empirical comparisons of different noise handling schemes.

Chapter 7 focuses on the scenario of finding robust optima in anticipation of uncertainties/noise in the design variables. The goal of finding robust optima is explicitly stated and formulated in the light of the framework of robust optimization. Different techniques that are proposed for finding robust optima are reviewed and compared conceptually. This chapter merges the individual results on algorithmic schemes of [KEDB10a, KEB10, KEDB10b, KRD+11] with each other and puts them into a global scope of existing studies.

Chapter 8 presents an empirical comparison of different strategies for finding robust optima. This chapter presents new results with an empirical comparison of different techniques for finding robust optima, amongst which are the algorithms presented in [KEDB10a, KEB10, KEDB10b, KRD+11].

Chapter 9 closes with a summary and an outlook.

Last, but not least, **Appendix A** and **Appendix B** contain collections of benchmark problems that can be used for empirical comparison of optimization algorithms for robust optimization scenarios. Appendix A provides descriptions of benchmark problems for optimization of noisy objective functions and Appendix B provides descriptions of benchmark problems for finding robust optima. These benchmark sets are used in the empirical studies of Chapter 6 and Chapter 8, respectively. **Appendix C** provides a brief description of Kriging, which is used as metamodeling technique in algorithmic approaches considered in Chapter 7 and Chapter 8.

Part I

From Optimization to Robust Optimization

Chapter 2

Optimization

This chapter lays out the background of this work by providing a summary of black-box optimization. It introduces the "classical" view on optimization together with the concepts and terminology that are commonly used within this context. This classical view on optimization will be extended in Chapter 3 to form a definition of robust optimization.

Section 2.1 starts with providing a global description and a formal definition of a black-box optimization problem as it will be used in this thesis. Section 2.2 discusses how the practice of optimization is generally perceived in practical applications. Section 2.3 zooms in on the concept of objective function landscape, which is a frequently used metaphor for perceiving optimization problems. Section 2.4 focuses on real-parameter optimization problems, being the main type of optimization problem discussed in this thesis. Section 2.5 provides an overview of black-box optimization algorithms and the general goal of automated optimization. Section 2.6 closes with a summary and discussion.

2.1 Optimization Problems

The model of Figure 2.1 describes an optimization problem. It considers a system that produces output y as a function of input x. Keeping it as general as possible, we note that x and y can be of any form and assume that there is no knowledge about the internal mechanisms of the system, i.e., it is considered to be a black-box. Given such a system and a large number of possible input settings, the central problem of optimization can be loosely formulated as:

What setting(s) of the input x yield(s) the best possible (optimal) output y?

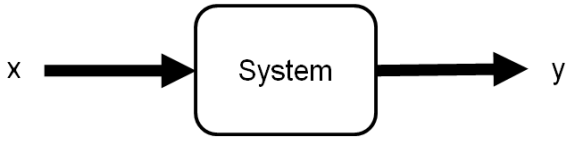


Figure 2.1: The general black-box model of a system.

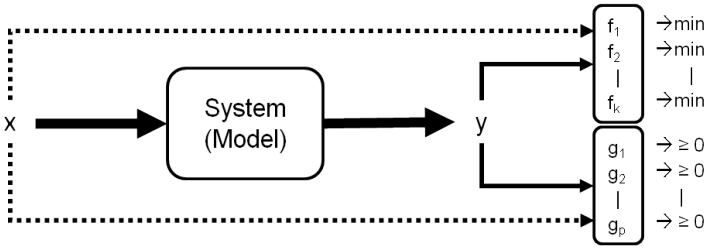


Figure 2.2: The general model of an optimization problem.

To deal with problems of this type, the model of Figure 2.1 is commonly transformed into a form as depicted in Figure 2.2. In many cases the system that is considered is replaced by an abstract model of the system (e.g., a mathematical model or a simulator). Furthermore, one or more objective functions $f_1, \ldots, f_k$ and optionally also a number of constraint functions $g_1, \ldots, g_p$ are introduced. The objective functions are of the form $f_i : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and assign score values to each input $x \in \mathcal{X}$ based on its respective output $y \in \mathcal{Y}$. Note that we could also neglect the intermediate mapping $\mathcal{X} \to \mathcal{Y}$, which yields the more common form $f_i : \mathcal{X} \to \mathbb{R}$. Without loss of generality, we assume that each score function is to be minimized. The constraint functions are of the form $g_i : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and rate the feasibility of each possible input x, again using y. Also here the intermediate mapping could be neglected to obtain the more common form $g_i : \mathcal{X} \to \mathbb{R}$. For a possible input x, it should hold that $g_i(x) \geq 0$ in order for x to be feasible[1].

The model of an optimization problem of Figure 2.2 can also be described mathematically:

Definition 2.1.1 (Optimization Problem)**:** An *optimization problem* is a triple $(\mathcal{X}, \mathcal{F}, \mathcal{G})$, where:

- $\mathcal{X}$ is the search space, which is the nonempty set of all possible solutions.
- $\mathcal{F} = \{f_1, \ldots, f_k\}$, $k \in \mathbb{N}_1$, is a set of one or more objective functions that are to be minimized. Each objective function is a function of the form $f : \mathcal{X} \to \mathbb{R}$ that maps elements of the search space to a score value.
- $\mathcal{G} = \{g_1, \ldots, g_p\}$, $p \in \mathbb{N}_0$, is a set of constraint functions that need to be satisfied. Each constraint function is of the form $g : \mathcal{X} \to \mathbb{R}$ mapping elements of the search space to a constraint value. For a certain input $x \in \mathcal{X}$ a constraint g is said to be satisfied if and only if $g(x) \geq 0$. Otherwise, if $g(x) < 0$, then solution x violates the constraint and is therefore infeasible.

[1]Constraints of the form $g(x) \geq 0$ are referred to as inequality constraints. In literature, also another type of constraint is used, namely the equality constraint, which is of the form $h(x) = 0$. In the definition given here, equality constraints are not included because they can easily be constructed using two inequality constraints (i.e., $g(x) \geq 0 \wedge -g(x) \geq 0 \Leftrightarrow g(x) = 0$).

Furthermore, we use the following definition of feasibility:

Definition 2.1.2 (Feasible Solution and Set of Feasible Solutions)**:** For an optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$, the *set of feasible solutions* $\mathcal{A}$ is the set

$$\mathcal{A} = \{x \in \mathcal{X} \mid g_i(x) \geq 0, i = 1, \ldots, p\}, \tag{2.1}$$

and each solution $x \in \mathcal{A}$ is called a *feasible solution*.

Given a problem that fits Definition 2.1.1, the goal of optimization can still vary and roughly be of one of the following forms:

1. Find *a* feasible solution that is optimal with respect to the objective function(s).
2. Find *all* feasible solutions that are optimal with respect to the objective function(s).
3. Find *a specific set of* feasible solutions that are optimal with respect to the objective function(s).

Here, "a specific set of solutions" loosely denotes the cases where, either implicitly or explicitly, also a secondary set selection criterion encompasses the optimization problem (e.g., searching for a diverse set of solutions requires a diversity notion based on sets of solutions). In addition to these three aims, a second class of optimization goals can also be identified in which the aim is to find solutions of which the objective function value(s) satisfies/satisfy (a) certain threshold value(s):

4. Find *a* feasible solution of which the objective function value(s) satisfies/satisfy (a) certain threshold value(s).
5. Find *all* feasible solutions of which the objective function value(s) satisfy (a) certain threshold value(s).
6. Find *a specific set of* feasible solutions of which the objective function value(s) satisfy (a) certain threshold value(s).

Note that the latter three aims could also be seen as constraint satisfaction problems (i.e., an objective function with a threshold is effectively a constraint function).

Given the definition of an optimization problem and the loosely defined possible goals of optimization, next we will give more formal definitions of optimality. However, in order to do this, we will make a distinction between *single objective optimization problems* and *multi-objective optimization problems* and provide separate definitions for both classes.

2.1.1 Single Objective Optimization Problems

A single objective optimization problem is a special instance of an optimization problem, defined as

Definition 2.1.3 (Single Objective Optimization Problem)**:** A *single objective optimization problem* is an optimization problem with precisely one objective function. For the triple $(\mathcal{X}, \mathcal{F}, \mathcal{G})$, the set $\mathcal{F}$ consists of exactly one objective function ($k = 1$ in Definition 2.1.1).

Given a single objective optimization problem, based on the definition of Törn and Žilinskas [TZ89], Bäck [Bäc96] defines the *global optimization problem* as the problem of determining a global minimizer:

Definition 2.1.4 (Global Minimum/Optimum and Global Minimizer/Optimizer)**:** For a single objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and a set of feasible solutions $\mathcal{A}$, the *global minimum* or *global optimum* f^* is the value

$$f^* = \min\{f(x) \,|\, x \in \mathcal{A}\}, \tag{2.2}$$

and every solution $x^* \in \mathcal{A}$ for which it holds that $f(x^*) = f^*$ is called a *global minimizer* or *global optimizer*.

This definition falls into the first item of the enumeration on page 13. Moreover, there are two other important remarks that have to be made. First, it is important to realize that there exist objective functions for which no global minimum exists. This can happen for objective functions of which the image $f[\mathcal{A}]$ of $f : \mathcal{A} \to \mathbb{R}$ is non-compact (e.g., the infimum of $f[\mathcal{A}]$ is not included in $f[\mathcal{A}]$). Secondly, one should note that when a global optimum does exist, there is only one global optimum, but there might be multiple global optimizers.

As it might happen that there are multiple solutions x for which holds that $f(x) = f^*$, an extended goal of global optimization is to find all global minimizers (see [TZ89]):

$$X^* = \{x \in \mathcal{A} \,|\, f(x) = f^*\}. \tag{2.3}$$

This extended goal falls into the second item of the enumeration above.

The alternative goals of finding solutions with objective function values satisfying a certain threshold value (items 4–6 in the enumeration on page 13) are for single objective optimization problems known as *super/sub level set* optimization problems:

Definition 2.1.5 (Sublevel Set)**:** For a single objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and set of feasible solutions $\mathcal{A}$, the *sublevel set* below level d is the set

$$L_d = \{x \in \mathcal{A} \,|\, f(x) \leq d\}. \tag{2.4}$$

2.1.2 Multi-Objective Optimization Problems

A multi-objective optimization problem is a special instance of an optimization problem, defined as

Definition 2.1.6 (Multi-Objective Optimization Problem)**:** A *multi-objective optimization problem* is an optimization problem with more than one objective function. I.e., for the triple

$(\mathcal{X}, \mathcal{F}, \mathcal{G})$, the set $\mathcal{F}$ consists of at least two objective functions.

For multi-objective optimization problems, the definition of optimality is often based on the notion of Pareto dominance on the objective space. Pareto dominance introduces a partial order on the space of objective function values, being $\mathbb{R}^k$ for a problem with k objectives. In the context of minimization, this order is defined as:

Definition 2.1.7 (Pareto Dominance, Weak Pareto Dominance, Strict Pareto Dominance, and Incomparability)**:** For any two vectors $\mathbf{u}$ and $\mathbf{v}$:
$\mathbf{u}$ *dominates* $\mathbf{v}$ (notation $\mathbf{u} \prec_{Pareto} \mathbf{v}$ or just $\mathbf{u} \prec \mathbf{v}$) iff:

$$\forall i \in \{1, \ldots, k\} : u_i \leq v_i \tag{2.5}$$

$$\text{and} \quad \exists j \in \{1, \ldots, k\} : u_j < v_j, \tag{2.6}$$

$\mathbf{u}$ *weakly dominates* $\mathbf{v}$ (notation $\mathbf{u} \preceq \mathbf{v}$) iff:

$$\mathbf{u} \preceq \mathbf{v} \vee \mathbf{u} = \mathbf{v}, \tag{2.7}$$

$\mathbf{u}$ *strictly dominates* $\mathbf{v}$ iff:

$$\forall i \in \{1, \ldots, k\} : u_i < v_i, \tag{2.8}$$

$\mathbf{u}$ and $\mathbf{v}$ are *incomparable* (notation $\mathbf{u} \,||\, \mathbf{v}$) iff:

$$\mathbf{u} \not\preceq \mathbf{v} \wedge \mathbf{v} \not\preceq \mathbf{u}. \tag{2.9}$$

The partial order introduced by using the notion of Pareto dominance on the solution space can be used to define the goal of multi-objective optimization as to find Pareto optimizers:

Definition 2.1.8 (Pareto Optimizer and Pareto Optimum)**:** For a multi-objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with a set of feasible solutions $\mathcal{A}$, the *set of Pareto optimal solutions* X^* is the set of all solutions $x^* \in A$ with function values $\mathbf{f}(x^*) = [f_1(x^*), \ldots, f_k(x^*)]$ for which it holds that there does not exist another solution $x \in \mathcal{A}$ with function values $\mathbf{f}(x) = [f_1(x), \ldots, f_k(x)]$ such that $\mathbf{f}(x)$ dominates $\mathbf{f}(x^*)$. I.e.,

$$X^* = \{x^* \in \mathcal{A} \,|\, \nexists x \in \mathcal{A} : \mathbf{f}(x) \prec \mathbf{f}(x^*)\}. \tag{2.10}$$

An element of the set of Pareto optimal solutions $x^* \in X^*$ is called a *Pareto optimizer* and its objective function value vector is called a *Pareto optimum*.

Although for some multi-objective optimization problems it is sufficient to find a Pareto optimal solution, in general, when a problem is defined as a multi-objective optimization problem, it is intended also to get insight in the trade-offs between the various objectives. Therefore the more usual (customary) aim for multi-objective optimization is to find the set of all Pareto optimal solutions or at least a representative subset of it.

Definition 2.1.9 (Pareto Front and Efficient Set)**:** For a multi-objective optimization problem

$(\mathcal{X}, \mathcal{F}, \mathcal{G})$, the set of all Pareto optima is called the *Pareto Front* and the set of all Pareto optimizers is called the *Efficient Set*.

With the definitions for a single- and multi-objective optimization problem, and the goal to find either one, multiple, or all optimizers, the basic problems of optimization have been introduced. How to solve optimization problems is another matter.

2.1.3 Discrete versus Real-Parameter Optimization Problems

Definition 2.1.1 only generally specifies the search space $\mathcal{X}$, the objective functions $\mathcal{F}$, and the constraint functions $\mathcal{G}$. Besides the separation of single- and multi-objective optimization problems, another distinction can be made by looking at the search space. Although it is not a comprehensive categorization, we distinguish two major classes of optimization problems: *discrete optimization problems* and *real-parameter optimization problems*.

Discrete optimization problems (which is a class that includes *combinatorial optimization problems*) are optimization problems where the search space is a discrete set of candidate solutions.

Definition 2.1.10 (Discrete Optimization Problem)**:** Any optimization problem of which the search space is a discrete set is called a *discrete optimization problem*.

This work focuses on the class of real-parameter optimization problems. Real-parameter optimization problems are optimization problems where the search space is a real-valued parameter space.

Definition 2.1.11 (Real-Parameter Optimization Problem)**:** A *real-parameter optimization problem* is an optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$, with $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{X}$ is of dimension n for some fixed $n \in \mathbb{N}_1$.

More specifically, for real-parameter optimization problems commonly a stricter class definition is taken, requiring the search space to be bounded by a box. We identify such types of problems as box-constrained real-parameter optimization problems.

Definition 2.1.12 (Box-Constrained Real-Parameter Optimization Problem)**:** A *box-constrained real-parameter optimization problem* is a real-parameter optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$, with

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x}_\mathrm{l})_i \leq (\mathbf{x})_i \leq (\mathbf{x}_\mathrm{u})_i, i = 1, \ldots, n\}, \tag{2.11}$$

for some fixed $n \in \mathbb{N}_1$, and where $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$ is a vector of lower bounds and $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$ is a vector of upper bounds.

An example of a class of search spaces that does not fall into the categorization of discrete versus real-parameter is the class of mixed-integer optimization problems, consisting of a combination of real-parameter and discrete (integer and/or categorical) parameters.

2.2 The Practical Goal of Optimization

Given the definition of an optimization problem, the rough classification of optimization goals, the distinction between single- and multi-objective optimization, and the rough classification of discrete versus real-parameter optimization problems, the question is: How to solve such problems? The solvability of optimization problems much depends on the structure of the search space and the basic assumptions about the objective and constraint functions.

Discrete optimization problems with finite (enumerable) search spaces are solvable within a finite number of steps by means of *complete enumeration*. That is, by evaluating every solution in the search space it is possible to determine all optimizers. However, this has practical limits, for example when the search space is sufficiently large and/or evaluations are very time/cost expensive. Given that it is not possible to evaluate all candidate solutions we can follow the reasoning of Bäck [Bäc96] that when a strict subset of the search space is evaluated it is possible that the global optimum of a function f differs arbitrarily much from the optimum found so far. Hence, if we cannot afford to evaluate every solution in the search space, an optimization problem is generally unsolvable.

For real-parameter optimization problems an even more discouraging observation was made by Törn and Žilinskas [TZ89] who in the context of single objective optimization, according to Bäck [Bäc96], proved that

> *"The problem of determining a member of the level set $L_{f^*+\epsilon}$ of an arbitrary global optimization problem on a real-parameter objective function f on a compact feasible region M within a finite number of steps is unsolvable."* [Bäc96, p. 37]

Note that a similar message can be deduced for multi-objective optimization.

Given this observation, we can state that global optimization problems as generally formulated by Definition 2.1.1 on page 12 are practically unsolvable unless we a) restrict ourselves to a class of optimization problems for which the objective functions satisfy certain additional conditions, or b) relax the solvability requirement [TZ89]. Both are done in practice. Regarding the class of problems that is considered, it is commonly assumed that the optimization problem exhibits some kind of underlying structure. An implicit assumption that is often made is that similar solutions are believed to have similar performance. While agreeing that the notion of similarity is not well-defined, neither its intuitiveness nor its validity can be denied. The goal of optimization is relaxed by taking a more practical viewpoint. Based on the definition provided by Törn and Žilinskas [TZ89] we define the general goal of optimization as:

Definition 2.2.1 (Practical Goal of Optimization)**:** Given an optimization problem with an optimization goal and a limited number of resources (i.e., a number of trials or an evaluation

budget), the *practical goal of optimization* is to use these resources in an optimal way to find (an) as good as possible solution(s).

Or, from a slightly different perspective, one could aim for finding solutions that are an improvement with respect to the currently best known solution (*melioration*).

In addition to the practical goal of optimization, it is generally noted that an optimization algorithm is a *global optimization algorithm* if, given an infinite evaluation budget, it will get arbitrarily close to the global optimum.

Definition 2.2.2 (Global Optimization Condition)**:** Let x_t denote the best solution found by the optimization algorithm at time t with function value f_t. We say that the optimization algorithm satisfies the *global optimization condition* iff for $t \to \infty$, $f_t - f^* < \epsilon$, for arbitrarily small positive values of ϵ.

Note that this goal is constructed in terms of finding one global optimizer (provided that it exists) and it should be extended when the goal is to find all (or a particular set of) global optimizers or level set solutions.

2.3 Objective Function Landscapes

A central dogma of geography formulated by Tobler is: "everything is related to everything else, but near things are more related than distant things" [Tob70]. This dogma represents an implicit assumption that is generally also used for optimization problems in practice, namely that there is a correlation between the (dis)similarity of two candidate solutions and the (dis)similarity of their objective function values. The conceptual step from a (dis)similarity measure to a distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is commonly a small one, leading to the assumption of the search space being a metric space $(\mathcal{X}, d)$ (e.g., Euclidean distance for real-parameter search spaces).

The assumption that the search space is a metric space introduces a view on objective functions as landscapes. The search space is the location space of this landscape and the objective function values denote the height or elevation at each location of the landscape. A landscape, consisting of peaks, valleys, ridges, etc., provides a way of visualizing optimization problems, but also allows us to talk about locality related properties of a given objective function. Especially for real-parameter search spaces, this point of view is very intuitive and often used implicitly, with Euclidean distance as distance measure. However, also in other types of search spaces it is possible to view a problem as an objective function landscape.

The simplest class of algorithmic methods that actively exploits correlation between solution similarity and objective function values is the class of the so-called *hill-climbing algorithms*. A hill-climbing algorithm is an iterative algorithm that starts with an arbitrary point in the search space and attempts to find a better solution by evaluating slight perturbations of that solution. If a perturbation produces a better solution, the algorithm proceeds with that new

solution, repeating until no further improvements can be found. Hence, when viewing it from a maximization perspective, it takes steps uphill into the direction of the global optimum. A well-known example of a hill-climbing algorithm (for minimization) is the *Steepest Descent* algorithm (see, e.g., [NW06]) that follows the path of the gradient.

For optimization algorithms that use operators based on local perturbations (such as hill-climbing algorithms, but also Evolutionary Algorithms), the objective function landscape metaphor can be used to visualize different geographical scenarios that have a different impact on the performance of these algorithms. For instance, when an objective function landscape consists of multiple peaks of different heights, a hill-climber can get stuck in one of the lower height peaks, i.e., a locally optimal solution. Interestingly, one can herewith observe that a hill-climbing algorithm does not satisfy the global optimization condition of Definition 2.2.2 and is therefore qualified as a *local optimization algorithm*. This supports the viewpoint of objective function landscapes as a practical way to analyze optimization algorithms.

2.4 Single Objective Real-Parameter Landscapes

For single objective real-parameter optimization problems, the different geological concepts that can influence the behavior of perturbation-based optimization algorithms can formally be defined. Below, the most important concepts will be formalized exactly based on Euclidean distance as dissimilarity measure:

Definition 2.4.1 (Weak Local Minimizer/Optimizer)**:** For a single objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and the set of feasible solutions $\mathcal{A}$, a *weak local minimizer* or *weak local optimizer* is a solution $\mathbf{x}^* \in \mathcal{A}$ for which it holds that

$$\exists \delta \in \mathbb{R}_{>0} \left(\forall \mathbf{x} \in \mathcal{A} \left(||\mathbf{x} - \mathbf{x}^*|| < \delta \Rightarrow f(\mathbf{x}^*) \leq f(\mathbf{x}) \right) \right). \tag{2.12}$$

Definition 2.4.2 (Strict Local Minimizer/Optimizer)**:** For a single objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and the set of feasible solutions $\mathcal{A}$, a *strict local minimizer* or *strict local optimizer* is a solution $\mathbf{x}^* \in \mathcal{A}$ for which it holds that

$$\exists \delta \in \mathbb{R}_{>0} \left(\forall \mathbf{x} \in \mathcal{A} \left(||\mathbf{x} - \mathbf{x}^*|| < \delta \wedge \mathbf{x} \neq \mathbf{x}^* \Rightarrow f(\mathbf{x}^*) < f(\mathbf{x}) \right) \right). \tag{2.13}$$

Definition 2.4.3 (Weak/Strict Local Minimum/Optimum)**:** For a single objective optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and the set of feasible solutions $\mathcal{A}$, a *(weak/strict) local minimum* or *(weak/strict) local optimum* f^* is a value for which there exists a weak/strict local minimizer/optimizer $\mathbf{x}^*$ such that $f^* = f(\mathbf{x}^*)$.

The distinction between weak and strict local minima/optima is subtle, but important. In this work, we will consider a weak local minimum/optimum as the "default" type when referring to a local minimum/optimum. The existence and quantity of local optima is an indicator for the difficulty of a single objective optimization problem (or rather the likelihood of local hill-climbing algorithms to get stuck at local optima). Following Schwefel [Sch77]:

Definition 2.4.4 (Unimodal, Multimodal and Multiglobal)**:** An objective function is said to be *unimodal* if it has only one optimizer (i.e., only a global optimizer). Otherwise, it is said to be *multimodal*. A landscape is called *multiglobal* if there are several global optimizers.

Besides the existence of local optima and global optima, another phenomenon in objective function landscapes is the possible existence of plateaus. A plateau is defined as:

Definition 2.4.5 (Flat Region and Plateau)**:** For a single objective, real-parameter optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{f\}$ and the set of feasible solutions $\mathcal{A}$, a *flat region* of $f|_{\mathcal{A}}$ is a connected set of points $P \subseteq \mathcal{A}$ such that

$$\forall \mathbf{x} \in P \left(f(\mathbf{x}) = c\right) \text{ and } \forall \mathbf{x} \in P \left(\exists \epsilon_{\mathbf{x}} > 0 \left(B_{\epsilon_{\mathbf{x}}}(\mathbf{x}) \cap \mathcal{A} \subseteq P\right)\right), \tag{2.14}$$

for some $c \in \mathbb{R}$, with $B_\epsilon(\mathbf{x}) = \{\mathbf{x}' \in \mathbb{R}^n \,|\, ||\mathbf{x} - \mathbf{x}'|| < \epsilon\}$, and with $\epsilon_{\mathbf{x}}$ defined separately for each $\mathbf{x} \in P$. A *plateau* P is a flat region with the additional property that there exists no flat region $P' \supset P$.

Note that when considering the possible existence of plateaus, there are solutions within a plateau that are both weak local minimizers and weak local maximizers, namely the solutions $\mathbf{x}^* \in \mathcal{A}$ for which it holds that

$$\exists \delta \in \mathbb{R}_{>0} \left(\forall \mathbf{x} \in \mathcal{A} \left(||\mathbf{x} - \mathbf{x}^*|| < \delta \Rightarrow f(\mathbf{x}^*) = f(\mathbf{x})\right)\right). \tag{2.15}$$

This is a somewhat paradoxical property that emerges from using Definition 2.4.1. However, the alternative of restricting to Definition 2.4.2 and changing the $\leq$-sign by the $<$-sign leads to the problem that in a similar case, the global optimum is not a local optimum.

2.5 Black-Box Optimization Algorithms

An optimization algorithm is an algorithmic method that can be applied to solve (a specific class of) optimization problems. There is a wealth of optimization methods available and choosing one for solving a given optimization problem depends much on the characteristics of the optimization problem at hand. There are many aspects that vary from problem to problem. Many optimization methods are especially designed for specific types of search spaces, objective and constraint functions, or tailored for special objective function classes.

This work focuses on optimization methods that are not dependent on any knowledge about the system or model of the optimization problem. That is, the model or system that lies at the core of the optimization problem is considered to be a black-box. For such problems, optimization algorithms are challenged to find good solutions by sequential trial-and-error of candidate solutions. In the strictest sense, the term black-box optimization implies that there is no knowledge about the model or system whatsoever, however, often basic implicit assumptions or properties such as continuity, causality, or even the assumption that the problem belongs to a certain class of problems are used.

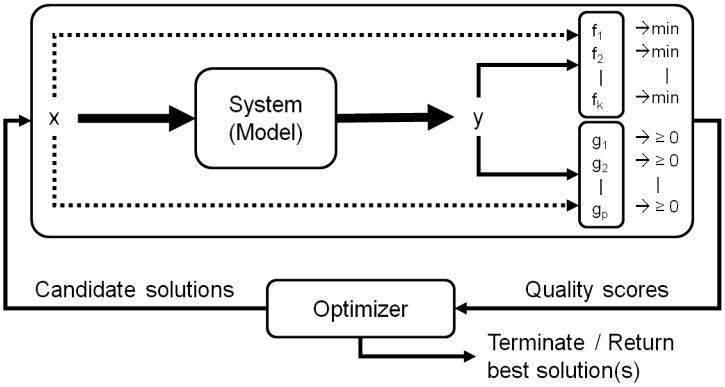


Figure 2.3: The general setup for black-box optimization.

Figure 2.3 visualizes the general black-box optimization loop. The principle is straightforward: An optimizer is coupled to the (model of the) system of interest (according to the model of Section 2.1). The optimizer generates one or more candidate solutions, feeds it/them to the system, and receives a quality score of these candidate solutions. Using this information, the optimizer generates a new set of candidate solutions, and again feeds those to the system to obtain their quality. This loop is repeated until either a satisfactory solution is found, a predefined evaluation budget has been reached, or any other termination criterion has been reached. Note that the term *optimizer* is used in two ways: for optimal solutions and for optimization algorithms. Evolutionary Algorithms form a sub-class of black-box optimization algorithms.

2.5.1 Quality Measures for Optimization Algorithms

A difficult issue in the field of black-box optimization is the assessment of the quality of optimization algorithms. The quality of an optimization algorithm depends much on the characteristics of the problem at hand and the class of optimization problems for which the algorithm is designed. Benchmark sets of test problems are often used for empirical comparison of multiple optimization algorithms, see, e.g., [SHL+05, HFRA09b, HFRA09a, HFRA10]. For these benchmark sets, there are two types of indicators that can be used for determine the quality of the optimization algorithm: The quality of the (set of) solution(s)

1. versus the number of objective function evaluations needed to obtain that quality, or
2. versus the total computation time needed to obtain that quality.

When assuming that the evaluation time of the candidate solutions exceeds the computational overhead introduced by the operations of the optimization algorithm then the former is the most appropriate measure. Moreover, when the optimization algorithms contain stochastic elements, multiple runs should be used for obtaining averaged quality scores.

2.6 Summary and Discussion

In this chapter the background of this work is summarized, being the traditional view on black-box optimization. Given this view on optimization, the two main distinctions between single- and multi-objective optimization problems and between discrete- and real-parameter optimization problems are presented. For such problems, it is shown that there is a distinction between the theoretical goal of optimization and how this goal is used in practice.

Furthermore, the concept of objective function landscapes is summarized, which is based on the assumption that the search space is a metric space and that there is a correlation between (dis)similarity of two solutions and their objective function values. In particular, definitions of commonly used terms are given in the context of single objective real-parameter optimization problems. These types of problems are the main focus of this work.

Black-box optimization algorithms are algorithmic methods that aim to solve optimization problems. Such algorithms sequentially test candidate solutions in a trial-and-error fashion in order to find optimal solutions. Spatial correlation is commonly exploited by optimization algorithms, such as hill-climbing algorithms. Although the quality of an optimization algorithm much depends on the problem at hand, benchmark sets of test problems can be used for empirical comparison of optimization algorithms for certain problem classes.

An issue that is missing in the general model of an optimization problem as described in Section 2.1 is the possible existence of uncertainty and noise. However, these are frequently occurring phenomena when dealing with real-world optimization problems and they can have a large influence on optimization in practice. In Chapter 3, the general model of an optimization model as presented in this chapter will be extended to include uncertainties and/or noise in order to form a definition of a black-box robust optimization problem.

Chapter 3

Robust Optimization

The traditional view on optimization as presented in Chapter 2 does not account for uncertainties and noise. However, this is not realistic for many real-world optimization problems. Consider, for instance, industrial engineering applications. A common optimization scenario is that a non-deterministic simulator replaces the real-world system and the aim is to find solutions such that the real-world realizations of these solutions are of a good quality, also when these are slightly perturbed due to manufacturing errors. In this scenario, uncertainties and noise arise in various ways, e.g., in the form of uncertainty because an approximate model is used instead of the real-world system, in the form of noise because the simulator is non-deterministic, and in the form of uncertainty introduced by the inability to generate exact realizations of the solutions. These observations give rise to two new questions: In what way can uncertainties and noise arise in the general model of an optimization problem as presented in Section 2.1? How do we account for this when optimizing on such systems?

The structure of this chapter is as follows: Section 3.1 starts with an overview/taxonomy of the various ways in which uncertainties and noise can emerge in optimization problems. In Section 3.2, the scope and goals of robust optimization are derived from this taxonomy. Section 3.3 presents three real-world optimization scenarios and discusses them in the context of robust optimization. Section 3.4 closes with a summary and discussion.

3.1 Uncertainties and Noise in Optimization Problems

Often, due to a variety of reasons, the theoretical model used for optimization differs from the real-world system for which optimal solutions are desired. Examples are:

1. The design variables cannot be controlled with unlimited precision. (**input**)
2. The operational (or environmental) conditions fluctuate or are known only to a certain extent. (**model**)
3. The output of the real-world system or of the (simulation) model is noisy. (**output**)

4. Approximation models replace the real-world system within the optimization loop. (**model/output**)

5. There is a degree of vagueness in the objectives and/or the constraint boundaries. (**objectives/constraints**)

These uncertainties can have a negative impact on the practical applicability of using idealized models or assumptions for solving real-world optimization problems. Not accounting for uncertainty and noise might lead to solutions that are found to be optimal with respect to the idealized model, but which are not useful or performing optimally in practice.

For optimization problems, the terms uncertainty and noise refer to behavior in any part of the optimization model that can not (fully) be predicted or controlled, or that is subject to vagueness. As illustrated, the causes of uncertainties and noise in real-world optimization problems can be manifold. In order to find good methods to deal with uncertainties and noise, a first step is to distinguish the different classes of uncertainty and noise that can arise in optimization problems. Among the different ways in which such a classification can be made (see, e.g., [BS07, JB05, ONL06]), the classification provided in this section will, to a great extent, follow the categorization of Beyer and Sendhoff [BS07].

3.1.1 Sources of Uncertainty and Noise

One way to distinguish different types of uncertainty and noise within optimization problems is to categorize them by looking at the parts of the optimization model in which they arise. When considering the model illustrated in Figure 2.2, one can identify five different locations where uncertainty and/or noise can enter the optimization model (see Figure 3.1):

A) Uncertainties and/or noise in the design variables

B) Uncertainties and/or noise in the environmental parameters

C) Uncertainties and/or noise in the output

D) Vagueness in the constraints

E) Preference uncertainty in the objectives

These sources of uncertainty/noise are integrated in different ways in the mathematical formulations of an optimization problem.

A) **Uncertainties and/or noise in the design variables**
With this, deficiencies or fluctuations in the real-world realizations of candidate solutions are addressed. These deficiencies/fluctuations can arise when in the real-world system, the design variables can only be realized or controlled with a limited precision. Manufacturing imprecisions in product engineering are exemplary for this type of uncertainty,

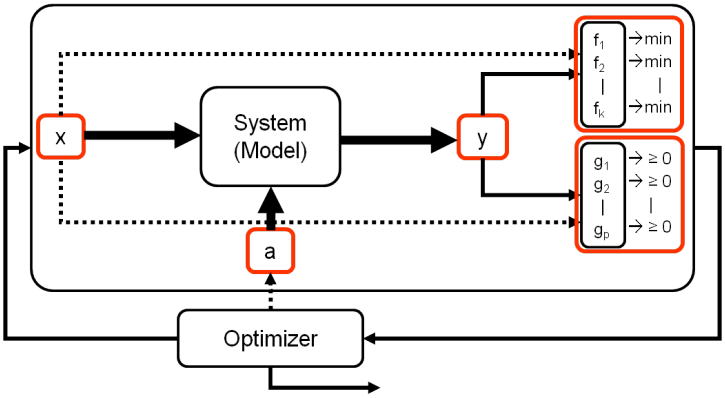


Figure 3.1: The general optimization model showing the different parts of the systems in which uncertainties can arise.

which is the main type of uncertainty considered by the robust design concept. This type of noise gives rise to two different scenarios:

Scenario 1: A simulation model replaces the real-world system within the optimization model. In this case the simulator accepts inputs as non-disturbed inputs, though in practice, solutions cannot be realized with unlimited precision. This can be compensated for by focusing the optimization on finding solutions that are also of high quality when slightly perturbed (i.e., finding robust optima).

Scenario 2: The real-world system is enclosed in the optimization loop (*experimental optimization*). In this case the disturbances in the input propagate through the output, hence to the objective and constraint functions. When the disturbances in the design variables can be measured a posteriori, the optimizer will receive deterministic measurements, but the sampling process cannot be controlled. In case that the disturbances cannot be measured a posteriori, these disturbances simply yield noise in the objective and constraint functions. The noise distributions can be of any kind depending on the distribution of the noise in the design variables, transformed through the output and the objective and constraint functions.

The effect of variations caused by uncertainties or noise in the design variables can be modeled within the formulation of an objective functions as:

$$\tilde{f}_i(x) = f_i(x + \delta_x), \quad i = 1, \ldots, k. \tag{3.1}$$

And similarly for the constraint functions:

$$\tilde{g}_j(x) = g_j(x + \delta_x), \quad j = 1, \ldots, p. \tag{3.2}$$

Here, $\delta_x \in \mathcal{X}$ is an uncontrollable random or uncertain variable representing the deviations/fluctuations in the input that are due to uncertainty and/or noise. An important

observation is that this noise or uncertainty can depend on the values of the design variables themselves, i.e., it can vary for different x. Furthermore, due to this remodeling, the outputs of the objective and constraint functions become random variables (in case the uncertainty δ_x can be modeled as a random variable) or sets of possible outputs.

B) **Uncertainties and/or noise in the environmental parameters**
Uncontrollable (environmental) parameters of a system are not considered in the classical optimization model as presented in Section 2.1 (see Figure 2.2). These parameters are in the classical view considered to be constants of the system. However, in practice many of such constants are noisy or uncertain. Fluctuating or unknown operating conditions and deficiencies of internal parameters are possible ways in which uncertainties can manifest themselves within this part of the optimization model. Similar to uncertainties in the design variables, the setting in which this type of uncertainty can be actively compensated for is when a simulation model replaces the real-world system within the optimization model. Otherwise, the uncertainty propagates to the objective and constraint functions.

To incorporate such scenarios into the model presented in Section 2.1, we extend it as follows: Let the set $\mathcal{C}$ denote the (possible) settings of uncontrollable environmental parameters of the system. The models of the objective and constraint functions are extended by also being a function of the environmental parameters, i.e., $f : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$ for all $f \in \mathcal{F}$ and $g : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$ for all $g \in \mathcal{G}$.

The effect of variations caused by uncertainties or noise in the environmental parameters can be modeled in the objective functions as:

$$\tilde{f}_i(x, a) = f_i(x, \alpha), \quad i = 1, \ldots, k. \tag{3.3}$$

And similarly for the constraint functions:

$$\tilde{g}_j(x, a) = g_j(x, \alpha), \quad j = 1, \ldots, p. \tag{3.4}$$

Here $\alpha \in \mathcal{C}$ is a noisy or uncertain counterpart of the constant $a \in \mathcal{C}$. As can be seen, by modeling the uncertainty/noise in this way, the constant environmental parameters a are replaced by random or uncertain variables α.

C) **Uncertainties and/or noise in the output**
The third class is formed by output uncertainties or noise. Here we can distinguish two different types: 1) The system is non-deterministic and produces noisy outputs (e.g., the measurements have a stochastic nature and precise evaluation is impossible). 2) The system produces uncertain output (e.g., models where the quality of the output of the model cannot be guaranteed to be conform the actual output of the system).

Note that both types could be present simultaneously, e.g., when using non-deterministic simulation models.

Noise or uncertainty in the output can be modeled within formulations of the objective functions as:

$$\tilde{f}_i(x) = f_i(x) + z_{f_i}(x), \quad i = 1, \ldots, k, \tag{3.5}$$

and in the constraint function definitions as:

$$\tilde{g}_j(x) = g_j(x) + z_{g_j}(x), \quad j = 1, \ldots, p, \tag{3.6}$$

with $z_{f_i}(x)$ and $z_{g_j}(x)$ being random/uncertain variables (possibly indexed by space), denoting the propagation of the output uncertainty/noise to the objective and constraint functions.

Note that, strictly seen, the noise in the objective and constraint functions is a product of the noise in the output. Internally, $z_{f_i}(x)$ is a (possibly non-linear) function of the noise in the output, i.e., $y(x) + z_y(x)$. However, as we have chosen to model the objective functions and constraint functions in terms of x, this cannot be modeled explicitly.

Furthermore, note that uncertainties in the design variables and environmental parameters in principle propagate as a noisy or uncertain output, i.e., fluctuating/uncertain design variables and environmental parameters yield fluctuating/uncertain system output.

D) **Vagueness in the constraints**

Often it is hard to obtain strict and bounded definitions of constraints. When dealing with constraints like "the temperature should not be too high" or "the risk should not be too high", it is not possible (or desirable) to draw a straight line between satisfied and not-satisfied. Such vagueness calls for methods that can account for this (e.g., by using fuzzy logic techniques [Zad65, BZ70]).

Fuzzy constraints can be described using the following notation:

$$g_j(x) \gtrsim 0. \tag{3.7}$$

Here, $\gtrsim$ is the fuzzified version of $\geq$ having the linguistic interpretation "$g_j(x)$ is essentially greater than 0". However, this notation does not take into account the degree to which the constraint is fuzzy. For example, considering the "temperature should not be too high" example, the notation of Eq. 3.7 does not specify the margins for which the temperature could still be acceptable.

A typical way of modeling uncertainties of this kind is to transform the constraints by means of membership functions. A membership function is a monotonous (but not necessarily linear) function V_g which maps a constraint function $g(\mathbf{x})$ to an area bounded by 0 (strictly violated) and 1 (completely satisfied). Everything between 0 and 1 is in the

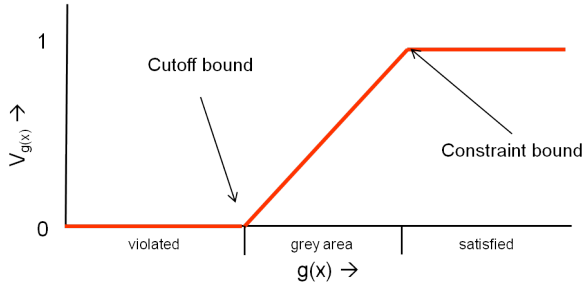


Figure 3.2: A simple linear membership function.

grey area (or transition area), and the higher the value of the membership function, the higher the degree of satisfaction. Figure 3.2 illustrates the basic idea of a membership function. Mathematically, this transformation has the following form:

$$g(x) \geq 0 \quad \text{becomes} \quad V_g(g(\mathbf{x})) \rightarrow \max . \tag{3.8}$$

A design issue that is introduced is that appropriate membership functions have to be constructed.

E) **Preference uncertainty in the objectives**
Preference uncertainty emerges when having multiple conflicting objectives, hence, when highly subjective trade-off choices have to be made regarding the importance of objectives. Preference might only be known afterwards, depending on the quality level that can be achieved for the objectives and the particularities of the trade-offs that exist. Approaches that use aggregate objective functions which combine multiple objectives functions into one objective function are prone to such uncertainties. Approaches that aim to approximate the complete Pareto frontier postpone such preference decisions.

Of these five sources of uncertainty, a distinction can be made between the first three (A, B, and C) and the last two (D and E). The former involve *uncertainty and/or noise in the system or (simulation) model*, whereas the latter involve *uncertainty in the optimization model* (i.e., the way in which a given problem is translated to an optimization problem).

3.1.2 Modeling Uncertainty and Noise

Besides the fact that uncertainties can arise in different parts of the optimization model, another way to distinguish different kinds of uncertainty and noise is to consider their nature. Up to now, a distinction has been made between uncertainty and noise, as being two distinct matters. The question is, however, whether these are indeed separate issues.

The distinction between uncertainty and noise is the same as the distinction between *aleatory uncertainty* and *epistemic uncertainty* which is a particularly popular view in the engineering field (e.g., [PC96, KD09]). Also, it can be related to the distinction between the two schools

of statistical theory: frequentist and Bayesian statistics [O'H04, Cox05]. The term aleatory uncertainty is used to describe uncertainties within a system or model that have an intrinsic stochastic nature. These are the uncertainties associated with the pure (often said to be irreducible) randomness within a system. Epistemic uncertainty, on the other hand, is the uncertainty that is due to incomplete or inadequate information (i.e., due to a lack of knowledge). Epistemic uncertainty should be reducible when more knowledge becomes available. Regarding the two schools of statistics, the frequentists can be said to accept uncertainty as aleatory, whereas in the Bayesian statistics the focus is on the *degree of belief*, which can be related to epistemic uncertainty [O'H04].

Although the distinction between aleatory and epistemic uncertainty intuitively makes sense, it is often a source of confusion (see, e.g., [KD09] for a discussion). It is often hard to specify whether a specific uncertainty is purely due to inherent randomness or due to limited knowledge or modeling capabilities. A purely deterministic mind would attribute every uncertainty to limited knowledge, and indeed in practice many cases of uncertainty are due to abstractions of details of the system. Similarly, the distinction between frequentist and Bayesian statistics seems to touch upon the same subject. Here, the debate evolves around the difference between *uncertainty of knowledge* versus *variability in outcome* [Cox05].

To avoid resorting to a lengthy philosophical discussion on this topic, we will make a distinction based on the difference in the mathematical modeling of the uncertainties. This is in line with the view presented by Beyer and Sendhoff [BS07]. In fact, one could say that the decision on the type of uncertainty is left to the person providing the optimization model. Hence, from the perspective of optimization, we are not so much interested in the actual type of uncertainty, but rather in the way in which it is modeled within the optimization problem formulation. Looking at the different ways in which uncertainties and noise can mathematically be modeled within an optimization model, one can distinguish three classes:

1) **Fuzzy**
   The uncertainty is formulated by fuzzy statements about the possibility or degree of membership by which a state of an uncertain variable is believed to be plausible (or desirable). Uncertainties of this type can be modeled with *fuzzy sets* [Zad65, BZ70]. Using fuzzy sets for this type of uncertainty requires modeling a particular uncertain variable from a given space of points A as a pair (A, m_A). Here, $m_A : A \to [0, 1]$ is a *membership function* that maps each $x \in A$ to a "grade of membership" in (A, m_A). The grade of membership is a value between 0 and 1. For a particular $x \in A$, the closer $m_A(x)$ will be to one, the higher its degree of membership of A or degree of plausibility.

2) **Deterministic**
   The uncertainty is formulated by statements about the crisp possibility of whether a state of an uncertain variable is possible or not. Uncertainties of this type can be modeled

using *crisp sets*. A particular uncertain variable of this type is modeled as a pair (A, m_A), with A being the crisp set and m_A denoting the membership function. The membership function is of the form $m_A : A \to \{0, 1\}$. Hence, a particular $x \in A$ can be noted to be either a member of the set A, when $m_A(x) = 1$, or not to be a member of A, when $m_A(x) = 0$.

3) **Probabilistic**
The uncertain variable is believed to be a stochastic random variable. A probability measure can be established measuring the probabilistic frequency of events that may occur. Uncertainties of the probabilistic type can be modeled using probability functions (or probability density functions in case of continuous domains). In this case, a function $p : A \to \mathbb{R}_{\geq 0}$ maps every event x in the space of all possible events A to a probability (density) value denoting the probability of that particular event. Note that the function p should conform to the classical Kolmogorov axiom system [Kol33].

From the perspective of mathematical modeling, this division can also be seen as a hierarchical structure of increased knowledge of the uncertainty. In the first type, there is uncertainty about the domain of the variation and the probabilities of the uncertainty events. In the deterministic type of uncertainty, the domain of the variation is known, but there is no knowledge about the probabilities of the events. In the probabilistic type of uncertainty, both the domain and the probabilities of the individual variation events are known. Although this distinction might be subtle (and the mathematical formulations might seem very similar), this difference is of great importance, as it requires different methods for treating these types of uncertainty.

Returning to the distinction between aleatory and epistemic uncertainty (and also the distinction between uncertainty and noise), we can see that aleatory uncertainty is the uncertainty of the probabilistic type and epistemic uncertainty is the uncertainty that is either of the deterministic or possibilistic type. A schematic view is given in Table 3.1. Moreover, when adopting this view, we can say that (agreeing with [KD09]) the aleatory/epistemic distinction is made by the person modeling the optimization problem and it is fair to make a distinction between these two types of uncertainty within the scope of the optimization model.

3.1.3 Stationary versus Non-Stationary Noise

One issue that has not yet been covered in the discussion about the modeling of noise is the distinction between *stationary noise* and *non-stationary noise*. Traditionally, these terms are used within the context of time-based systems in which the output is noisy. The term non-stationary noise indicates that the noise distribution changes over the course of time, whereas the term stationary noise indicates that the noise distribution is independent of time.

For optimization problems, this notion does not restrict to time, but can also be used in the context of space, i.e., non-stationary uncertainty/noise differs among the candidate solutions.

Conceptual classification	Mathematical modeling	Mathematical properties
Epistemic (uncertainty)	Possibilistic	Uncertainty domain unknown Probabilities unknown
	Deterministic	Uncertainty domain known Probabilities unknown
Aleatory (noise)	Probabilistic	Uncertainty domain known Probabilities known

Table 3.1: A classification of the types of uncertainty in terms of their mathematical properties based on the conceptual distinction between epistemic and aleatory uncertainty.

Looking back at the descriptions of the sources of uncertainty one notes that for uncertainties of type A and C, the noise/uncertainty might vary for different values of $x \in \mathcal{X}$. If the noise/uncertainty varies for different values of x, we call the noise/uncertainty *non-stationary*, if the noise/uncertainty is the same for all $x \in \mathcal{X}$ it is called *stationary*.

3.1.4 Cases of Uncertainty and Noise

Five sources combined with three types of uncertainties/noise yields theoretically 15 different concepts of uncertainty/noise that can be encountered within optimization problems. When including the distinction between stationary and non-stationary uncertainty/noise and the consideration that multiple types of uncertainties/noise can be present simultaneously within a real-world optimization problem, this picture becomes even more discouraging. Indeed, the variety of cases of uncertainty/noise that can arise from the combinations of the different cases complies with the complexity of real-world scenarios. In practice, however, some scenarios occur more often than others, and those are worthwhile studying in relative isolation (i.e., one type of noise within one part of the optimization model). Table 3.2 summarizes the combinations of class/type of uncertainty/noise as they can occur within optimization problems.

Uncertainty in the design variables (class A) is often uncertainty of type 3, i.e., probabilistic uncertainty. As this source of uncertainty is based on scenarios in which in the real-world the design variables cannot be set infinitely accurate, its relation to aleatory uncertainty with respect to the optimization model is undeniable. Hence, these are cases in which it is well doable and reasonable to describe the uncertainty using probability distributions. In cases where the uncertainty in the design variables is modeled as type 1 or type 2 uncertainty, these descriptions should be strict in order for optimization to make sense at all (i.e., having a broad uncertainty range for each candidate solution basically makes the solutions incomparable, hence not worth optimizing). Both stationary and non-stationary noise are possible for this uncertainty class. Hence, there are scenarios in which the fluctuations of the design variables depend on the settings themselves, but it may also be that the fluctuations have the same

Class	Type	Stationarity / non-stationary
A) Uncertainties and/or noise in the design variables	1) Possibilistic 2) Deterministic **3) Probabilistic**	Stationary or non-stationary
B) Uncertainties and/or noise in the environmental parameters	**1) Possibilistic** **2) Deterministic** **3) Probabilistic**	Stationary
C) Uncertainties and/or noise in the output	**1) Possibilistic** **2) Deterministic** **3) Probabilistic**	Stationary or non-stationary
D) Vagueness in the constraints	1) Possibilistic 2) Deterministic ~~3) Probabilistic~~	Stationary
E) Preference uncertainty in the objectives	1) Possibilistic 2) Deterministic ~~3) Probabilistic~~	Stationary

Table 3.2: Classification and categorization of different manifestations of uncertainty/noise as they can occur within optimization problems. Note that the bold types of uncertainty are considered to be "more common" and the crossed-out types are consdered to be nonexistent.

characteristics for all candidate solutions.

In class B, all three types can be encountered. Uncertainties of probabilistic nature occur when system parameters are due to fluctuations. Possibilistic and deterministic uncertainty occur when model parameters that represent real-world parameters are unknown. Class B uncertainties are always stationary with respect to the given design variables. That is, being fluctuations in the operation conditions of the system, they cannot depend on the particular settings of the design variables.

In class C, also all types of nature of uncertainty can be encountered. When dealing with noisy output, this uncertainty is of type 3. When the uncertainty arises due to the use of uncertain prediction models, the uncertainty is of type 1 or type 2. Moreover, when using stochastic simulation models, the uncertainty can be a composite of type 3 and type 1/2. Also this type of uncertainty can be both stationary and non-stationary with respect to the design variables.

In class D and class E, the nature of the uncertainties is always described as type 1 or 2 uncertainty. These are classes of uncertainty that are somewhat different than the other classes. These classes do not regard uncertainty within the model or the system, but rather the uncertainty of specifying what is desirable or acceptable. These types of uncertainty are only stationary with respect to the design variables.

3.2 Robust Optimization

Up to now, we have discussed the various ways in which uncertainty and noise can present themselves within optimization problems. Given this taxonomy, the next question is: what is robust optimization? In Section 1.1 it is noted that there exist different views on this matter. Some consider robust optimization to involve input uncertainties and/or noise in the design variables (e.g., [BBC11, BNT10]), whereas others consider a broader view, considering robustness with respect to uncertainties within the system or (simulation) model (e.g., [BS07]).

In this work, we consider robust optimization to be the practice of optimization given uncertainties and/or noise in the system or (simulation) model. Note that we exclude modeling uncertainty. For any kind of uncertainty and/or noise of class A, B, or C, robust optimization deals with the questions:

- In what way do uncertainties and/or noise within the system or (simulation) model affect optimization algorithms and the practical applicability of solutions found by optimization algorithms?
- How should optimization algorithms be adapted in order to account for uncertainties and/or noise within the system or (simulation) model?

With this we can extend the practical goal of optimization of Definition 2.2.1 and formulate the general goal of robust optimization as:

Definition 3.2.1 (Practical Goal of Robust Optimization)**:** Given an optimization problem with uncertainty and/or noise within the system or (simulation) model, and given an optimization goal and a limited number of resources. The *practical goal of robust optimization* is to use these resources to find (an) as good as possible solution(s) despite uncertainty and/or noise, that are also optimal and useful in the face of the uncertainties/noise.

Note here that optimality with respect to the uncertainties and/or noise should be defined within the scope of the uncertainty and/or noise at hand. Besides that, from this formal definition two intrinsically different targets of robust optimization can be distinguished, related to uncertainty and noise within optimization problems:

1. Target to find optimal solutions in noisy/uncertain environments.
2. Target to find robust solutions.

The first aim is to deal with the fact that the system on which the optimization is performed is not guaranteed to be noise/uncertainty-free and the problem is to design an optimization method that is able to deal with this in order to find solutions which are of good quality (also in practice). The other aim represents the desire to find solutions that are also of good quality when variations in the design variables or environmental parameters occur. This aim

Robust Optimization Target	Uncertainty class
Finding robust solutions	A) Uncertainties and/or noise in the design variables B) Uncertainties and/or noise in the environmental parameters
Optimization in noisy and uncertain environments	C) Uncertainties and/or noise in the output

Table 3.3: The two targets of optimization can roughly be related to the place in which the uncertainty/noise emerges.

emerges when instead of the real-world system, models are used for optimization and candidate solutions or internal model parameters cannot be guaranteed to be set infinitely precise in the real-world system. Hence, it focuses on the robustness of the solutions themselves. Table 3.3 summarizes this global categorization of robust optimization problems.

In literature often isolated scenarios for robust optimization are considered. That is, isolated in the sense that only one particular type of noise/uncertainty, present in one particular part of the system or (simulation) model is considered. The most prominent scenarios are:

- Optimization of noisy objective functions (e.g., [FG88, Bey00])
- Optimization of uncertain objective functions (e.g., [KAE+09])
- Optimization on systems in which the design variables are affected by uncontrollable and unmeasurable perturbations (e.g., [BOS03])
- Finding robust optima in anticipation of perturbations of the design variables (e.g., [TG97, GA05, ONL06, PBJ06])
- Finding robust optima in anticipation of fluctuations of the environmental parameters (e.g., [Hop09])
- Finding robust optima in anticipation of different operation conditions of the environmental parameters (e.g., [RKD+11])

In the remainder of this work, we will also consider such isolated scenarios. In particular, we consider two scenarios as exemplary for robust optimization, optimization of noisy objective functions and finding robust optima in anticipation of perturbations of the design variables, and study the way in which to solve such robust optimization problems. However, the reader should be aware of the fact that the practice of robust optimization comprises a broad variety of scenarios of uncertainties and/or noise.

3.3 Real-World Robust Optimization Scenarios

This section is devoted to present three real-world optimization scenarios in which uncertainty and noise are inherent parts of the system of interest. These scenarios are related to the view on robust optimization as presented in this chapter.

3.3.1 Deep Drawing Optimization

The optimization of the design of a deep drawing process for sheet metal forming in engineering is a typical example of a robust optimization problem (see, e.g., [SH04, PLBG07]). The design of a deep drawing process involves finding proper settings for the process (e.g., drawing forces), such that the end product of the deep drawing process is of the desired geometrical shape that complies to, or is as good as possible with respect to properties relating to plasticity, thickness, and probability of forming failure. In practice, finite element software is often used for virtual testing of candidate designs. Hence, the design of a deep drawing process is an optimization problem for which automated optimization techniques are in principle well-suited. Uncertainty and noise arise in several ways in such optimization problems:

A) **Uncertainties and/or noise in the design variables:**
In the real-world manufacturing process, the drawing forces, such as the drawbead forces, the binder force, and the punch force cannot be set infinitely accurately (type 3). Designs should be robust against fluctuations in these variables.

B) **Uncertainties and/or noise in the environmental parameters:**
Parameters within the simulation model are uncertain and/or noisy, e.g., the friction coefficient (type 1/2) or the blank thickness (type 3). Designs should be robust against these uncertainties and fluctuations.

C) **Uncertainties and/or noise in the output:**
Due to the limited accuracy of a (finite element) simulator, the simulation output is not guaranteed to match real-world behavior exactly (type 1/2). The optimization algorithm should be robust against this type of uncertainty and find good solutions despite the difficulty in the quality assessment of candidate designs.

The robust optimization goals for such problems are to find optimal designs (or solutions) that are robust against fluctuations and uncertain conditions in real-world practice, and furthermore to assure robustness of the optimization process itself, that has to deal with the uncertainty in the assessment of the quality of candidate solutions.

3.3.2 Building Performance Optimization

With increasing quality of Building Performance Simulation (BPS) tools, including optimization approaches for finding optimal building designs becomes more and more viable

[HHPW07, EHM+08, Hop09, HTHB11]. Consider, for example, the problem of optimizing building designs for thermal comfort (maximization) and energy consumption (minimization) with respect to the following design variables: infiltration rate (air exchange rate per hour in the building), window fraction (the amount of glass percentage on one wall of the building), load equipment (power equipment per net floor surface), and load lighting (power lighting per net floor surface). By using BSP tools for the evaluation of candidate solutions (i.e., alternative designs), automated optimization can be used to optimize building designs. The ways in which uncertainty and noise arise in such optimization problems are:

B) **Uncertainties and/or noise in the environmental parameters:**
The operation conditions of a building in the real-world are uncertain and fluctuating. An obvious example is the outside temperature that changes from hour to hour and from day to day. For successful integration of optimization in this context, candidate designs should be evaluated with respect to these fluctuating environmental operating conditions.

C) **Uncertainties and/or noise in the output:**
BPS tools are not guaranteed to match the real-world behavior exactly (type 1/2). The optimization algorithm should be able to find high quality designs despite the difficulty in the quality assessment.

Moreover, although we do not consider them in the scope of robust optimization, also uncertainties of class E emerge in such problems. That is, thermal comfort is a typical objective in which preference uncertainty arises as this is an inherently subjective objective, these indicators are fuzzy measures.

3.3.3 Molecular Design Optimization

In *de novo* design of molecular structures, the challenge is to find molecular structures that could be active components of drugs. In order for a molecular structure (or ligand) to be suitable as a drug component, a number of criteria should be met. First, it should be active on the targeted receptor. Second, it should fulfill a number of criteria such that it is actually taken in by the body, it is not toxic or harmful in other ways, and it should be possible to actually create (synthesize) the structure (preferably as easily as possible).

In practice, automated methods can be used for *in silico* design of candidate molecular structures (see, e.g., [NAP09, KBIvdH08, KAE+09]). A simple setup, for instance, is to use docking simulations to predict the binding affinity of a candidate ligand to a receptor, and simple descriptors (such as Lipinski's rule of five [LLDF01]) to determine the likelihood of a candidate ligand to be generally suitable as a drug. Given this, the design task (or optimization task) is to find molecular structures that have a high docking score, and also score well on the simple descriptors (hence, it is a multi-objective optimization problem). The ways in which uncertainty and noise arise in such optimization problems are:

C) **Uncertainties and/or noise in the output:**
The simulation output of the docking simulator is an approximation of the real-world behavior (type 1/2). Due to the use of a stochastic simulator, the outputs of the simulator are noisy (type 3). The optimization algorithm should be robust against these types of uncertainties and noise and find good solutions despite the difficulty in the quality assessment of candidate molecular structures.

Moreover, although we do not note them as being in the scope of robust optimization, also uncertainties of class D and class E emerge in such problems. Vagueness in the constraints can emerge when constraint bounds are used for the simple descriptors that determine the likelihood of a candidate ligand to be generally suitable as a drug (see, e.g., [KEB+09b]). Preference uncertainty can emerge when the objectives are combined into one aggregate scoring function.

3.4 Summary and Discussion

In this chapter we have extended the model of optimization problems as presented in Chapter 2 to a model that includes uncertainties and noise as they can arise in real-world optimization problems. It has been shown that the various sources and types of uncertainty and noise yield a combinatorial explosion of different scenarios. Yet, a few isolated scenarios can be identified that emerge frequently, hence, are worthwhile subjects of study.

In this work, we consider robust optimization as the practice of optimization given uncertainties and/or noise in the system or (simulation) model. Hence, we exclude uncertainties that arise from the modeling of an optimization problem. Robust optimization deals with two goals: 1) the aim to find optimal solutions in noisy/uncertain environments, and 2) the aim to find robust solutions.

The concepts introduced in this chapter were exemplified by means of three real-world optimization scenarios; deep drawing optimization, building performance optimization, and molecular design optimization. For these three scenarios it was illustrated how the various forms of uncertainties/noise enter the optimization model, therewith stressing the importance of robust optimization.

In the second part of this work we will focus on two scenarios of robust optimization for real-parameter optimization problems: optimization of noisy objective functions and the problem of finding robust optima. For these two scenarios we will study how Evolutionary Algorithms, and in particular Evolution Strategies, should be adapted in order to deal with robust optimization problems.

Part II

Evolution Strategies for Robust Optimization

Chapter 4

Evolutionary Algorithms and Evolution Strategies

The paradigm of evolutionary computation is derived from the model of organic evolution and refers to the application of the Darwinian principles of evolution for computational purposes. The term *Evolutionary Algorithm* refers to an algorithmic method that adopts the paradigm of evolutionary computation for solving optimization problems. Within these methods, a *population* of candidate solutions (*individuals*) repeatedly undergoes the processes of reproduction and selection, with the *fitness* of each candidate solution being expressed in terms of its quality with respect to the given optimization problem. Hence, the basic elements of natural evolution are used to breed populations of (near-)optimal solutions.

The main components of an Evolutionary Algorithm are summarized in Section 4.1. For a thorough introduction to Evolutionary Algorithms, the reader is referred to, e.g., [Bäc96] and [ES03]. Section 4.2 focuses on Evolution Strategies and introduces the two algorithmic techniques that are the main focus of this work; the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. Section 4.3 closes with a summary and discussion.

4.1 Evolutionary Algorithms

Figure 4.1 depicts the general evolution cycle of an Evolutionary Algorithm as a flowchart. This flowchart is one of the possible variants. It considers a population of candidate solutions or individuals that, after being initialized in some fashion, enters an evolution loop. A subset of this population is selected (based on fitness) as *parents* for creating the *offspring* of the next *generation*. The parents are then used for generating a set of offspring by *recombination* of two or more of the parents and *mutation* of the recombined individual. Thereafter, the population of the next generation is selected either from the offspring and the parents (*elitism*) or solely from the offspring. This loop is repeated until a termination criterion is met.

Representation: Each individual represents a candidate solution for the optimization problem at hand and should be modeled in some appropriate/convenient way in order to be used within

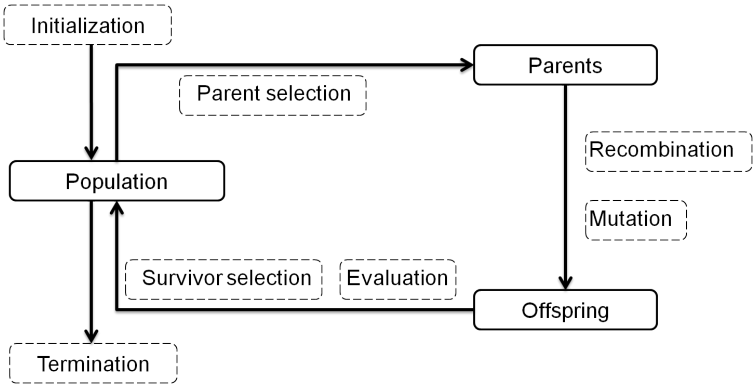


Figure 4.1: The iteration cycle of an Evolutionary Algorithm.

the context of Evolutionary Algorithms. For some problems, the representation follows naturally from the description of the search space. For other problems, an abstract representation has to be designed, making a genotype/phenotype distinction. In the latter case, also a genotype to phenotype decoding mechanism has to be designed. Two key points in choosing an appropriate representation are: 1) It should be possible for each element of the search space to be modeled (preferable uniquely) according to the chosen representation. 2) The representation should allow for the implementation of genetic operators such as mutation and recombination.

Initialization: The initial population can be based on known good solutions that serve as starting point for the evolutionary search, or can be generated randomly. When generating the initial population randomly, it should be well-spread over the search space, which yields a higher chance of identifying promising regions in the search space. For most representations, the initialization procedure is fairly straightforward (e.g., using Latin Hypercube sampling in real-parameter search spaces).

Evaluation: The evaluation component uses the objective and constraint functions of the optimization problem to assign a quality score or fitness to each individual. Individuals with a higher fitness will have a higher probability of surviving and passing on their genetic material (i.e., the candidate solution that they represent) to future generations.

Selection: There are two types of selection: *parental selection* (or *mating selection*) and *survivor selection* (or *environmental selection*). Parental selection is a stochastic selection type that selects the parents that are used for the recombination of a new offspring (i.e., for the generation of each offspring). In this selection type, the fitter individuals have a higher probability to be selected as parent for recombination. The latter, survivor selection, is a deterministic selection type that is applied at an earlier stage. It selects the μ fittest individuals (the survivors) either out of the λ offspring, or, if *elitism* is used, out of the λ offspring and the μ old parents. This selection type is commonly referred to as $(\mu^{+}_{,}\lambda)$-selection, with $(\mu{+}\lambda)$ denoting elitist selection, and (μ, λ) denoting non-elitist selection.

Mutation and recombination: Mutation and recombination are the two types of *genetic operators* or *variation operators*. Mutation operators add small perturbations to the (genotype representations of the) individuals in the population. Recombination operators recombine two or more individuals in the population into a new individual by means of crossover (of the chromosomes). The choice of the genetic operators depends on the chosen representation of individuals. An important criterion for the genetic operators is that it should be possible to get from any solution in the search space to any other solution in the search space by applying the genetic operators a finite number of times. Besides that, the genetic operators should be well-defined, such that applying the operators to any solution (set of solutions) yields a valid solution that is also in the search space. Lastly, the genetic operators should be unbiased.

Termination: The termination condition can depend on the available computation time, the available number of evaluations/generations, or on convergence criteria such as a pre-defined target fitness that is to be reached. After termination, the best solution(s) of the final population or the best solution(s) found throughout the evolution loop can be considered as (a) good solution(s) for the optimization problem. That is, provided the evolution process is granted enough time.

4.2 Evolution Strategies

Evolution Strategies form a special branch of Evolutionary Algorithms, specifically designed for real-parameter optimization problems as described by Definition 2.1.11 on page 16. They have been proposed originally by Rechenberg [Rec73] and Schwefel [Sch77]. Over the years, the class of Evolution Strategies has broadened into many algorithmic variants, making it hard to pin down the canonical Evolution Strategy. In this section we will briefly summarize the basic concepts and the most important Evolution Strategy variants with their respective algorithmic descriptions.

4.2.1 The $(1+1)$-Evolution Strategy

The simplest Evolution Strategy is the $(1+1)$-Evolution Strategy (one parent, one offspring), presented by Rechenberg [Rec73]. Algorithm 4.1 describes the working mechanism of this simple Evolution Strategy.

The $(1+1)$-ES starts with a random solution $\mathbf{x}_\mathrm{p}$ drawn uniform randomly from the search space $[\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u}]$ (denoted $\mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$). This solution, the parent individual, is evaluated and the algorithm enters the evolution cycle. In each generation, one offspring $\mathbf{x}_\mathrm{o}$ is created from the parent by adding a small random vector to a copy of the parent. The perturbation, or mutation, is drawn from an uncorrelated multivariate Gaussian distribution:

$$\mathbf{x}_\mathrm{o} = \mathbf{x}_\mathrm{p} + \sigma\mathbf{z} \ , \ \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.1}$$

Algorithm 4.1: The $(1+1)$-Evolution Strategy

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{p}$ with objective function value f_p

Set parameters: $c \leftarrow 0.85$, $G \leftarrow \max(5, n)$

Initialize: $g \leftarrow 0$, $G_\mathrm{s} \leftarrow 0$, $\sigma \leftarrow \frac{||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||}{3\sqrt{n}}$, $\mathbf{x}_\mathrm{p} \leftarrow \mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$, $f_\mathrm{p} \leftarrow f(\mathbf{x}_\mathrm{p})$

while not terminate **do**

  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

  $\mathbf{x}_\mathrm{o} \leftarrow \mathbf{x}_\mathrm{p} + \sigma\mathbf{z}$

  $f_\mathrm{o} \leftarrow f(\mathbf{x}_\mathrm{o})$

  if $f_\mathrm{o} \leq f_\mathrm{p}$ **then**

    $\mathbf{x}_\mathrm{p} \leftarrow \mathbf{x}_\mathrm{o}$

    $f_\mathrm{p} \leftarrow f_\mathrm{o}$

    $G_\mathrm{s} \leftarrow G_\mathrm{s} + 1$

  end if

  $g \leftarrow g + 1$

  if $g \bmod G = 0$ **then**

    $p_\mathrm{s} \leftarrow G_\mathrm{s}/G$

    $$\sigma \leftarrow \begin{cases} \sigma/c & , \text{if } p_\mathrm{s} > 1/5 \\ \sigma \cdot c & , \text{if } p_\mathrm{s} < 1/5 \\ \sigma & , \text{otherwise} \end{cases}$$

    $G_\mathrm{s} \leftarrow 0$

  end if

end while

return $(\mathbf{x}_\mathrm{p}, f_\mathrm{p})$

The newly generated offspring is evaluated and replaces the parent if it has a better fitness. This loop is repeated until the termination criterion is met. After termination, the best solution (i.e., the parent after the last iteration) is returned together with its fitness value.

The magnitude of the random perturbation added to the offspring is determined by σ, which is the so-called *stepsize parameter*. It is updated every G iterations based on the success rate p_s, which is the ratio of the mutations that generated an offspring fitter than the parent. The update of σ follows the so-called $1/5$*th-success rule* [Rec73]; if the success rate is higher than $1/5$th, the stepsize should be increased, if the success rate is lower than $1/5$th, the stepsize should be decreased, and if the success rate is equal to $1/5$th, it remains unchanged. For increasing and decreasing the stepsize, a constant multiplication factor $0.817 \leq c < 1$ is used.

For the parameter c, a reasonable setting is $c = 0.85$ (see [Bäc96]). For the parameter G, a reasonable setting is $G = n$ for $n \geq 5$. The initial stepsize σ used here is proportional to the expected distance to the optimum of the initial parent, which according to Schwefel [Sch77] is a reasonable initial stepsize. In this work we adopt an initialization of $\sigma = ||\mathbf{x}_u - \mathbf{x}_l||/(3\sqrt{n})$. This is based on the fact that the expected length of a mutation step is $\mathbf{E}[\mathbf{z}] = \sigma\sqrt{n}$ and that the average distance between two random points in an n-dimensional cube is proportional to $\sqrt{n}$ [BP09] (we take $||\mathbf{x}_u - \mathbf{x}_l||/3$ as an approximation), hence, the initial mutation step is taken to be proportional to the average distance of the initial solution to the optimum.

4.2.2 The $(\mu/\rho^+_,\lambda)$-SA-Evolution Strategy

The $(\mu/\rho^+_,\lambda)$-SA-ES, originally proposed by Schwefel [Sch77], is a population based extension of the $(1+1)$-ES. Instead of generating only one offspring from one parent, λ offspring are generated from μ parents, based on both recombination and mutation. Another difference as compared to the $(1+1)$-ES is the adaptation of the control parameters of the mutation operator.

Algorithm 4.2 shows the general outline of a $(\mu/\rho^+_,\lambda)$-SA-ES. Individuals are of the form $\mathbf{a} = (\mathbf{x}, \mathbf{s}, f)$, where $\mathbf{x} \in \mathbb{R}^n$ is a vector of *object variables* (i.e., the candidate solution represented by the individual), $\mathbf{s}$ is a set of *endogenous strategy parameters* (or just *strategy parameters*), and f holds the fitness value. The strategy parameters are control parameters for the mutation operator and are evolved together with the object variables. By including the strategy parameters in the evolution process, these parameters do not have to be set externally, but evolution itself adapts them to appropriate settings. Co-evolution of internal strategy parameters is called *self-adaptation* (SA).

The algorithm starts with an initial *parent population* P_p consisting of μ parents, initialized in some fashion. For each generation, λ offspring are generated from the parent population P_p, evaluated, and added to the *offspring population* P_o. After generating the offspring, μ individuals are selected either solely from the offspring population (*comma-selection*) or from the union of the parent population and the offspring population (*plus-selection*) to form the parent population of the next generation. At the end of each generation the generation counter

g is increased and the best solution of the new parent population is stored, to be returned after termination. This procedure is repeated until the termination criterion is met.

For the generation of each offspring, a set $R \subseteq P_{\mathrm{p}}$ of ρ parents is selected randomly from the parent population for generating the new offspring (*marriage*). The strategy parameters and object variables of the selected parents are recombined (*recombine_s* and *recombine_x*) to form recombinants $\mathbf{s}$ and $\mathbf{x}$. These recombinants are thereafter mutated (*mutate_s* and *mutate_x*) and evaluated to form the new offspring. Note that first the strategy parameters, then the object variables should be mutated (according to the new strategy parameters) in order to achieve co-evolution of the strategy parameters.

Marriage: The *marriage* operation selects a random subset of ρ parents for recombination. The parameter ρ is called the *mixing number*. Within Evolution Strategies, recombination is commonly done either with $\rho = 2$ parents or with $\rho = \mu$ parents (*global recombination*).

Recombination: There are two different types of recombination: *discrete* and *intermediate*. Recombination is used for both the object variables and the strategy parameters. With discrete recombination, for each element of the recombinant it is decided uniform randomly from which of the parents the corresponding element should be copied. That is, given the ρ parental vectors $\{\mathbf{r}_1^{\mathrm{p}}, \ldots, \mathbf{r}_\rho^{\mathrm{p}}\}$, the ith of the new recombinant $\mathbf{r}^{\mathrm{o}}$ is constructed by

$$(\mathbf{r}^{\mathrm{o}})_i = \left(\mathbf{r}_{m_i}^{\mathrm{p}}\right)_i \ , \ m_i = \mathrm{rand}\{1, \ldots, \rho\}. \tag{4.2}$$

Here, the recombinant $\mathbf{r}^{\mathrm{o}}$ can be either the recombinant of the object variables $\mathbf{x}$ or of the strategy parameters $\mathbf{s}$. With intermediate recombination, the value of each element of the offspring's object variables or strategy parameters is set to the average over all parents. That is,

$$(\mathbf{r}^{\mathrm{o}})_i = \frac{1}{\rho} \sum_{j=1}^{\rho} \left(\mathbf{r}_j^{\mathrm{p}}\right)_i . \tag{4.3}$$

Figure 4.2 illustrates the working mechanism of the two different recombination schemes for recombination of two parents.

Within Evolution Strategies, it is not uncommon that different recombination types are used for the object variables and the strategy parameters. Standard practice is to use discrete recombination for the object variables, and intermediate recombination for the strategy parameters. Regarding notation, the recombination type is sometimes included in the denotation of particular Evolution Strategies by means of two subscripts on the mixing number ρ (D for discrete and I for intermediate). The first subscript denotes the recombination type for the object variables and the second denotes the recombination type for the strategy parameters. For example, a $(\mu/\rho_{DI}^{+},\lambda)$-ES denotes a $(\mu/\rho^{+},\lambda)$-ES with discrete recombination of the object variables and intermediate recombination of the strategy parameters.

Mutation: As with the $(1{+}1)$-ES, mutation works by adding a random vector generated from a multivariate Gaussian distribution, which in the simplest case is uncorrelated and isotropic.

Algorithm 4.2: General Outline of a $(\mu/\rho^{+},\lambda)$-SA-ES

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{opt}$ with objective function value f_opt

Initialize: $g \leftarrow 0$, $P_\mathrm{p} \leftarrow$ initialize $(\{\mathbf{a}_k = (\mathbf{x}_k, \mathbf{s}_k, f_k), k = 1, \ldots, \mu\})$
while not terminate **do**
  $P_\mathrm{o} \leftarrow \emptyset$
  for $k = 1 \to \lambda$ **do**
    $R_k \leftarrow$ marriage (P_p, ρ)
    $\mathbf{s}_k \leftarrow$ recombine_s (R_k)
    $\mathbf{x}_k \leftarrow$ recombine_x (R_k)
    $\mathbf{s}_k \leftarrow$ mutate_s $(\mathbf{s}_k)$
    $\mathbf{x}_k \leftarrow$ mutate_x $(\mathbf{x}_k)$
    {Box-constraint handling}
    $f_k \leftarrow f(\mathbf{x}_k)$
    $P_\mathrm{o} \leftarrow P_\mathrm{o} \cup \{(\mathbf{x}_k, \mathbf{s}_k, f_k)\}$
  end for
  if comma–selection **then**
    $P_\mathrm{p} \leftarrow$ select (P_o, μ)
  else if plus–selection **then**
    $P_\mathrm{p} \leftarrow$ select $(P_\mathrm{p} \cup P_\mathrm{o}, \mu)$
  end if
  $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt}) \leftarrow$ select_best (P_p)
  $g \leftarrow g + 1$
end while
return $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt})$

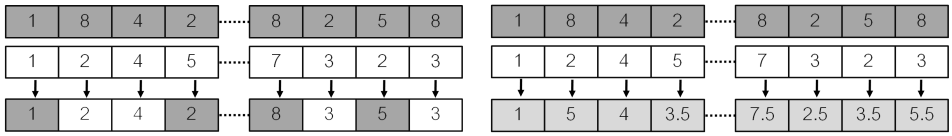


Figure 4.2: Two recombination types, illustrated for the recombination of two parents. Left: discrete recombination of two vectors where each element of the new vector is a copy of the corresponding element of one of the two parents. Right: intermediate recombination of two vectors where each element of the new vector is the average of the corresponding elements of the parents.

However, besides the object variables, also the strategy parameters are included in the mutation scheme. Moreover, as these parameters are used to control the distribution from which the perturbations are drawn, the strategy parameters are mutated before the object variables in order to achieve co-evolution of the strategy parameters. For isotropic mutations, there is only one strategy parameter, $\mathbf{s} = (\sigma)$. It scales the magnitude of the isotropic perturbations (i.e., it controls the stepsize). The combined mutation operation for an offspring with strategy parameter σ and object variables $\mathbf{x}$ is specified as:

$$\sigma = \sigma \exp\left(\tau \mathcal{N}(0,1)\right), \tag{4.4}$$

$$\mathbf{x} = \mathbf{x} + \sigma\mathbf{z}\ ,\ \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.5}$$

Here, σ is mutated using the a log-normal distribution. The parameter τ is the so-called learning parameter, which controls the magnitude of the changes of σ and is recommended to be chosen as $\tau \propto 1/\sqrt{n}$. In this work, we adopt the setting $\tau = 1/\sqrt{2n}$, which is recommended for multimodal fitness landscapes (see [BS02]).

Besides the isotropic distribution, two other variants exist: a distribution with different scalings along the main axes and a distribution with correlation of the variables. The philosophy behind using more involved distributions is that these offer the possibility to align the mutation distribution with the iso-lines of the fitness landscape, therewith speeding up the search. When denoting the mutation of the object variables as the addition of a Gaussian random variable sampled from $\sigma\mathcal{N}(\mathbf{0}, \mathbf{C})$, with $\mathbf{C}$ being the covariance matrix of the distribution, the three mutation variants can be distinguished by the form of $\mathbf{C}$. For isotropic mutations, $\mathbf{C}$ equals the identity matrix. For scaled uncorrelated mutations, $\mathbf{C}$ is a diagonal matrix. For correlated (scaled and rotated mutations), $\mathbf{C}$ is a semi-definite covariance matrix. Figure 4.3 exemplifies the differences between these three types of mutations for the mutation of a two-dimensional vector of object variables $\mathbf{x}$. It shows the density map for the generated mutations, with height lines for points of equal probability. The mutation mechanisms for the other two mutation types are described in, e.g., [Bäc96, BS02, ES03].

Regarding notation, when using mutative self-adaptation as described above, this is sometimes incorporated into the notation as $(\mu/\rho\overset{+}{,}\lambda)$-SA-Evolution Strategy. In particular, when using isotropic mutations with only one strategy parameter, σ, this is written as the $(\mu/\rho\overset{+}{,}\lambda)$-

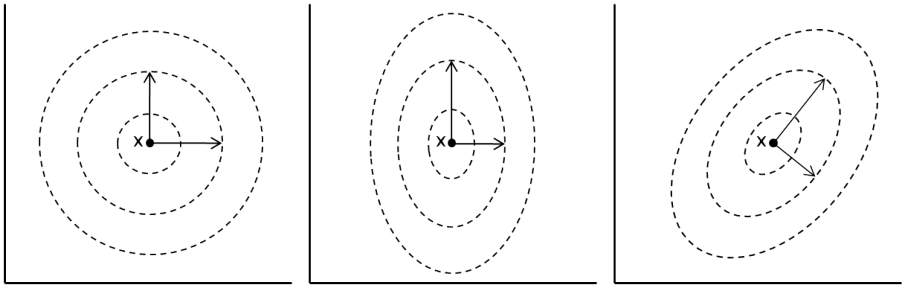


Figure 4.3: The effect of three mutation types, shown by means of iso-lines of the density map of the mutation distribution for a two-dimensional vector of object variables $\mathbf{x}$. Left: isotropic distribution. Center: scaled uncorrelated mutations. Right: correlated mutations.

σSA-Evolution Strategy.

Population size: Traditionally, the population size of Evolution Strategies is set to $\mu = 15$ and $\lambda = 100$, adhering to the recommended ratio $\mu/\lambda \approx 1/7$ (see, e.g., [Bäc96]). When using only a single stepsize parameter, smaller population sizes become also possible, such as $(1\overset{+}{,}10)$-, $(4\overset{+}{,}28)$-, and $(5\overset{+}{,}35)$-strategies.

Canonical settings: There are many different $(\mu/\rho\overset{+}{,}\lambda)$-SA-Evolution Strategy variants possible. In this work, we focus on one variant in particular, namely the $(\mu/\rho_{DI},\lambda)$-σSA-ES, with $\rho = 2$, $\mu = 5$, and $\lambda = 35$. It uses discrete recombination of the object variables and intermediate recombination of the strategy parameters, mutation is based on an isotropic multivariate Gaussian distribution, (which involves only one strategy parameter, σ), and the selection type is comma-selection. The initialization of the population is done by

$$\begin{cases} \mathbf{x}_k \sim \mathcal{U}\left(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u}\right) \\ \sigma_k = \frac{||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||}{3\sqrt{n}} \end{cases}, \; k = 1, \ldots, \mu. \tag{4.6}$$

4.2.3 The Covariance Matrix Adaptation Evolution Strategy

The *Covariance Matrix Adaptation* Evolution Strategy (CMA-ES), proposed by Hansen and Ostermeier [HO96, HO01], can be seen as a second generation Evolution Strategy. It is a $(\mu/\mu_\mathrm{W},\lambda)$ Evolution Strategy that uses comma-selection and global weighted intermediate recombination (indicated by the subscript W) in which all offspring are generated from the same recombinant $\langle\mathbf{x}\rangle_\mathrm{W}$, computed as the weighted center of mass of the μ selected individuals, i.e.,

$$\langle\mathbf{x}\rangle_\mathrm{W} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}, \tag{4.7}$$

with $\sum_{i=1}^{\mu} w_i = 1$ and $\mathbf{x}_{i:\lambda}$ denoting the object variables of the ith best individual. Within the CMA-ES, the offspring are mutated copies of this recombinant, generated as

$$\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.8}$$

$$\mathbf{y}_k = \mathbf{C}^{\frac{1}{2}}\mathbf{z}, \tag{4.9}$$

$$\mathbf{x}_k = \langle \mathbf{x} \rangle_{\mathrm{W}} + \sigma \mathbf{y}_k, \tag{4.10}$$

for $k = 1, \ldots, \lambda$. Hence, the mutations are drawn from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{C})$. The motivation for using such a mutation mechanism is to adapt the mutation distribution to the local curvature of the fitness landscape. When this is achieved, this has as effect that the mutations are taken along the gradient direction, which significantly increases the convergence speed of the algorithm, especially for ill-conditioned problems. In order to use such a mutation scheme, a proper adaptation scheme of the covariance matrix $\mathbf{C}$ and of the stepsize σ is required. In the CMA-ES, these two issues are handled by two different control mechanisms. For the former, a *covariance matrix adaptation* (CMA) scheme is used, for the latter, a stepsize adaptation mechanism named *cumulated stepsize adaptation* (CSA) is used.

The covariance matrix determines the direction of the mutations. It is initialized with $\mathbf{C} = \mathbf{I}$ and updated each generation based on a weighted empirical covariance estimation of the μ best individuals of the population (*rank-μ-update*) and on the direction of the so-called evolution path $\mathbf{p}_{\mathrm{c}}$ (*rank-one-update*). The evolution path (initialized with $\mathbf{p}_{\mathrm{c}} = \mathbf{0}$) is loosely defined as the sequence of successive steps taken by the population over a number of generations. It is a vector that tracks the direction followed by the population in the past few generations. The update of the evolution path and the covariance matrix are done in a cumulative way, with

$$\mathbf{p}_{\mathrm{c}} = (1 - c_{\mathrm{c}})\mathbf{p}_{\mathrm{c}} + \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}}\langle \mathbf{y} \rangle_{\mathrm{W}}, \tag{4.11}$$

$$\mathbf{C} = (1 - c_1 - c_\mu)\mathbf{C} + c_1 \mathbf{p}_{\mathrm{c}}\mathbf{p}_{\mathrm{c}}^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^{\mathrm{T}}. \tag{4.12}$$

Here, $\langle \mathbf{y} \rangle_{\mathrm{W}} = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$ represents the direction of the step taken by the population's center of mass $\langle \mathbf{x} \rangle_{\mathrm{W}}$. The parameter μ_{eff} is the so-called *variance effective selection mass*. The parameters c_{c}, c_1, and c_μ are the cumulation factors for the evolution path update, the rank-one-update, and the rank-μ-update respectively.

The stepsize parameter σ scales the mutations. The cumulated stepsize adaptation mechanism also uses the so-called conjugate evolution path $\mathbf{p}_\sigma$ (initialized with $\mathbf{p}_\sigma = \mathbf{0}$), which registers both the length and direction of the evolution path. When the length of the evolution path $\mathbf{p}_\sigma$ is short, then steps are taken in opposite directions that cancel each other out, yielding an ineffective circling behavior. In this case, the stepsize is decreased. When the evolution path length is large, consecutive steps are taken in the same direction, from which it can be concluded that the stepsize can be increased for faster convergence. As a reference for the evolution path length, the CSA mechanism considers the path length that would occur under

Technical Note 4.1: Parameter settings of the CMA-ES

Default population size:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor,\ \mu = \lfloor \mu' \rfloor,\ \mu' = \frac{\lambda}{2}. \tag{4.15}$$

Recombination weights:

$$w_i = \frac{w_i}{\sum_{j=1}^{\mu} w_j'},\ w_j' = \ln(\mu' + 0.5) - \ln j,\ \text{for } i = 1, \ldots, \mu. \tag{4.16}$$

Variance effective selection mass:

$$\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}. \tag{4.17}$$

Covariance matrix adaptation parameters:

$$c_{\text{c}} = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n},\ c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}},\ c_\mu = \min\left(1 - c_1, 2\frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \mu_{\text{eff}}}\right). \tag{4.18}$$

Stepsize adaptation parameters:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5},\ d_\sigma = 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma. \tag{4.19}$$

random selection. The CSA update mechanism can be summarized as

$$\mathbf{p}_\sigma = (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{-\frac{1}{2}}\langle \mathbf{y} \rangle_{\text{W}}, \tag{4.13}$$

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma||}{\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]} - 1\right)\right). \tag{4.14}$$

The update of the evolution path length $\mathbf{p}_\sigma$ is similar to the update of $\mathbf{p}_{\text{c}}$. The stepsize update uses an exponential update, with d_σ being a damping factor, and $\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]$ being the expected length of a random vector drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

The setting of the parameters used within the CMA-ES are given in Technical Note 4.1. For more details regarding these settings, the reader is referred to [HO96, HO01]. Finally, for the sake of completeness, Algorithm 4.3 provides an algorithmic description of the CMA-ES.

4.2.4 Box-Constraint Handling

For real-parameter optimization problems, the search space is bounded by the hyperbox $[\mathbf{x}_{\text{l}}, \mathbf{x}_{\text{u}}]$ and in many scenarios it is desirable to only generate candidate solutions that lie within this hyperbox. Since mutation can yield solutions that are not within this hyperbox,

Algorithm 4.3: The CMA Evolution Strategy

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{opt}$ with objective function value f_opt

Set parameters: the parameters λ, μ, μ', $w_1, \ldots, w_\mu, \mu_\mathrm{eff}$, c_c, c_1, c_μ, c_σ, d_σ are set according to Technical Note 4.1

Initialize: $g \leftarrow 0$, $\mathbf{p}_\mathrm{c} = \mathbf{0}$, $\mathbf{C} = \mathbf{I}$, $\mathbf{p}_\sigma = \mathbf{0}$, $\sigma = ||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||/(3\sqrt{n})$, $\langle \mathbf{x} \rangle_\mathrm{W} \sim \mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$

$g \leftarrow 0$

while not terminate **do**

  for $k = 1 \to \lambda$ **do**

    $\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

    $\mathbf{y}_k \leftarrow \mathbf{C}^{\frac{1}{2}} \mathbf{z}_k$

    $\mathbf{x}_k \leftarrow \langle \mathbf{x} \rangle_\mathrm{W} + \sigma \mathbf{y}_k$

    $\{$Box-constraint handling$\}$

    $f_k \leftarrow f(\mathbf{x}_k)$

  end for

  $\langle \mathbf{y} \rangle_\mathrm{W} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$

  $\langle \mathbf{x} \rangle_\mathrm{W} \leftarrow \langle \mathbf{x} \rangle_\mathrm{W} + \sigma \langle \mathbf{y} \rangle_\mathrm{W}$

  $\{$Box-constraint handling$\}$

  $\mathbf{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c}) \mathbf{p}_\mathrm{c} + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c}) \mu_\mathrm{eff}} \langle \mathbf{y} \rangle_\mathrm{W}$

  $\mathbf{C} \leftarrow (1 - c_1 - c_\mu) \mathbf{C} + c_1 \mathbf{p}_\mathrm{c} \mathbf{p}_\mathrm{c}^\mathrm{T} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^\mathrm{T}$

  $\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma) \mu_\mathrm{eff}} \mathbf{C}^{-\frac{1}{2}} \langle \mathbf{y} \rangle_\mathrm{W}$

  $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{||\mathbf{p}_\sigma||}{\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]} - 1 \right) \right)$

  $g \leftarrow g + 1$

  $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt}) \leftarrow (\mathbf{x}_{1:\lambda}, f_{1:\lambda})$

end while

return $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt})$

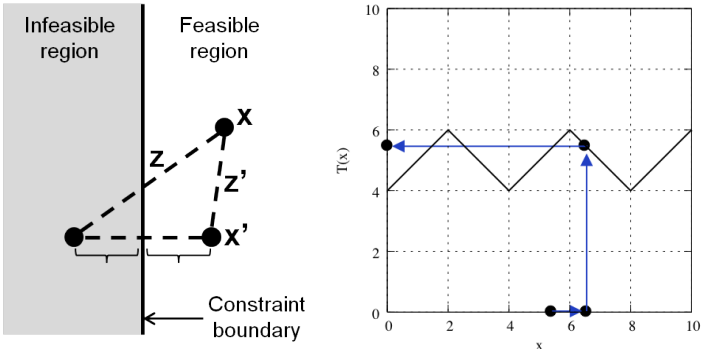


Figure 4.4: Left: visualization of reflection box-constraint handling in a two-dimensional search space. Right: an illustration of the working mechanism of the transformation function $\mathrm{T}_{[a,b]}$ with $a = 4$ and $b = 6$ (right figure, courtesy of Li [Li09]).

an additional *box-constraint handling* mechanism is needed in order to accomplish this.

For box-constraint handling, two straightforward methods are to reject mutations that fall outside the box or to apply a cutoff rule forcing mutated offspring back to the nearest point on the constraint boundary. In this work we use a *reflection* mechanism (see [Li09]) in which mutations are mirrored in the constraint boundaries. That is, the ith element of a candidate solution $\mathbf{x}$ is transformed as:

$$(\mathbf{x})_i = \mathrm{T}_{[(\mathbf{x}_\mathrm{l})_i,(\mathbf{x}_\mathrm{u})_i]}\left((\mathbf{x})_i\right), \tag{4.20}$$

with

$$\mathrm{T}_{[a,b]}(x) = x + y' \cdot (b-a), \tag{4.21}$$

$$y' = \begin{cases} |y - \lfloor y \rfloor| & \text{if } \lfloor y \rfloor \bmod 2 = 0 \\ 1 - |y - \lfloor y \rfloor| & \text{otherwise} \end{cases}, \; y = \frac{x-a}{b-a}. \tag{4.22}$$

Figure 4.4 visualizes the working mechanism of reflection.

The reason for choosing this mechanism is that it is relatively simple to implement and it preserves much of the normal dynamics of Evolution Strategies, also near the constraint boundaries. Moreover, also in practical scenarios where the optimizer might be located very close to a box-constraint boundary, this method is well applicable.

When using this box-constraint handling mechanism within the CMA-ES, note that the recombinant $\langle \mathbf{x} \rangle_\mathrm{W}$ should be computed by addition of $\sigma \langle \mathbf{y} \rangle_\mathrm{W}$, not as the weighted mean of the selected offspring, and handled separately (also with reflection). For the $(\mu/\rho^+,\lambda)$-SA-Evolution Strategy, reflection only needs to be applied to the offspring.

4.3 Summary and Discussion

This chapter has provided a brief introduction to Evolutionary Algorithms and in particular Evolution Strategies, which are designed to solve single-objective real-parameter optimization problems. It has introduced the two main algorithmic schemes that will be considered throughout this work in the context of robust optimization; the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES.

The $(5/2_{DI}, 35)$-σSA-ES is an element of the class of classical Evolution Strategies as proposed by Schwefel [Sch77]. It is a population based Evolution Strategy in which the strategy parameters are included in the encoding of the individuals in order to co-evolve them together with the object variables (self-adaptation). It uses two-parent recombination with discrete recombination of the object variables and intermediate recombination of the strategy parameters. For mutation it uses an isotropic multivariate Gaussian distribution scaled according to the stepsize parameter.

The CMA-ES [HO96, HO01] can be seen as a derandomized version a classical Evolution Strategy. The main difference is that it adopts a different way of controlling the strategy parameters. For the mutation, it uses a multivariate Gaussian distribution based on a full covariance matrix and scaled with a stepsize parameter. The covariance matrix is adapted based on the direction of successful mutations (rank-one and rank-μ update) and the stepsize parameter is adapted based on the length of the evolution path (cumulative stepsize adaptation). Furthermore, it adopts global weighted intermediate recombination for the object variables and uses a small population size as compared to the $(5/2_{DI}, 35)$-σSA-ES.

The two algorithmic schemes are considered to be instantiated canonically as described in Section 4.2.2 and Section 4.2.3, respectively, and are used in combination with reflection box-constraint handling as discussed in Section 4.2.4.

In this work, we study to what extent the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES are suitable in robust optimization scenarios, or how they should be adapted in order to make them such. Two scenarios are considered in particular: optimization of noisy objective functions and finding robust optima.

Chapter 5

Optimization of Noisy Objective Functions

Optimizing systems or models of systems that exhibit noisy output is a common scenario for real-world optimization problems. Figure 5.1 illustrates such a scenario schematically, in which the system or (simulation) model produces an output that is noisy and where this noise in the output propagates to the objective and constraint functions. This chapter focuses on a restricted subclass of such problems, being unconstrained single objective real-parameter optimization problems in which the noise in the output propagates as additive noise in the objective function.

The intent of this chapter is to answer the following questions: 1) What is the goal of optimization when having noisy objective functions? 2) What is the effect of noise in the objective functions on Evolution Strategies? 3) How should Evolution Strategies, and in particular the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, be adapted in order to deal with noisy objective functions?

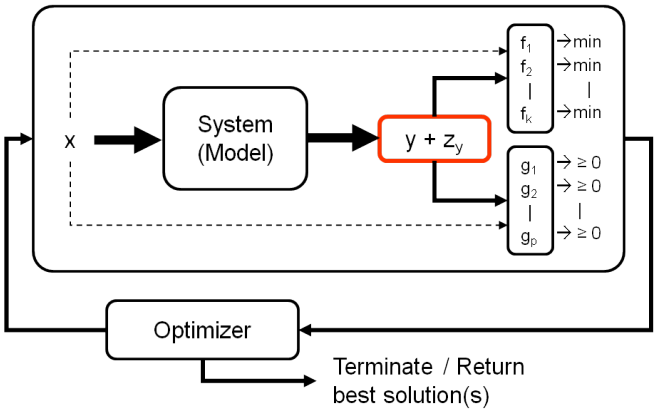


Figure 5.1: A typical robust optimization scenario: the system or model of the system for which an optimization problem needs to be solved produces a noisy output.

This chapter consists of two parts. In the first part (Section 5.1 and Section 5.2), the problem of noisy optimization and the effects of noise on Evolution Strategies are studied. Section 5.1 starts by providing a description of noisy objective functions and the goals of optimization in case of noisy objective functions. Section 5.2 studies the effects of noise in the objective functions on Evolution Strategies. In the second part of this chapter (Section 5.3 to 5.6), a number of noise handling techniques usable for Evolution Strategies are described and evaluated. Section 5.3 reviews some basic noise handling techniques, Section 5.4 reviews techniques known as adaptive averaging techniques, Section 5.5 briefly summarizes metamodel based noise handling techniques, and Section 5.6 provides a general discussion on noise handling techniques. Section 5.7 closes with a summary and discussion.

5.1 Noisy Objective Functions

The optimization problems considered in this chapter are unconstrained single-objective real-parameter optimization problems, with objective functions of the form

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + z(\mathbf{x}). \tag{5.1}$$

That is, the objective function $\tilde{f}(\mathbf{x})$ consists of a deterministic, noise-free part $f(\mathbf{x})$ and an additive stochastic part $z(\mathbf{x})$, which is a random variable indexed by space (i.e., it can be seen as a *random field* or *noise landscape*). In case of a *stationary* distribution, $z(\mathbf{x})$ are identically distributed for all $\mathbf{x} \in \mathbb{R}^n$, otherwise the noise is said to be *non-stationary*. Furthermore, the noise is *unbiased* if $\mathbf{E}[z(\mathbf{x})] = 0$ for all $\mathbf{x} \in \mathbb{R}^n$.

A common goal of optimization of noisy objective functions is to find optimal solutions for the deterministic part of the function $\tilde{f}(\mathbf{x})$. Hence, the underlying deterministic function $f(\mathbf{x})$ is considered to be the "true" objective function and the aim is to find optimal solutions for that underlying function despite the noisy evaluations. This view is considered explicitly in, e.g., [HB94, HNGK09], and is appropriate when the noise is due to measurement errors instead of being intrinsic to the system. This goal of optimization can be stated explicitly by denoting the objective function that is effectively seen as the objective function for optimization. This is called the *effective objective function* (denoted f_{eff}) which in this case is simply stated as

$$f_{\text{eff}}(\mathbf{x}) = f(\mathbf{x}). \tag{5.2}$$

An alternative goal, stated for example by Jin and Branke [JB05], is to find optimizers for the *expected objective function* (denoted f_{exp}), i.e.,

$$f_{\text{eff}}(\mathbf{x}) = f_{\text{exp}}(\mathbf{x}) = \mathbf{E}[\tilde{f}(\mathbf{x})]. \tag{5.3}$$

This aim is appropriate for systems with intrinsic noise. Although these two effective objective functions look very similar, it should be noted that these are only equivalent when the noise is unbiased.

Other goals for optimization of noisy objective functions are:

1. To optimize based on percentiles or the median.

2. To optimize based on a lower confidence bound, e.g.,

$$f_{\text{eff}}(\mathbf{x}) = \inf \left\{ a \in \mathbb{R} \mid P\left(\tilde{f}(\mathbf{x}) < a)\right) > p_\alpha \right\} \to \min, \tag{5.4}$$

   using an appropriate setting for the conflict level p_α, or

$$f_{\text{eff}}(\mathbf{x}) = \mathbf{E}[\tilde{f}(\mathbf{x})] - \omega\sqrt{\text{Var}[\tilde{f}(\mathbf{x})]}, \tag{5.5}$$

   using an appropriate weight ω.

3. To restate the optimization problem as a multi-objective problem, requiring optimization of the mean and minimization of the variance, i.e.,

$$\begin{aligned} {f_{\text{eff}}}^{(1)}(\mathbf{x}) &= \mathbf{E}[\tilde{f}(\mathbf{x})], & (5.6)\\ {f_{\text{eff}}}^{(2)}(\mathbf{x}) &= \text{Var}[\tilde{f}(\mathbf{x})] \to \min . & (5.7) \end{aligned}$$

4. To restate the optimization problem as a multi-objective problem, requiring optimization of the mean and optimization of the lower confidence bound, e.g.,

$$\begin{aligned} {f_{\text{eff}}}^{(1)}(\mathbf{x}) &= \mathbf{E}[\tilde{f}(\mathbf{x})], & (5.8)\\ {f_{\text{eff}}}^{(2)}(\mathbf{x}) &= P\left(\tilde{f}(\mathbf{x}) < T_{\text{crit}}\right) \to \min, & (5.9) \end{aligned}$$

   with T_{crit} being some critical level.

As pointed out by Sano and Kita [SK00], the latter two goals could be appropriate for optimization of investment, aiming to achieve high return and low risk solutions. However, note that for stationary noise, these alternatives do not effectively change the optimization goal as compared to the expected objective function. That is, in terms of optimization, these measures yield rankings amongst all solutions in the search space that are equivalent to the ranking based on the expected objective function.

The effective objective function states the optimization goal. However, it is obvious that it is impossible to precisely evaluate the effective objective functions stated above. Hence, for the evaluation of candidate solutions an alternative evaluation function $\hat{f}_{\text{eff}}(\mathbf{x})$ should be used that yields unbiased approximations of the effective objective functions. For instance, when using just one noisy evaluation for each candidate solution, one effectively uses $\hat{f}_{\text{eff}}(\mathbf{x}) = \tilde{f}(\mathbf{x})$, which yields unbiased approximations of the expected objective function. In literature, the step of explicitly stating an effective objective function is often omitted.

5.2 The Effects of Noise on Evolutionary Algorithms

The effects of noise on Evolution Strategies (and Evolutionary Algorithms in general) have been extensively researched over the past two decades. As noted by Beyer [Bey00], the commonly accepted viewpoint is that Evolutionary Algorithms are fairly robust against noise in the objective function based on the two empirical arguments that 1) evolution in nature is also highly influenced by noise, yet seems to work fine, and 2) in practice, Evolutionary Algorithms have shown to yield usable results for practical noisy optimization problems (that is, in the sense of melioration). Here it is considered that the fitness of each individual is determined by using one noisy evaluation (i.e., $\hat{f}_{\text{eff}}(\mathbf{x}) = \tilde{f}(\mathbf{x})$) that effectively approximates the expected objective function (i.e., $f_{\text{eff}}(\mathbf{x}) = f_{\text{exp}}(\mathbf{x})$).

In Evolutionary Algorithms, essentially only the selection operation is directly influenced by noise. Moreover, as noted by Heidrich-Meisner [HM11], for rank based selection, noise only affects selection when it changes the ranking among the individuals of the population.

The presence of noise does not necessarily have a negative impact on the performance of Evolutionary Algorithms. Noise has similar effects as the randomness that is intentionally included in commonly used selection methods, like the randomness in proportional selection and tournament selection in Genetic Algorithms [Bäc96], and this randomness can help to escape local optima. These alleged benefits of randomness induced by noise are supported by studies on theoretical cases in which adding a small noise signal to the original objective function yielded better convergence reliability [BH94] or even a higher convergence velocity [MNB08].

On the other hand, noise can also have harmful effects. The study of Beyer [Bey00] showed that for a simple quadratic function with stationary Gaussian noise, Evolutionary Algorithms fail to get infinitely close to the optimum. Instead, the population stagnates at a certain residual distance from the optimum. Given the general insight that the regions around the local optima of many continuous functions can be approximated by a quadratic model, similar effects can be expected for a wide range of noisy optimization problems.

An intuitive explanation of the reason why Evolutionary Algorithms are relatively good in dealing with noisy objective functions is that in the early stages of the evolution process the differences in fitness between all pairs of individuals are generally much bigger than the variations due to noise. Because of this, the selection mechanisms will keep a strong bias toward selecting the better solutions for reproduction, which is sufficient for Evolutionary Algorithms to progress. However, as the evolution proceeds and the population zooms in on an optimum, the differences in both the search space as well as the objective space will decrease, whereas the variance of the noise factor generally stays at the same order of magnitude. Hence, the signal to noise ratio decreases and in effect the bias toward selecting better solutions will decrease. Eventually, the selection process will degrade to uniform random selection.

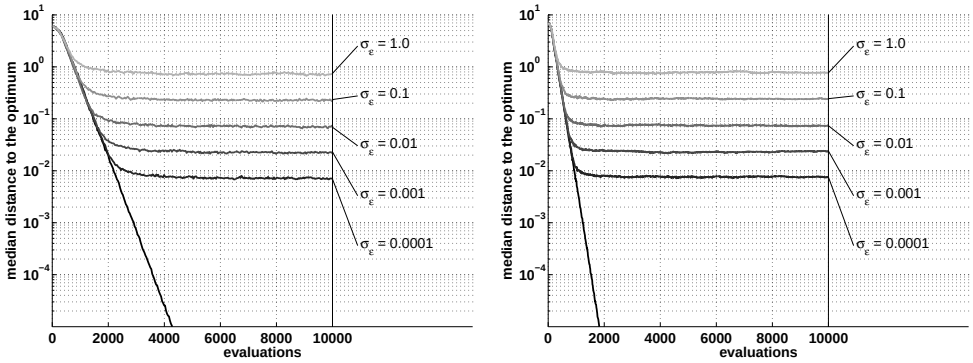


Figure 5.2: The convergence dynamics in terms of distance to the optimizer versus number of evaluations (median over 100 runs) of the $(5/2_{DI}, 35)$-σSA-ES (left) and the CMA-ES (right) on the noisy sphere problem for different noise levels ($\sigma_\epsilon = 0$, $\sigma_\epsilon = 0.0001$, $\sigma_\epsilon = 0.001$, $\sigma_\epsilon = 0.01$, $\sigma_\epsilon = 0.1$, and $\sigma_\epsilon = 1$).

Hence, although noise may be advantageous in some cases or at some stages of the optimization process, it can deteriorate the accurate localization of an optimum, and canonical Evolutionary Algorithms need to be equipped with noise handling mechanisms in order to enable them to locate optima of the expected objective function more precisely.

To illustrate the effect of noise on Evolution Strategies, and in particular on the variants that are the focus of this work, we set up the following experiment:

Experiment 5.2.1 (Performance of Evolution Strategies on the noisy sphere problem)**:** We perform 100 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on the 10-dimensional noisy sphere problem (see Appendix A.1) with varying noise levels ($\sigma_\epsilon = 0$, $\sigma_\epsilon = 0.0001$, $\sigma_\epsilon = 0.001$, $\sigma_\epsilon = 0.01$, $\sigma_\epsilon = 0.1$, and $\sigma_\epsilon = 1$). Each run has a budget of $10,000$ function evaluations.

Figure 5.2 shows the results of Experiment 5.2.1 by means of the performance, measured in terms of the median distance to the optimizer, versus evaluations. The plots show that for both algorithms the performance in the early stages of the evolution is not affected by noise, yielding the same *progress rate* (that is, the improvement in the direction of the optimum) for each noise level. However, for each noise level, there is a specific point when the progress rate deteriorates and finally reaches zero. That is, the optimization process stagnates at a certain distance to the optimizer. The higher the noise level, the earlier the stagnation and the higher the distance to the optimizer at which the stagnation occurs.

For the $(1, \lambda)$-σSA-ES, the (μ, λ)-σSA-ES (without recombination) and the $(\mu/\mu, \lambda)$-σSA-ES, Beyer [Bey00] derived lower bounds for the residual distance R_∞ for the noisy sphere problem. For the (μ, λ)-σSA-ES (without recombination) on the noisy sphere problem it is

derived as

$$R_\infty \geq \frac{1}{2}\sqrt{\frac{\sigma_\epsilon N}{4\sqrt{\mu}c_{\mu,\lambda}}}\ ,\ c_{\mu,\lambda} \sim \sqrt{2\ln(\lambda/\mu)}, \tag{5.10}$$

with $c_{\mu,\lambda}$ being the so-called *progress coefficient* (see [Bey94] for the derivation of the progress coefficient). An approximation for the $(\mu/\mu,\lambda)$-σSA-ES, which most closely resembles the weighted recombination used in the CMA-ES, was obtained by Arnold and Beyer [AB02] as

$$R_\infty \simeq \frac{1}{2}\sqrt{\frac{\sigma_\epsilon N}{4\mu c_{\mu/\mu,\lambda}}}\ ,\ c_{\mu/\mu,\lambda} = \mathcal{O}(\sqrt{\ln(\lambda/\mu)}), \tag{5.11}$$

with $c_{\mu/\mu,\lambda}$ being derived in [Bey95, Bey96]. From this, it can be concluded that, in order to increase the convergence accuracy of Evolution Strategies, either the population size should be increased (that is, either μ, λ, or both) or the noise factor σ_ϵ should be decreased. Moreover, comparing Eq. 5.10 with Eq. 5.11, it can be concluded that for the noisy sphere problem, using (multi-) recombination improves convergence accuracy. Regarding the latter, Hammel and Bäck [HB94] reported that also two-parent recombination improves the performance of Evolution Strategies on noisy functions. Section 5.3 will discuss the technique of increasing the population size or decreasing the evaluation error as an active way of noise handling in more detail.

Finally, an issue specific for Evolution Strategies is the effect of noise on the adaptation of the strategy parameters (i.e., the stepsize, and for the CMA-ES also the update of the covariance matrix). Obviously, when the signal-to-noise ratio within a population becomes too small, the failures in selecting the fitter individuals will also affect the adaptation of the strategy parameters. However, especially when considering noise handling schemes it is important to know how the adaptation mechanisms of the strategy parameters are affected by noise, and, following that, at which noise ratio these adaptation mechanisms will yield inappropriate/counterproductive parameter settings.

Using the results of Experiment 5.2.1, Figure 5.3 shows the development of the stepsize for the noise level $\sigma_\epsilon = 1.0$ (both the mean performance and the development of a single run) compared to the stepsize development for the noise-free case. For the $(5/2_{DI},35)$-σSA-ES, the stepsize in each generation is the average stepsize of all selected parents, and for the CMA-ES, the plotted stepsize is the scaling factor of the mutations, σ (i.e., not accounting for the covariance matrix factor). It can be seen that also the stepsize stagnates at some level. Hence, the mutations remain fairly high although effectively the population does not get closer to the optimum. This behavior is comparable for both algorithmic schemes (although they use different stepsize adaptation mechanisms). Moreover, the single run dynamics show that the stepsize develops like a bounded random walk.

Not many studies exist on the adaptation of the stepsize in noisy scenarios. Two studies by Arnold and Beyer [AB04, AB08] consider the behavior of cumulative stepsize adaptation.

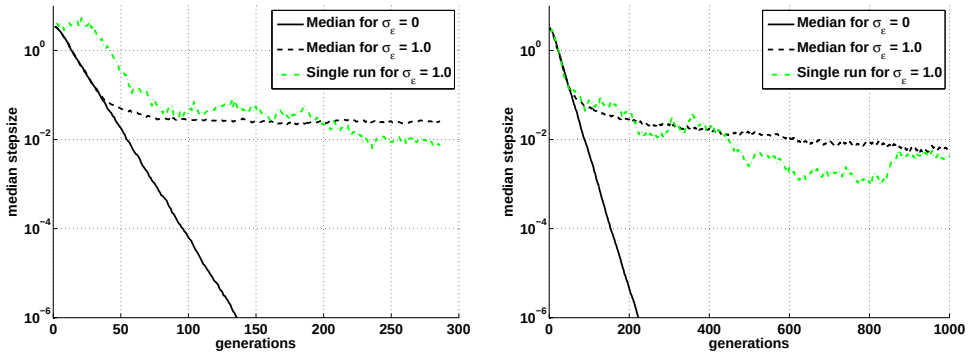


Figure 5.3: The stepsize development (both of a single run and the median over 100 runs) of the $(5/2_{DI}, 35)$-σSA-ES (left) and the CMA-ES (right) on the noisy sphere problem with noise level $\sigma_\epsilon = 0.1$ compared to the stepsize development on the noise-free sphere problem (median over 100 runs).

They conclude that noise affects the proper adaptation of the mutation strength as compared to the theoretical optimal mutation strength, and that this effect can be counteracted (again) by increasing the population size. Another interesting result is presented in [Bey00], showing an example where the failure of a proposed noise handling scheme is attributed primarily to a wrong adaptation of the stepsize.

In conclusion, we can summarize the findings from literature that are most interesting for practical application of Evolution Strategies on noisy objective functions:

- The robustness with respect to noisy objective functions is implicitly defined as the ability of Evolutionary Algorithms of finding high quality solutions with respect to the expected objective function value (i.e., $f_{\text{eff}} = f_{\text{exp}}$).
- As long as the noise level is small compared to the difference in objective function values of the individuals, noise does not affect the performance of Evolutionary Algorithms.
- Increasing the population size (μ, λ, or both) increases the convergence accuracy, meaning that the population will be able to converge closer to a local optimizer.
- Using any common type of recombination increases the ability to closer approximate the optimum on the noisy sphere problem.
- For adaptation of the strategy parameters, increasing the population size increases the reliability of the adaptation of the stepsize for cumulative stepsize adaptation.

5.3 Basic Noise Handling

As mentioned, Evolutionary Algorithms and Evolution Strategies are fairly robust against noise. However, at a certain point during the optimization when the signal-to-noise ratio

becomes too low, they will stagnate and not converge any closer to the optimum. When more accurate localization of an optimizer is required, additional measures are needed. This section will provide an overview of some basic techniques that can be used in the context of optimization of the expected objective function.

5.3.1 Resampling

Resampling or *explicit averaging* is a straightforward approach to obtain better convergence accuracy by approximating f_{exp} as the sample mean over m samples

$$\hat{f}_{\text{exp}}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} \tilde{f}(\mathbf{x}). \tag{5.12}$$

For Gaussian noise $z(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\epsilon^2)$, this yields an approximation error of

$$\bar{\sigma}_\epsilon = \sqrt{\text{Var}[\hat{f}_{\text{exp}}(\mathbf{x})]} = \sigma_\epsilon / \sqrt{m}. \tag{5.13}$$

Because the sample mean is an unbiased estimator of f_{exp}, this approach effectively reduces the noise level of objective function $\tilde{f}$ by a factor of $\sqrt{m}$, allowing for a closer convergence to the optimum. An obvious downside of this approach is that using multiple samples per fitness evaluation increases the computational effort by a factor m. Especially for limited evaluation budgets, determining an appropriate setting for m can be a tedious task.

5.3.2 Increasing the Population Size

As discussed in the previous section, for Evolution Strategies it has been observed that increasing the population size is also a way of improving the convergence accuracy. Interestingly, also in the context of Genetic Algorithms, this observation was made by Fitzpatrick and Grefenstette [FG88], and Miller and Goldberg [MG96] showed that for infinite population sizes, proportional selection is not affected by noise. Increasing the population size as an active way of noise handling is also referred to as *implicit averaging*, as opposed to the alternative of *explicit averaging* by means of resampling.

For Evolution Strategies, increasing both μ and λ can reduce the effects of noise. Considering that increasing μ does not increase the number of evaluations per generation, this may suggest that we have a free way of increasing the convergence accuracy. However, as noted in [HB94], the price of increasing μ is a lower selection pressure, which yields a lower convergence speed. Hence, increasing μ has an indirect effect on the convergence speed. Alternatively, one could increase both μ and λ. Although this does have a direct effect on the number of evaluations per generation, and yields a slower convergence speed, it leads to a higher convergence accuracy.

In order to obtain a clearer view on the effects of noise and different population sizes for the two particular schemes considered in this work, consider the following small experiment:

$(\mu/2_{DI}, \lambda)$-σ**SA-ES**	**CMA-ES**
$\lambda = 35$, $\mu = 5, 10, 15, 20, 25, 30$	$\lambda = 10$, $\mu = 3, 5, 7, 9$
$\lambda = 100$, $\mu = 15, 25, 35, 45, 55, 65$	$\lambda = 30$, $\mu = 10, 15, 20, 25, 30$

Table 5.1: The population size settings considered in Experiment 5.3.1.

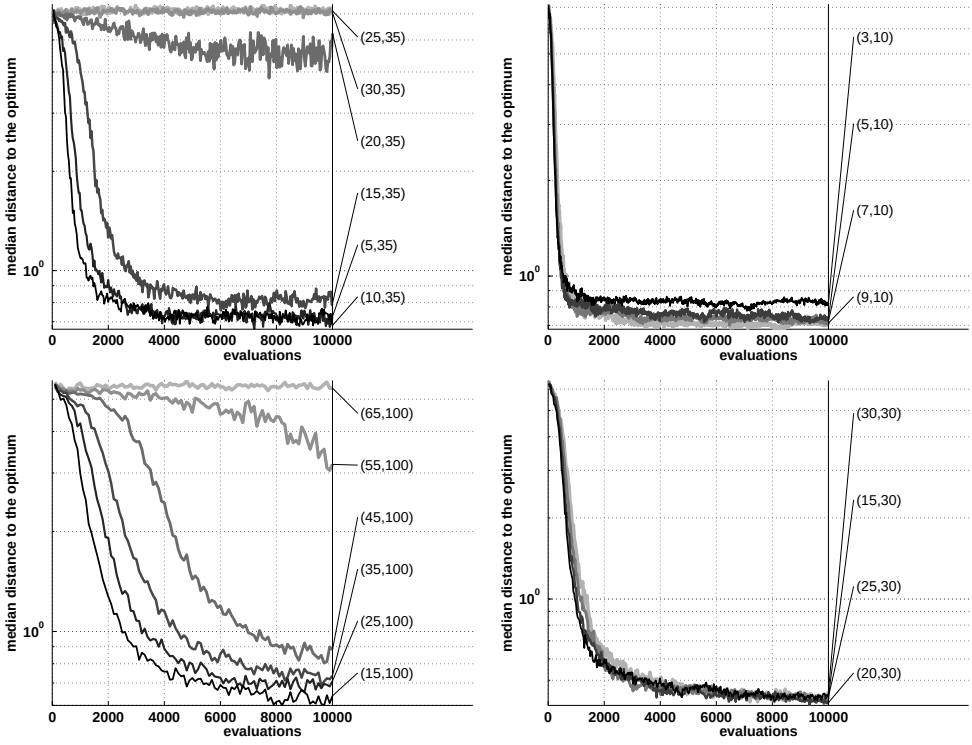


Figure 5.4: The performance of the $(\mu/2_{DI}, \lambda)$-σSA-ES (left) and the CMA-ES (right) on the noisy sphere problem for varying population sizes. The top row shows the default value for λ, varying μ, the bottom row shows a value of λ that is approximately three times the default value, again varying μ. The performance is measured in terms of distance to the optimizer versus evaluations (median over 100 runs).

Experiment 5.3.1 (Effect of higher population sizes on the noisy sphere problem)**:** We perform 100 runs of a $(\mu/2_{DI}, \lambda)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on the 10-dimensional noisy sphere problem (see Appendix A.1). For each scheme, two settings of λ are considered, each with varying settings of μ (see Table 5.1). For each run, an evaluation budget of $10,000$ function evaluations is used.

Figure 5.4 shows the results of Experiment 5.3.1 by means of plots of the performance, measured in terms of the median distance to the optimizer, versus the number of evaluations. From these results, two conclusions can be drawn: First, increasing λ yields a higher convergence accuracy, but a lower convergence speed. This is indeed in line with what was expected.

Second, increasing μ can improve convergence quality, but for the $(\mu/2_{DI}, \lambda)$-σSA-ES, the convergence speed drastically decreases with increasing values for μ, whereas for the CMA-ES, $\mu \approx \lambda$ still seems to work and actually yields a good trade-off between convergence speed and convergence accuracy. The latter is a remarkable result, as setting $\mu = \lambda$ seemingly implies that there is no selection pressure. A simple explanation for why this works for the CMA-ES is that it uses weighted recombination, assigning a higher weight to the fitter individuals in the logarithmically weighted average (which can be seen as an implicit selection mechanism).

In [Bey00], it is recommended that for the $(\mu/\mu_I, \lambda)$-ES, given a fixed offspring number λ, the value of μ should be chosen such that $\mu = \lambda/2$. For the (μ, λ)-σSA-ES, in [HB94] it is recommended to set $\mu \approx 1/7$, whereas in [Bey00], a derivation based on Eq. 5.10 showed that it should be set to $\mu = \lambda/e$. Based on this, and on the results shown in Figure 5.4, we postulate that good alternative settings in case of 10-dimensional noisy objective functions with stationary noise, are: $\mu \approx \lambda/e$ for the (μ, λ)-σSA-ES and $\mu \approx \lambda$ for the CMA-ES (this should be investigated in more depth). For the setting of λ, there is an inherent trade-off between convergence speed and convergence accuracy, making it purely dependent on the available budget of function evaluations.

5.3.3 Implicit Averaging versus Explicit Averaging

Given the two techniques to increase convergence accuracy, implicit and explicit averaging, the question arises which one is better. In [BOS03], it is stressed that for the $(\mu/\mu_I, \lambda)$-ES, given a fixed noise strength, it is more efficient to increase the offspring number by a factor m instead of resampling the objective function m times. On the other hand, Hammel and Bäck [HB94] conclude exactly the opposite based on an experimental study on the (μ, λ)-ES (i.e., resampling is better than increasing the population size). In order to form a picture, we perform the following experiment:

Experiment 5.3.2 (Implicit versus explicit averaging)**:** For comparing implicit versus explicit averaging, we perform 100 runs of a $(\mu/2_{DI}, \lambda)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on the 10-dimensional noisy sphere problem (see Appendix A.1) and on a multimodal problem; the 10-dimensional noisy Griewank problem (see Appendix A.6). We take for implicit averaging for the $(\mu/2_{DI}, \lambda)$-σSA-ES: a $(5, 35)$-, a $(25, 175)$-, and a $(50, 350)$-strategy, and for the CMA-ES: a $(5, 10)$-, a $(25, 50)$-, and a $(50, 100)$-strategy. For the resampling schemes we consider: $m = 1$ (i.e., no resampling), $m = 5$, $m = 10$. Evaluation budget for each run: $10,000$.

Figure 5.5 shows for Experiment 5.3.2 the convergence plots in terms of distance to the optimizer versus evaluations. Interestingly, we can observe a remarkable difference between the $(\mu/2_{DI}, \lambda)$-σSA-ES and the CMA-ES. For the $(\mu/2_{DI}, \lambda)$-σSA-ES, explicit resampling seems to yield better results than implicit averaging. That is, using larger population sizes seems to slow down the convergence. However, for the CMA-ES, increasing the population

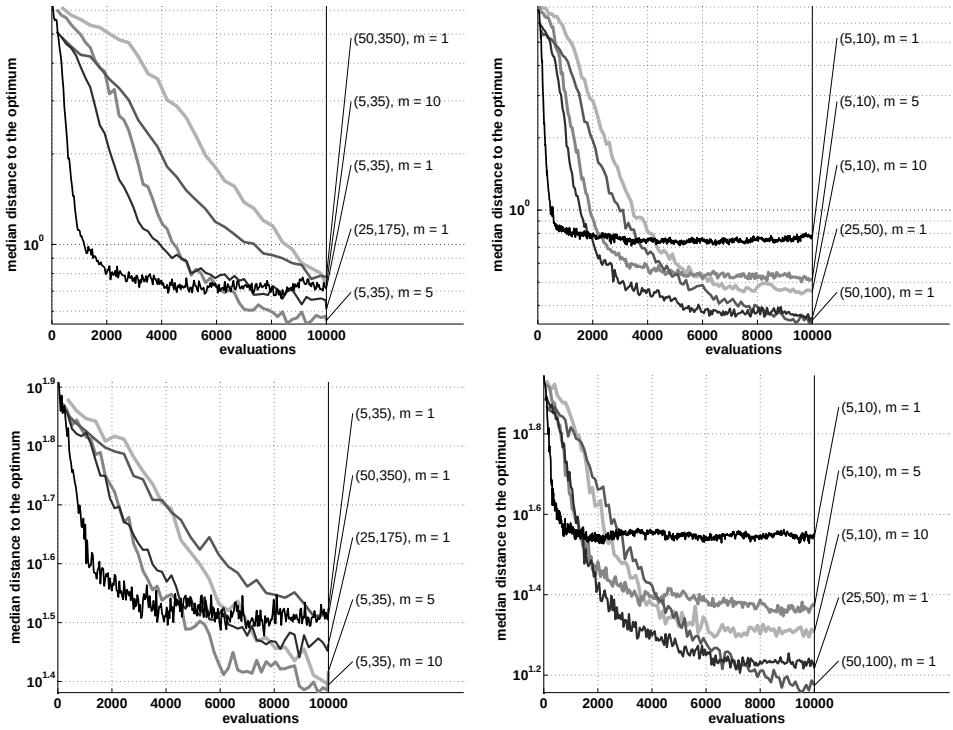


Figure 5.5: The performance of the $(\mu/2_{DI}, \lambda)$-σSA-ES (left) and the CMA-ES (right) on the noisy sphere problem (top row) and the noisy Griewank problem (bottom row). Comparing implicit versus explicit resampling. The performance is measured in terms of distance to the optimizer versus evaluations (median over 100 runs).

size seems to be most beneficial. Moreover, for both implicit averaging and explicit averaging, the CMA-ES obtains much better results than the $(\mu/2_{DI}, \lambda)$-σSA-ES. The results shown in Figure 5.5 can well explain the difference between the conclusions of Hammel and Bäck [HB94], and those of Beyer [BOS03].

5.3.4 Rescaled Mutations

An alternative approach of noise handling that does not require additional evaluations per generation is to use *rescaled mutations*. This technique was proposed originally by Rechenberg [Rec94] and further investigated by Beyer [Bey98, Bey00]. Using rescaled mutations is based on the slogan "*mutate large, but inherit small*" and it basically does just that. Instead of performing mutation in the normal way, a large mutation is applied to each individual (that is, large compared to the current stepsize) and selection is based on the fitness values of the offspring generated by these large mutations. However, after selection, instead of using the large mutation for the selected offspring, the mutation is rescaled to a small mutation. Hence, selection is based on large mutations, while small mutations in the same directions as the

successful large mutations are eventually used after selection.

As an example, consider the $(1, \lambda)$-strategy and let offspring i be obtained by $\mathbf{x}_i = \mathbf{x}_\mathrm{p} + \mathbf{z}_i$, $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$. Now, let $\mathbf{x}_{1:\lambda}$ denote the best offspring, which was generated by mutation $\mathbf{z}_{1:\lambda}$. Instead of using $\mathbf{x}_{1:\lambda}$ as parent for the next generation, the mutation is rescaled by a factor of $1/\kappa$, $\kappa \geq 1$. Hence, the parent for the next generation is computed as

$$\mathbf{x}_\mathrm{p} = \mathbf{x}_\mathrm{p} + \frac{1}{\kappa}\mathbf{z}_{1:\lambda}. \tag{5.14}$$

When assuming local or quasi linearity of the search space, the direction of the best large step should also be the best direction of improvement for a small step. Hence, by using large mutations for evaluations, differences between individuals become more apparent.

Although the idea behind using rescaled mutations is appealing and provides theoretically promising results for the noisy sphere problem, in practice, it does not yield convincing results, not even for the noisy sphere problem (see, e.g., [Bey00]). For the noisy sphere problem, this is attributed to a wrong adaptation of the stepsize (even after inclusion of fixes for the stepsize adaptation). For other problem types, one could argue that the assumed quasi linearity of the search space only holds very locally, i.e., only works for $\kappa \approx 1$. Based on these considerations, we can conclude that this noise handling technique is not off-the-shelf usable in practical scenarios.

5.3.5 Thresholding

Thresholding is a noise handling technique proposed by Markon et al. [MMA+01]. Thresholding is a simple technique, however, usable only for *plus*-selection strategies. The idea behind thresholding is that an offspring is only accepted to replace a parent if it is at least a constant $\tau > 0$ better. That is, in order to prevent the selection of outliers, offspring are required to be considerably fitter than their parents in order to be selected.

The study by Markon et al. [MMA+01] uses the $(1+1)$-ES as algorithmic basis and includes theoretical analysis of thresholding on the noisy sphere problem with Gaussian noise. For this setting, they provide a derivation on how to set τ when having an estimate of the noise strength and an estimate of the fitness difference with respect to the optimum. For this setting of τ, the $(1+1)$-ES with thresholding can converge arbitrarily close to the optimum.

Although the theory behind the study of Markon et al. [MMA+01] is sound and the results look promising, the gap with practical applicability is still considerable. Obtaining accurate estimates of the noise strength and the fitness difference with respect to the optimum is a tedious task in itself. Furthermore, the extension to multi-membered *comma*-strategies requires a number of adaptations. An approach that can be considered as a multi-membered extension of this approach will be described in Section 5.4.3.

5.4 Adaptive Averaging

Among the basic noise handling approaches discussed up to now, the two straightforward approaches of implicit and explicit averaging seem the most effective. However, both suffer from the problem that they only decrease the effect of noise, but do not eliminate it. Moreover, these techniques introduce a trade-off between using a small population/sample size that yields a high convergence speed, but a low convergence accuracy versus using a large population/sample size that yields a low convergence speed, but high convergence accuracy.

From this perspective it is desirable to have a method that adapts the intensity of the noise handling scheme such that convergence is maintained, but evaluations are not wasted. For implicit averaging, this means to start out with a small population size, and increase the population size when the population converges. For explicit averaging, this could be done in a similar way, and in addition one could distribute the sampling budget over the individuals in an efficient way. Methods that control the evaluation intensity are referred to as *adaptive averaging* methods. Several of such techniques have been proposed in the context of Evolutionary Algorithms. This section summarizes the most prominent ideas.

5.4.1 Duration Scheduling and Sample Allocation

Aizawa and Wah [AW93, AW94] proposed two adaptive resampling schemes for noisy objective functions in the context of Genetic Algorithms, based on two underlying scheduling problems that emerge when dealing with noisy objective functions:

- **Duration scheduling problem**: the problem of determining whether the quality of the objective function approximations of the individuals in the population is sufficient to end the current generation and use the current approximations for selection.

- **Sample allocation problem**: the problem of allocating evaluations (samples) to each individual in the population given a budget of evaluations such that it is most beneficial for the current generation.

The former emerges when premature convergence needs to be prevented while having no practical limitations on the evaluation budget. The latter emerges when evaluation is costly and it is important to spend the evaluations as effectively as possible within each generation. They proposed two separate approaches for these two scheduling problems.

Both the duration scheduling approach and the sample allocation approach are based on two assumptions: 1) the noise is stationary and has a Gaussian distribution, and 2) the "real" underlying objective function values of the individuals in a population are normally distributed. Based on these assumptions and on approximations of the parameters of both distributions (see Technical Note 5.1), a Bayesian approach is used to approximate the fitness of each individual (see Technical Note 5.2).

Technical Note 5.1: Estimating the Population and Noise Variance

When assuming that the noise in the objective functions is stationary and has a Gaussian distribution, $\mathcal{N}(0, \sigma_\epsilon^2)$, and the "real" underlying objective function values of the individuals in a population are also normally distributed, $\mathcal{N}(f_0, \sigma_0^2)$, then σ_0^2, f_0, and σ_ϵ^2 can be approximated as

$$\hat{\sigma}_\epsilon^2 = \frac{\sum_{i=1}^{\lambda}((m_i - 1) \cdot s_i^2)}{(\sum_{i=1}^{\lambda} m_i) - \lambda}, \; s_i^2 = \sum_{j=1}^{m_i}(\tilde{f}_{i,j} - \bar{f}_i)^2, \tag{5.15}$$

$$\hat{f}_0 = \frac{1}{\lambda} \cdot \sum_{i=1}^{\lambda} \bar{f}_i, \tag{5.16}$$

$$\hat{\sigma}_0^2 = \frac{1}{\lambda \cdot (\lambda - 1)} \left(\lambda \cdot \sum_{i=1}^{\lambda} (\bar{f}_i)^2 - \left(\sum_{i=1}^{\lambda} \bar{f}_i \right)^2 \right) - \frac{\hat{\sigma}_\epsilon^2}{m}, \tag{5.17}$$

where $m_1 = \ldots = m_\lambda = m$ is assumed.

Remark: An issue not mentioned in [AW94], but very relevant in practice, is that the estimate from Eq. 5.17 becomes unusable when the ratio $\sigma_0^2/(\sigma_\epsilon^2/m)$ becomes too small. That is, if the ratio between the population variance and the variance of the sample mean (σ_ϵ^2/m) becomes smaller, the estimate $\hat{\sigma}_0^2$ becomes less accurate and might even become negative (which, being a variance, is an unreasonable estimate). This should be accounted for in practical situations, because it harms the fitness estimates. Furthermore, it should be noted that this approach requires at least two samples for each individual in the population.

The **duration scheduling** approach aims to automatically adapt the number of samples used for resampling such that the Evolutionary Algorithm maintains progress. Primarily it uses a static incremental scheme, determining a budget t_k of samples available for generation k as

$$t_k = \lambda \cdot \left(m_0 + \left\lceil \gamma \sum_{i=1}^{k-1} t_i \right\rceil \right), \tag{5.22}$$

where λ is the population size, m_0 is the initial number of samples used for each individual, and γ is a heuristic parameter. Secondly, it uses the following bounds for the ratio between the effective variance of the noise (see Eq. 5.13) and the population variance as an indicator of whether the current generation can be terminated:

$$\delta_l \leq \frac{\sigma_\epsilon/\sqrt{m}}{\sigma_0} \leq \delta_u. \tag{5.23}$$

If this ratio is small enough (i.e., $< \delta_l$), no resampling is necessary and the resampling loop is terminated even if the current evaluation budget t_k is not spend entirely. When it is greater than δ_u, resampling is necessary and resampling is continued even if the evaluation budget t_k is exceeded. Here, σ_ϵ and σ_0 are estimated as described in Technical Note 5.1.

Technical Note 5.2: Bayesian Fitness Approximation

Assume that the objective function value of candidate solution $\mathbf{x}_i$ is normally distributed, i.e.,

$$\tilde{f}_i \sim \mathcal{N}(f_i, \sigma_\epsilon^2). \tag{5.18}$$

Furthermore, assume that the "real" objective function values $f_1, \ldots, f_\lambda$ of the λ individuals within the population are normally distributed, $\mathcal{N}(f_0, \sigma_0^2)$, yielding for each individual i, a *prior distribution* $h(f_i) \sim \mathcal{N}(f_0, \sigma_0^2)$.

Let $\bar{f}_i$ be the mean of m_i fitness evaluations $(\tilde{f}_{i,1}, \ldots, \tilde{f}_{i,m_i})$ for candidate solution $\mathbf{x}_i$. By definition, we know:

$$p(\bar{f}_i \mid f_i) \sim \mathcal{N}(f_i, \sigma_\epsilon^2/m_i). \tag{5.19}$$

Using Bayes formula, we obtain a *posterior distribution* for individual i

$$h^*(f_i \mid \bar{f}_i) = \frac{p(\bar{f}_i \mid f_i) \cdot h(f_i)}{\int_{-\infty}^{\infty} p(\bar{f}_j \mid f_j) \cdot h(f_j) \cdot df_j}. \tag{5.20}$$

From this, the best estimator $\hat{f}_i$ for f_i with estimation error $\hat{\sigma}_i$ is given by

$$\hat{f}_i = \frac{m_i \cdot \bar{f}_i + \alpha \cdot f_0}{m_i + \alpha}\ ,\ \hat{\sigma}_i = \frac{\sigma_\epsilon^2}{m_i + \alpha}\ ,\ \alpha = \frac{\sigma_\epsilon^2}{\sigma_0^2}. \tag{5.21}$$

The values of σ_0^2, f_0, and σ_ϵ^2 can be estimated as described in Technical Note 5.1.

Regarding the implementation details of the duration scheduling approach, $\gamma = 5 \times 10^{-3}$, $\delta_l = 1.0$, $\delta_u = 4.0$ are used in [AW94]. The setting of m_0 is noted to be based on Eq. 5.23, however, the exact procedure is not described in [AW94]. Besides that, a pre-sampling step should be used to estimate σ_ϵ and σ_0, which are required within Eq. 5.23, but this pre-sampling step is not described explicitly. Also, no note is made of how often the estimates of σ_ϵ and σ_0 are updated. For the implementation of this scheme, these issues should be accounted for.

The **sample allocation** approach uses a fixed budget of objective function evaluations T every generation and aims to divide the evaluations such that it is most beneficial for selection. Technical Note 5.3 summarizes this sample allocation procedure[1]. The idea is to distribute the evaluation budget $T = (m_1 + \ldots + m_\lambda)$ in an optimal way among the individuals of the current generation. This can be accomplished by selecting the individual for resampling for which resampling will lead to the highest reduction of the following expected risk function:

$$\bar{R} = \sum_{i=1}^{\lambda} P_i \hat{\sigma}_i^2. \tag{5.24}$$

[1] The description in Technical Note 5.3 differs slightly from the description presented in [AW94], which is done for the sake of clarity.

Technical Note 5.3: Sample Allocation Procedure

1. Take a fixed number of samples (at least two[a]) for each individual in the population and set the evaluation counter $t = m_0 \cdot \lambda$.
2. For each individual $i = 1, \ldots, \lambda$ compute $\hat{\sigma}_i$ and P_i (or w_i as an approximation).
3. Take one additional sample for the individual that has the largest feedback value according to Eq. 5.28 and increase the evaluation counter $t = t + 1$.
4. Repeat step 2 to 4 until $t = T$.

[a] In [AW94] it is said to take one sample for each individual. However, at least two samples for each individual are needed to obtain estimates for σ_ϵ and σ_0.

Here, P_i is the probability that individual i is the best individual and $\hat{\sigma}_i^2$ is the estimated prediction error of individual i (see Eq. 5.21). Technical Note 5.4 describes the procedure to find the individual that contributes most to the risk function Eq. 5.24.

To summarize, Aizawa and Wah [AW94] proposed two adaptive resampling methods that are both based on a Bayesian approach for estimating the fitness that differs from the common way of doing explicit averaging. Furthermore, they introduced a way of measuring the selection uncertainty based on the ratio between the population variance and the approximation error Eq. 5.23, which is used in a duration scheduling approach. For in-generation allocation of additional fitness evaluations, a sample allocation scheme based on an expected risk function is proposed.

For applying the proposed techniques in practice, a few remarks are in place. First, it should be noted that for Evolution Strategies, using solely the Bayesian fitness approximation is not sensible, because it does not change the ranking amongst the individuals in the population as compared to explicit averaging. Furthermore, in [AW94] it is noted that the duration scheduling problem and sample allocation problem do not occur concurrently. However, this is debatable, because even if a sufficiently large evaluation budget is available for duration scheduling, it can still be desirable to spend the evaluations within a generation as effectively as possible. Lastly, no note is made of the possibility that the estimate $\hat{\sigma}_0^2$ (see Technical Note 5.1) can become too crude to be practical, or even negative (making it unusable).

5.4.2 Adaptive Resampling Based on the t-Test

Another class of adaptive averaging schemes is formed by approaches that apply reevaluation based on statistical testing of the ranking of the individuals. An obvious first choice is the t-test, which can be used for pairwise comparison of solutions. This approach is suitable when optimizing on the expected objective function, assuming that the noise on the objective

Technical Note 5.4: Minimization of the Expected Risk

Given Bayesian objective function approximations as described by Technical Note 5.2, then the probability P_i that individual i is the best individual is given by

$$P_i = \int_{-\infty}^{\infty} \left[\prod_{j \neq i} H^*(f_j \mid \bar{f}_j) \right] h^*(f_i \mid \bar{f}_i) df_i, \tag{5.25}$$

where, $H^*(f_j \mid \bar{f}_j)$ is the cumulative distribution function of $h^*(f_i \mid \bar{f}_i)$. Given that $\phi(.)$ and $\Phi(.)$ are the probability density function and the cumulative distribution function of the Gaussian distribution, respectively, and using the approximations of $\hat{f}_i$ and $\hat{\sigma}_i$, this becomes:

$$P_i = \int_{-\infty}^{\infty} \left[\prod_{j \neq i} \Phi\left(\frac{f_i - \hat{f}_j}{\hat{\sigma}_j} \right) \right] \phi\left(\frac{f_i - \hat{f}_j}{\hat{\sigma}_j} \right) df_i. \tag{5.26}$$

Alternatively, given that the computation of P_i requires numerical integration, a weight w_i can be used instead of P_i, in which each individual is only compared to the best (or second best if it is the best itself):

$$w_i = \Phi\left(\frac{\hat{f}_i - \hat{f}_j}{\sqrt{\hat{\sigma}_i^2 + \hat{\sigma}_j^2}} \right) \ , \ j = \begin{cases} k & , i \neq k \\ l & , i = k \end{cases}, \tag{5.27}$$

where k is the index of the best individual, and l is the index of the second best individual, based on the fitness approximations $\hat{f}_1, \ldots, \hat{f}_\lambda$.

Based on either P_i or w_i, the individual that should be selected for reevaluation in order to minimize the risk function of Eq. 5.24 is the individual i, computed as

$$\underset{i \in \{1,\ldots,\lambda\}}{\operatorname{argmax}} \left[P_i \frac{\hat{\sigma}_\epsilon^2}{(m_i + \alpha)^2} \right]. \tag{5.28}$$

Technical Note 5.5: Fitness Comparison Using the t-Test

Assume that the fitness of individual i is normally distributed, i.e.,

$$\tilde{f}_i \sim \mathcal{N}(f_i, \sigma_\epsilon^2). \tag{5.29}$$

Given two individuals $\mathbf{x}_i$ and $\mathbf{x}_j$, with mean fitness values $\bar{f}_i$ and $\bar{f}_j$ obtained through resampling using m_i and m_j samples respectively, sample variances s_i^2 and s_j^2, and suppose $\bar{f}_i \leq \bar{f}_j$. We can test the hypothesis

$$H_0 : \bar{f}_i \leq \bar{f}_j \tag{5.30}$$

against the hypothesis

$$H_1 : \bar{f}_i > \bar{f}_j \tag{5.31}$$

using a one-sided t-test to a significance α. Given the t-statistic

$$t_{ij} = \frac{\bar{f}_i - \bar{f}_j}{\sqrt{\frac{s_i^2}{m_i} + \frac{s_j^2}{m_j}}}, \tag{5.32}$$

we can reject H_0 if $t_{ij} > t_{(\alpha, 2m-2)}$. Here, $t_{(\alpha,2m-2)}$ is the t-distribution with $2m-2$ degrees of freedom, computed for α.

function is Gaussian. Approaches based on this statistical testing have been proposed in different studies in different settings [Sta98, CP04, KEB09a].

The t-test can be used to test whether or not the differences of the mean objective function values of two individuals is significant with a certain significance level. The approach to do so is briefly summarized in Technical Note 5.5. During the evaluation phase of the optimization, one can require for a one-sided t-test with a significance α between (certain/all) pairs of individuals, and continue resampling until this is achieved.

A first consideration that is relevant when following such an approach is whether or not a Bonferroni correction should be applied for multiple comparisons [Dun61]. The choice of applying a Bonferroni correction determines whether the statistics are based on comparison of separate pairs or on comparison of the full population. In [Sta98, CP04, KEB09a], the Bonferroni correction is not used. Secondly, note that instead of using a one-sided t-test as described in Technical Note 5.5, also a two-sided t-test could be used (see [KEB09a]). However, effectively this will not make much difference. That is, for both the one-sided as the two-sided t-test the t-statistic is the same, only the threshold value for a certain significance level α changes, yet this does not yield an essentially different indicator measure for resampling. Lastly, one has to decide which pairs of individuals are compared against each other. The approaches proposed in literature are:

- **Subsequent pair testing:** Based on a population sorted by fitness approximation

$\bar{f}_{1:\lambda}, \ldots, \bar{f}_{\lambda:\lambda}$, test for every subsequent pair of individuals $\{\mathbf{x}_{i:\lambda}, \mathbf{x}_{i+1:\lambda}\}, i = 1, \ldots, \lambda - 1$ whether the fittest is fitter with a significance α. For each individual belonging to a pair for which the significance is too low, take one additional sample. Repeat this loop until no more individuals need resampling. This method was studied in [KEB09a].

- **Test against best:** Sort the population by fitness approximation $\bar{f}_{1:\lambda}, \ldots, \bar{f}_{\lambda:\lambda}$, test every individual $\mathbf{x}_{i:\lambda}, i = 2, \ldots, \lambda$ against the best individual $\mathbf{x}_{1:\lambda}$ for a significance α. For each individual belonging to a pair for which the significance is too low, take one additional sample. Repeat this loop until no more individuals need resampling. This method was also studied in [KEB09a].

- **Pairwise tournaments:** In [CP04], pairwise comparison was used within tournament selection with tournament sizes of two. Within every tournament, resampling is continued until the fittest individual is better with a significance α.

For Evolution Strategies, using $(\mu\overset{+}{,}\lambda)$-selection, only the first two approaches are applicable.

Although the idea of using the t-test for adaptive resampling seems promising at first sight, empirical results in [CP04] and [KEB09a] show that these approaches come with serious problems. In the experiments of [KEB09a], different t-test based evaluation schemes are compared for a $(1, 10)$-σSA-ES on a 10-dimensional noisy sphere problem (with $\sigma_\epsilon = 0.1$) and an evaluation budget of $20,000$ function evaluations. These schemes are: subsequent pair adaptive resampling instances, test against best adaptive resampling instances, and a fixed sample size resampling method with a sample size tuned for this problem instance (which is $m = 50$). From the results, shown in Figure 5.6, it can be seen that tuning the parameter α is quite a tedious task. It should be strict enough in order to determine a correct ranking with sufficient confidence, but on the other hand, if it is too strict, samples might be wasted in assuring that pairs of individuals are actually different. The former leads to early convergence and the latter leads to slow convergence, as can be observed in the plots.

Moreover, a performance loss can even be observed when comparing the best adaptive resampling methods against the best fixed sample size resampling method. The latter can be attributed to an explosion of the number of samples required to achieve a certain significance level for pairs of individuals that happen to have objective function values that lie very close to each other (in the perspective of the overall differences between all λ offspring). These pairs require (unnecessarily) long resampling loops. An even more extreme scenario emerges when two individuals have exactly the same mean objective function value, which can cause (obviously undesirable) infinite evaluation loops. On objective function landscapes that contain many or large plateaus, this method is therefore likely to fail when this scenario is not accounted for.

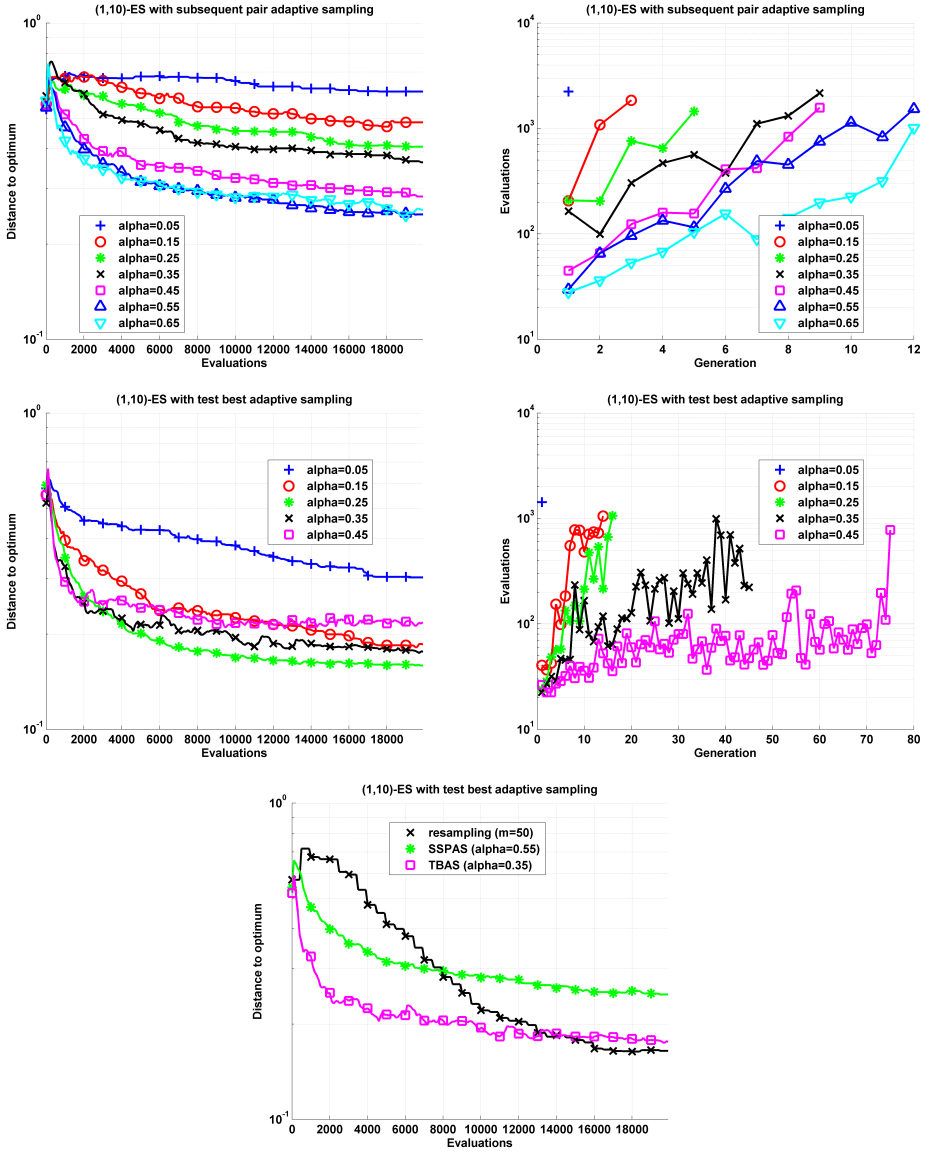


Figure 5.6: Results from [KEB09a]. Different instances of a $(1, 10)$-σSA-ES on a 10-dimensional noisy sphere problem, with $\sigma_\epsilon = 0.1$. Top row: the performance and required evaluations per generation of a $(1, 10)$-σSA-ES implementing the subsequent pair test adaptive sampling approach. Middle row: the performance and required evaluations per generation of a $(1, 10)$-σSA-ES implementing the test against best adaptive sampling approach. Bottom row: the performance of the best adaptive sampling instances compared to the performance when using a good fixed sample size approach. The results were obtained using 10 runs per algorithmic scheme, using an evaluation budget of $20,000$.

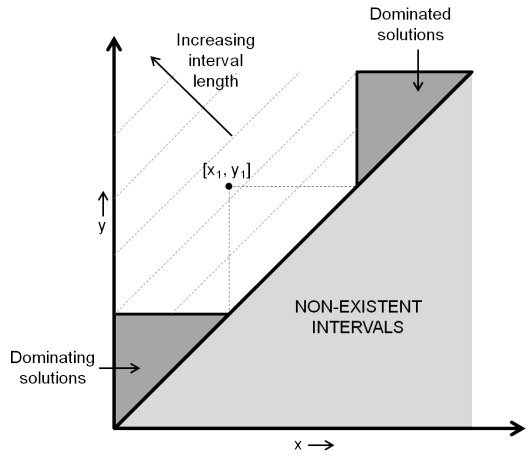


Figure 5.7: The dominance relationship for interval orders visualized geometrically. In this figure, the space of intervals is presented on a two-dimensional plane, divided into the regions of dominating intervals, dominated intervals, and incomparable intervals of the interval $[x_1, y_1]$. The line $x = y$ represents the intervals that are reduced to single points. From this figure, we see that decreasing the interval length decreases the distance to the line $x = y$, which increases the comparability.

5.4.3 Partial Order Based Adaptive Averaging

Rudolph [Rud01] proposed to actively consider the partial order that emerges when considering the noisy fitness of each candidate solution as an uncertainty interval, see Technical Note 5.6. For this, it should be assumed that the noise in the objective function is bounded within known intervals. Given this viewpoint, one could apply an Evolutionary Algorithm that selects based on the dominance relation that emerges from this partial order. Figure 5.7 visualizes the dominance relationship for interval orders. In this figure, the space of intervals is presented on a two-dimensional plane, divided into the regions of dominating intervals, dominated intervals, and incomparable intervals of the interval $[x_1, y_1]$. The line $x = y$ represents the intervals that are reduced to single points. From this figure, we see that decreasing the interval length decreases the distance to the line $x = y$, which increases the comparability.

Rudolph considered an Evolutionary Algorithm using an elitist selection strategy and for which it is guaranteed that every collection of offspring can be generated from any collection of parents. For such algorithms, it holds that for any finite search space with any noisy objective function with the noise bounded within the interval $[-a, a]$, the population will, with a probability 1, after a finite number of generations, enter a state in which all solutions have an objective function value that lies at most $3a$ away from the optimum (see [Rud01] for details).

This scheme can be extended as an adaptive averaging scheme by using the sample mean of m evaluations. For this, confidence intervals can be generated that are stricter than the original bounds, which will lead to more accurate convergence precision. For this, an adaptive resampling technique can be devised as follows: run the Evolutionary Algorithm described

Technical Note 5.6: Partial Orders Induced by Noise

For the set of intervals $\mathcal{F} = \{[x_1, x_2] \subset \mathbb{R} : x_1 \leq x_2\}$, one can introduce a strict partial order $\prec$

$$[x_1, x_2] \prec [y_1, y_2] \quad \text{iff} \quad x_2 < y_1, \tag{5.33}$$

which can be extended to a partial order by adding the relations

$$[x_1, x_2] = [y_1, y_2] \quad \text{iff} \quad x_1 = y_1 \text{ or } x_2 = y_2, \tag{5.34}$$

$$[x_1, x_2] \preceq [y_1, y_2] \quad \text{iff} \quad [x_1, x_2] \prec [y_1, y_2] \vee [x_1, x_2] = [y_1, y_2], \tag{5.35}$$

Although strictly speaking it is sufficient to consider only a strict partial order, for compliance with literature, we discuss the method in the context of partial orders. The pair $(\mathcal{F}, \preceq)$ is called a *partially ordered set* (*poset*). Furthermore, given two elements $x, y \in \mathcal{F}$, x is said to dominate y iff $x \prec y$ and both elements are said to be incomparable (denoted $x||y$) iff $x \not\prec y$ and $y \not\prec x$ [a].

In the context of noisy objective functions, one can view the noisy fitness value $\tilde{f}(\mathbf{x})$ of an individual $\mathbf{x} \in \mathcal{X}$ as an element of the interval $[f(\mathbf{x}) - a, f(\mathbf{x}) + a]$. Hence, given a noisy evaluation $\tilde{f}(\mathbf{x})$, then the true fitness $f(\mathbf{x})$ of $\mathbf{x}$ should lie within the random interval $[\tilde{f}(\mathbf{x}) - a, \tilde{f}(\mathbf{x}) + a]$.

When assuming that all feasible solutions $\mathbf{x} \in \mathcal{A}$ have been evaluated once, one can define the set of minimal elements in the set of evaluated objective function values as

$$\tilde{\mathcal{F}}^* = \{\tilde{f}(\mathbf{x}^*) \,|\, \mathbf{x}^* \in \mathcal{A} \text{ and } \nexists \mathbf{x} \in \mathcal{A} : [\tilde{f}(\mathbf{x}) - a, \tilde{f}(\mathbf{x}) + a] \prec [\tilde{f}(\mathbf{x}^*) - a, \tilde{f}(\mathbf{x}^*) + a]\}. \tag{5.36}$$

For this, Rudolph [Rud01] showed that

$$\max\{\tilde{\mathcal{F}}^*\} \leq f^* + 3a, \tag{5.37}$$

where f^* denotes the optimum of the "true" objective function. That is, all solutions in $\tilde{\mathcal{F}}^*$ lie at most $3a$ from the optimum.

[a] By abuse of notation, the notion of incomparability also includes equality.

above until the population consists of all non-dominated solutions and no improvements have been found for a number of generations. At that point, increase the sample size to tighten the confidence intervals and increase convergence accuracy.

This approach is different from other noise handling approaches in that it is the only approach that actively considers the noisy objective function values in the context of a partial order based on confidence intervals. However, comparing it to the thresholding approach (see Section 5.3.5), there is some overlap. The elitist thresholding scheme maintains non-dominated solutions in a similar way, by only accepting solutions that are a factor of τ better. In a similar fashion one could construct confidence bounds that have a similar effect as using this threshold.

The algorithm proposed in [Rud01] is not straightforwardly incorporable in the algorithmic schemes considered in this work. This is because it considers an elitist evolution loop that differs from the general evolution loop of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, and because it is based on the assumption that the noise is strictly bounded within known intervals $[-a, a]$. In order to test the idea of using the non-dominance relation amongst intervals, we consider the adaptive averaging procedure of Algorithm 5.1. This procedure, which replaces the evaluation procedure of the canonical $(5/2_{DI}, 35)$-σSA-ES and CMA-ES, counts the number of non-dominated solutions based on Gaussian confidence intervals (using confidence level δ) that are constructed from a number of samples $\lfloor m_{\text{eval}} + 1 \rfloor$. If this number reaches the parental population size μ (corresponding to a parental population that contains only non-dominated solutions), then the number of samples used for the next generation is increased with a factor α_m. The ranking amongst the individuals is based on the mean objective function values. When there are at most μ non-dominated solutions, then all of them will be selected when selecting based on the mean objective function values.

To gain insight in the behavior of this evaluation scheme we perform the following experiment:

Experiment 5.4.1 (Performance of poset based adaptive averaging on the noisy sphere problem)**:** We perform 10 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) incorporating the evaluation procedure of Algorithm 5.1 (named PUH-$(5/2_{DI}, 35)$-σSA-ES) on the 10-dimensional noisy sphere problem (see Appendix A.1). We take parameter setting $\alpha = 1.5$ and use varying $\delta = 0.1, 0.3, 0.5$. As benchmark, we include a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Each run uses a budget of $100,000$ objective function evaluations.

The results of Experiment 5.4.1 are presented in Figure 5.8 and Figure 5.9. Figure 5.8 shows the convergence dynamics of the three instances of this adaptive averaging scheme and as a benchmark the convergence dynamics of a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Figure 5.9 shows the single run and average dynamics of the three instances of this adaptive averaging scheme. The left column shows the distance to the optimizer versus the number of generations, the middle column the development of sample

Algorithm 5.1: Poset Based Adaptive Averaging

Procedure parameters: confidence level δ, averaging increment factor α

Procedure variables: sample size indicator m_{eval}, initialized at $m_{\text{eval}} = 2$

1. For all candidate solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ obtain $m = \lfloor m_{\text{eval}} + 1 \rfloor$ noisy objective function evaluations
$$\tilde{f}_{i,j} = \tilde{f}(\mathbf{x}_i)\ ,\ i = 1, \ldots, \lambda\ ,\ j = 1, \ldots, m. \tag{5.38}$$
2. For each individual $\mathbf{x}_i$, compute the mean objective function value $\bar{f}_i$, the sample variance s_i^2, and confidence bound $[\bar{f}_i - c_i, \bar{f}_i + c_i]$, with:
$$c_i = c_i^{\text{Gaussian}} = \frac{s_i}{\sqrt{m}} \Phi^{-1}\left(\frac{1+\delta}{2}\right). \tag{5.39}$$
3. Compute the number of non-dominated solutions as
$$\#\text{nds} = \left|\{i \in \{1, \ldots, \lambda\} | \nexists j \in \{1, \ldots, \lambda\} : f_j + c_j < f_i - c_i\}\right|. \tag{5.40}$$
4. Update the sample size m_{eval} using the update rule
$$m_{\text{eval}} = \begin{cases} \alpha \cdot m_{\text{eval}} & , \text{if } \#\text{nds} \geq \mu \\ m_{\text{eval}} & , \text{otherwise} \end{cases}. \tag{5.41}$$
5. Generate a ranking $\mathbf{x}_{1:\lambda}, \ldots, \mathbf{x}_{\lambda:\lambda}$ based on the sample means $\bar{f}_1, \ldots, \bar{f}_\lambda$.

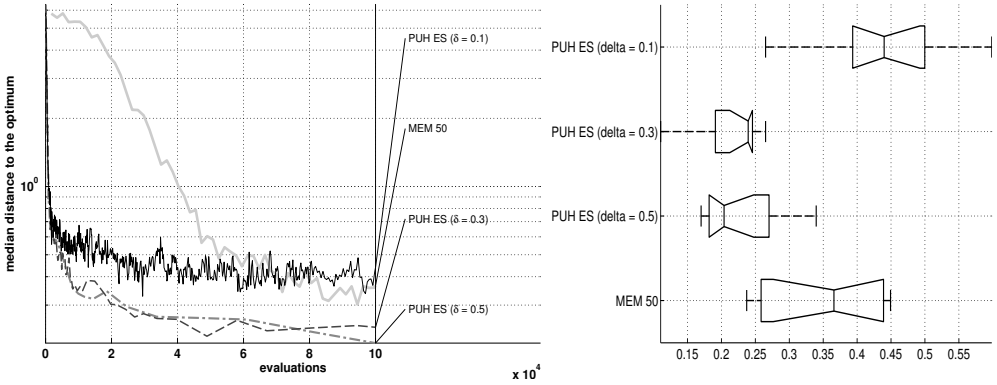


Figure 5.8: Left: The convergence dynamics (median over 10 runs) of the PUH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem using $\alpha = 1.5$ and using varying $\delta = 0.1, 0.3, 0.5$, compared against a fixed sample size resampling scheme with $m = 50$. Right: boxplots of the final solution quality.

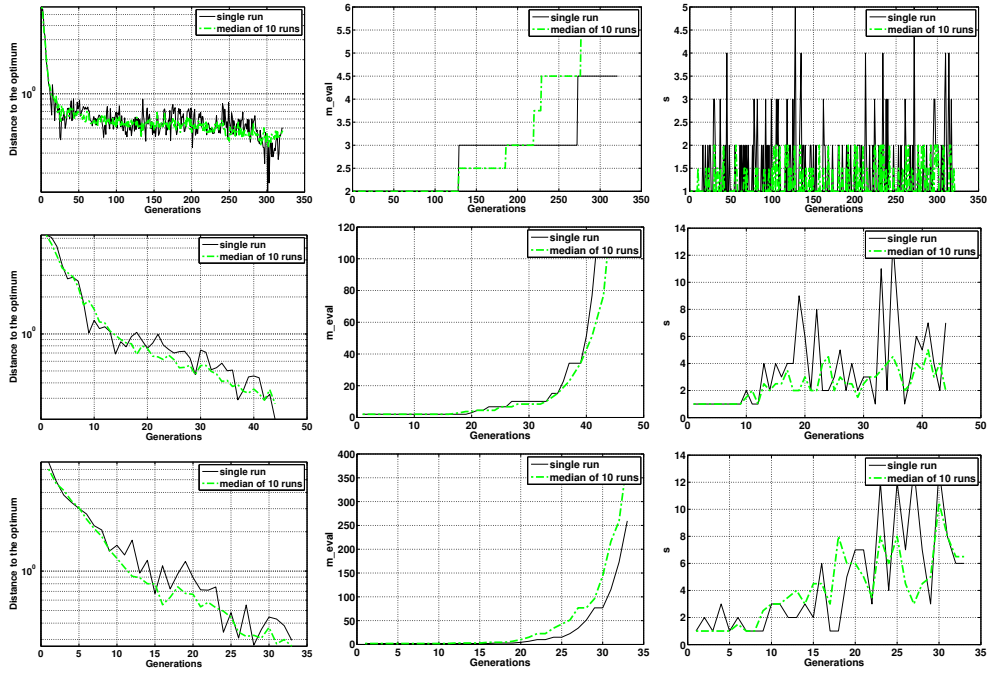


Figure 5.9: The dynamics (median over 10 runs) of the PUH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem, using different confidence levels $\delta = 0.1$ (top row), $\delta = 0.3$ (middle row), $\delta = 0.5$ (bottom row). Left: the distance to the optimizer versus number of generations. Center: the development of m_{eval}. Right: the development of the uncertainty level (i.e., the number of non-dominated solutions in the offspring population).

size parameter m_{eval}, and the right column the development of the uncertainty indicator (i.e., the number of non-dominated solutions).

In Figure 5.8 we observe promising convergence behavior of this scheme for all considered settings of δ when comparing it to the fixed sample size resampling scheme. Also the convergence plots of Figure 5.9 show promising behavior, however, we observe a big difference between the three PUH-$(5/2_{DI}, 35)$-σSA-ES variants. When δ is small (e.g., $\delta = 0.1$), corresponding to having loose confidence bounds, the uncertainty level stays low for a large number of generations. Yet, from the convergence plot of $\delta = 0.1$, we also see that the hovering behavior that is due to noise already occurs after approximately 20 generations. On the other hand, the convergence plots for higher values of δ do not show this hovering behavior, but also complete much less generations based on the same evaluation budget. The latter is due to a much faster growing uncertainty level, yielding an exponentially growing sample size m_{eval}. Here, we observe a typical dilemma of adaptive averaging techniques, which is to find a good balance between requiring a high selection accuracy that yields a good progress each generation and accepting inaccuracies in the selection that yields slower generation-wise convergence, but which allows for completing much more generations with the same evaluation budget. In this small experiment setup, $\delta = 0.3$ seems to be the most promising choice.

From these results, we can conclude that using the concept of dominance based on partial orders on uncertainty intervals seems indeed a viable way to do uncertainty handling. However, the results do not show whether this approach can outperform a well chosen fixed sample size resampling scheme on a fixed evaluation budget. Also a more fine-grained tuning of this approach remains to be done.

5.4.4 Selection Through Racing

Heidrich-Meisner and Igel [HMI09a] suggest the use of so-called *Hoeffding* and *Bernstein Races* [MM94, MM97] for handling noisy fitness evaluations. They proposed their approach in the context of policy learning and incorporated it in the CMA-ES. In [HMI09a], it is stated that the goal of the noise handling scheme is to ensure with a given confidence that the μ selected individuals from the population are indeed the μ best. To achieve this, they 1) control the overall number of evaluations, and 2) control the distribution of evaluations among the individuals in the population. Note that this two-phase distinction is identical to the distinction between duration scheduling and sample allocation made by Aizawa and Wah [AW94].

The evaluation/selection procedure as proposed in [HMI09a] uses confidence bounds based on Hoeffding's or Bernstein's inequality as described in Technical Note 5.7. The underlying assumption of this approach is that the measured objective function values $\tilde{f}_{i,j}$ of a candidate solution $\mathbf{x}_i$ are bounded within known bounds $[a, b]$. Given this assumption, Hoeffding's or Bernstein's inequality can be used to construct confidence bounds for the estimate of the sample mean of each individual in the population when having multiple objective function

Technical Note 5.7: Hoeffding and Bernstein Bounds

Given m noisy evaluations of individual i, $\tilde{f}_{i,1}, \ldots, \tilde{f}_{i,m}$ and the sample mean $\bar{f}_i$ of these m samples. Furthermore, assume that the fitness value is almost surely bounded within the interval $[a, b]$, i.e., $\Pr(\tilde{f}_i \in [a, b]) \approx 1$. Using Hoeffding's inequality we can state that

$$\Pr\left(\left|\bar{f}_i - \mathbf{E}[\tilde{f}_i]\right| \geq R\right) \leq 2\exp\left(-\frac{2R^2 m}{(b-a)^2}\right). \tag{5.42}$$

Using this, we can state that with a probability of at least $1-\delta$ it holds that

$$\left|\bar{f}_i - \mathbf{E}[\tilde{f}_i]\right| \leq (b-a)\sqrt{\frac{\ln\frac{2}{\delta}}{2m}}. \tag{5.43}$$

A more general bound can be obtained by using the *empirical Bernstein bound*, which uses the empirical standard deviation, obtained through $\hat{\sigma}_i^2 = \frac{1}{m}\sum_{j=1}^{m}\left(\tilde{f}_{i,j} - \bar{f}_i\right)^2$. For this, it holds with a probability of $1-\delta$ that

$$\left|\bar{f}_i - \mathbf{E}[\tilde{f}_i]\right| \leq \hat{\sigma}_i\sqrt{\frac{2\ln\frac{3}{\delta}}{m}} + \frac{3(b-a)\ln\frac{3}{\delta}}{m}. \tag{5.44}$$

Hence, using the Hoeffding or the Bernstein inequality, we can compute a confidence interval $[\bar{f}_i - c_i, \bar{f}_i + c_i]$, with

$$c_i^{\text{Hoeffding}} = (b-a)\sqrt{\frac{\ln\frac{2}{\delta}}{2m}}, \tag{5.45}$$

$$c_i^{\text{Bernstein}} = \hat{\sigma}_i\sqrt{\frac{2\ln\frac{3}{\delta}}{m}} + \frac{3(b-a)\ln\frac{3}{\delta}}{m}. \tag{5.46}$$

Technical Note 5.8: Selection Through Races

Given for each individual $i = 1, \ldots, \lambda$ object variables $\mathbf{x}_i$, absolute lower and upper bound a and b of the fitness values, a maximal evaluation budget per individual m_{limit}, the number of to be selected individuals μ, and required confidence level δ.

1. Let $S = \emptyset$, $D = \emptyset$, and $U = \{1, \ldots, \lambda\}$ be respectively the set of selected, discarded, and undecided individuals. Evaluate each individual once, $\tilde{f}_{i,1} = f(\mathbf{x}_i)$ for $i = 1, \ldots, \lambda$, initialize the lower and upper bounds for each individual, $LB_i = a$, $UB_i = b$ for $i = 1, \ldots, \lambda$, initialize the sample counter $m = 1$, and let $u_m = |U|$.

2. Set $m = m + 1$, and let $u_m = |U|$. Then, for each of the undecided individuals $i \in U$, reevaluate once $\tilde{f}_{i,m} = f(\mathbf{x}_i)$ and recompute the sample mean $\bar{f}_i = \frac{1}{m} \sum_{j=1}^{m} \tilde{f}_{i,j}$.

3. For each undecided individual $i \in U$, compute a new confidence interval $\left[\bar{f}_i - c_i, \bar{f}_i + c_i, \right]$ using the Hoeffding or Bernstein bound requiring a confidence of $1 - \delta/n_b$, $n_b = \sum_{j=1}^{m-1} u_j + (m_{\text{limit}} - m + 1)u_m$ [a]. Update the lower and upper bound based on the new confidence interval: $LB_i = \max\{LB_i, \bar{f}_i - c_i\}$, $UB_i = \min\{UB_i, \bar{f}_i + c_i\}$.

4. For each of the undecided individuals $i \in U$:

   - If $|\{j \in U | LB_i < UB_j\}| \geq \lambda - \mu - |D|$, then individual i is probably among the best μ, so add it to the set of selected individuals $S = S \cup \{i\}$ and remove it from the set of undecided individuals $U = U \backslash \{i\}$.
   - If $|\{j \in U | UB_i < LB_j\}| \geq \mu - |S|$, then individual i is probably not among the best μ, so add it to the set of discarded individuals $D = D \cup \{i\}$ and remove it from the set of undecided individuals $U = U \backslash \{i\}$.

5. Repeat step 3 to 5 until $|S| = \mu$ or $m = m_{\text{limit}}$.

6. Adapt m_{limit}, if $|S| = \mu$ then $m_{\text{limit}} = \max\{3, (1/\alpha) \cdot m_{\text{limit}}\}$, else $m_{\text{limit}} = \min\{m_{\max}, \alpha \cdot m_{\text{limit}}\}$. Use $\{\bar{f}_1, \ldots, \bar{f}_\lambda\}$ as the fitness values for selection and m_{limit} as the updated evaluation limit.

[a] In order to assure for n estimated intervals a $1 - \delta$ confidence, we require a confidence of $1 - \delta/n$ for each individual interval (Boole's inequality). Given that $n_{b,i}$ denotes the total number of computed intervals for individual i after the full evaluation loop, then there are $n = n_{b,1} + \ldots + n_{b,\lambda}$ estimated bounds in total. As a (worst-case) estimate for n, $n_b = \sum_{j=1}^{m-1} u_j + (m_{\text{limit}} - m + 1)u_m$ can be used.

evaluations.

A procedure named *race* (see Technical Note 5.8) is proposed that incorporates an in-generation resampling loop that uses these confidence bounds. At the start of the racing procedure, a confidence bound is computed for each individual. Thereafter, within the resampling loop, the individuals that are marked *undecided* are reevaluated and their confidence bounds are updated. An individual is qualified as undecided when it does not belong to the μ best (*selected*) or the $\lambda - \mu$ worst (*discarded*) individuals. By applying resampling on the undecided individuals, the confidence bounds become tighter. The resampling loop is repeated until there are μ individuals marked as selected, meaning that there is sufficient belief (i.e., a confidence of $1-\delta$) that the μ best individuals are indeed the μ best. Or, in terms of the dominance relation on intervals proposed by Rudolph [Rud01], resampling is done on the non-dominated solutions until μ or less non-dominated solutions remain.

Within the *race* procedure, an upper evaluation limit m_{limit} for each individual is used, which is required for obtaining a finite value for n_b. If this limit is reached, the race is stopped and the best μ individuals are selected based on the sample mean. The limit is updated (i.e., increased or decreased with a factor α) each generation of the evolution cycle, based on whether the full budget m_{limit} is used. Besides that, an absolute evaluation limit $m_{\max}$ is included to prevent the sample size from getting too large.

As alternative for the *race* procedure Heidrich-Meisner [HM11] proposed the so-called ϵ-*race* procedure. In this adapted version, the limits m_{limit} and $m_{\max}$ are not required. Instead of resampling each of the undecided individuals once each racing step, each individual is reevaluated $\theta(t)-\theta(t-1)$ times in the tth racing step, with $\theta(t) = t^2$. Besides that, the required confidence for the nth computed bound is set to $\delta_n = c\delta/n^2$, with $c = 6/\pi^2$. Incorporating these changes removes the need for a fixed maximum race length. Finally, the notion of ϵ-similarity condition was introduced in order to avoid long races. That is, the racing-loop is stopped when the difference between the highest upper bound and the lowest lower bound of the individuals that are still undecided drops below a certain threshold ϵ.

A downside of using Bernstein and Hoeffding bounds is that these require known bounds for the objective function values. This means that either the noise should be bounded within known bounds or the bounds of the objective function should be known. If both are unknown, which is the scenario that we consider in this work, then these bounds should be estimated or the objective function should be transformed as described in [HM11, p. 112].

A more serious issue is that the Hoeffding and Bernstein bounds, as used in the racing procedures, are based on assumed bounds on the objective function and not (or hardly) on the measured noise. For instance, when using the Hoeffding bound in a racing procedure, it does not matter whether the noise is very small or very high, the Hoeffding bound is only based on the assumed bounds on the full domain of the objective function. The Bernstein bound does consider the sample variance, but still contains a large factor (the rightmost term in Eq. 5.46)

that is not based on it. This is conceptually very undesirable and in this work a reason not to consider this noise handling technique as suitable option.

Although studying modifications of these techniques falls beyond the scope of this work, two options could be tried to fix the aforementioned problems. First, the racing procedures could be adapted when the noise is bounded within known bounds (i.e., $z \in [a, b]$). In that case, the bounds for each individual $i = 1, \ldots, \lambda$ could be initialized separately, based on one evaluation of the objective function, yielding possibly much tighter bounds. Second, for Gaussian noise, an alternative to using Bernstein and Hoeffding bounds is to use the more classical type of confidence bounds within the same selection procedure. Given the selection procedure based on races (Technical Note 5.8), an alternative would therefore be to use Gaussian confidence intervals, as considered by Rudolph [Rud01] (see Eq. 5.39).

5.4.5 Rank-Change Based Uncertainty Measures

Hansen et al. [HNGK09] propose a scheme for handling noisy objective functions implemented within the CMA-ES; the *Uncertainty Handling* CMA-ES (UH-CMA-ES). Although originally proposed in the context of an application to feedback control of combustion, the main concepts can be applied in more general scenarios, as shown by Heidrich-Meisner and Igel [HMI09b]. Moreover, the uncertainty handling scheme can be applied within any rank-change based optimization algorithm.

The uncertainty handling scheme separates two components: the quantification of the uncertainty and the treatment of the uncertainty. That is, the evaluation intensity/accuracy is increased or decreased based on measurements of the impact of the noise on the selection.

The **uncertainty quantification** is based on counting the number of rank-changes that occur when reevaluating (a part of) the population. If the number of rank-changes after reevaluation is high, then it can be assumed that the uncertainty is high and the noise should be reduced. If there are only few rank-changes, then the uncertainty is low and a higher noise level may be allowed. The procedure is described in detail in Technical Note 5.9.

The **uncertainty treatment** scheme used in [HNGK09] consists of two methods: 1) increasing the evaluation time t_{eval}, and 2) increasing the population variance by increasing the stepsize. The former is specifically suitable for the problem considered in [HNGK09], as it is possible to increase the accuracy of the fitness function by increasing the measuring time. The latter is a secondary treatment method, used when the evaluation time has reached its maximum $t_{\max}$. For the uncertainty treatment, as suggested in [HNGK09], a cumulated version $\bar{s}$ of the uncertainty measure s is introduced, updated every generation using $\bar{s} = (1 - c_s)\bar{s} + c_s s, c_s \in [0, 1]$. Whenever $\bar{s}$ is greater than zero, the evaluation time is increased with a factor of α_t.

For selection, the solutions are re-ranked according to their rank sum: $\mathrm{rank}\,(L_i^{\mathrm{new}}) + \mathrm{rank}\left(L_i^{\mathrm{old}}\right)$. Ties are resolved firstly using the absolute rank change $|\Delta_i|$, using for the not reevaluated solutions: $\Delta_i = (1/\lambda_{\mathrm{reev}}) \sum_{j=1}^{\lambda_{\mathrm{reev}}} |\Delta_j|$, secondly using the sample mean.

Technical Note 5.9: Rank-Change Based Uncertainty Quantification

For each solution $\mathbf{x}_i$, $i = 1, \ldots, \lambda$ an approximation of its fitness is obtained, i.e.,

$$L_i^{\text{old}} = f(\mathbf{x}_i). \tag{5.47}$$

Then, a parameter λ_{reev} is computed using the parameter r_λ (recommended $r_\lambda = 0.3$), with $\lambda_{\text{reev}} = f_{\text{pr}}(r_\lambda \cdot \lambda)$, where the function $f_{\text{pr}} : \mathbb{R} \to \mathbb{Z}$ is defined as

$$f_{\text{pr}}(x) = \begin{cases} \lfloor x \rfloor + 1 & \text{with probability } x - \lfloor x \rfloor \\ \lfloor x \rfloor & \text{otherwise} \end{cases}. \tag{5.48}$$

Furthermore, λ_{reev} is set to 1 whenever it has been set to 0 for more than $2/(r_\lambda \cdot \lambda)$ consecutive generations. The first λ_{reev} solutions are selected for reevaluation, i.e.,

$$L_i^{\text{new}} = \begin{cases} f(\mathbf{x}_i) & \text{if } i \leq \lambda_{\text{reev}} \\ L_i^{\text{old}} & \text{otherwise} \end{cases}. \tag{5.49}$$

For each solution $\mathbf{x}_i$, the rank change Δ_i is computed as

$$\Delta_i = \text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}}) - \text{signum}\left(\text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}})\right), \tag{5.50}$$

where $\text{rank}(L_i)$ is the rank of function value L_i in the set $\mathcal{L} = \{L_k^{\text{old}}, L_k^{\text{new}} | k = 1, \ldots, \lambda\}$. Hence, Δ_i counts the number of values from the set $\mathcal{L} \backslash \{L_i^{\text{old}}, L_i^{\text{new}}\}$ that lie between L_i^{old} and L_i^{new}. Based on the individual rank-changes, the uncertainty level s is determined as

$$\begin{aligned} s \quad = \quad & \frac{1}{\lambda_{\text{reev}}} \sum_{i=1}^{\lambda_{\text{reev}}} \left(2|\Delta_i| \right. \\ & -\Delta_\theta^{\text{lim}} \left(\text{rank}\left(L_i^{\text{new}}\right) - \mathbb{I}\{L_i^{\text{new}} > L_i^{\text{old}}\}\right) \\ & \left. - \Delta_\theta^{\text{lim}} \left(\text{rank}\left(L_i^{\text{old}}\right) - \mathbb{I}\{L_i^{\text{old}} > L_i^{\text{new}}\}\right)\right), \end{aligned} \tag{5.51}$$

where $\Delta_\theta^{\text{lim}}(R)$ is the $\theta \times 50\%$ percentile of the set $\{|1 - R|, |2 - R|, \ldots, |2\lambda - 1 - R|\}$. It represents the rank change for a given rank R that would occur when given a completely random function and is a reference for Δ_i. The indicator function $\mathbb{I}$ returns 1 if its argument is true, otherwise 0.

Algorithm 5.2: Rank-Change Based Adaptive Averaging

Procedure parameters: confidence level θ, averaging increment factor α

Procedure variables: sample size indicator m_{eval}, initialized at $m_{\text{eval}} = 2$

1. For all candidate solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ obtain $m_1 = \lceil m_{\text{eval}}/2 \rceil$ noisy objective function evaluations

$$\tilde{f}_{i,j} = \tilde{f}(\mathbf{x}_i)\ ,\ i = 1, \ldots, \lambda\ ,\ j = 1, \ldots, m_1, \tag{5.52}$$

   compute the mean objective function value $\bar{f}_{i,\text{old}}$ for each individual $i = 1, \ldots, \lambda$ based on this sample set, and store them in the set $L^{\text{old}} = \{\bar{f}_{1,\text{old}}, \ldots, \bar{f}_{\lambda,\text{old}}\}$.

2. Repeat step 1 using $m_2 = \lfloor m_{\text{eval}}/2 \rfloor$ and store the mean objective function values in the set $L^{\text{new}} = \{\bar{f}_{1,\text{new}}, \ldots, \bar{f}_{\lambda,\text{new}}\}$.

3. Compute the rank-changes $\Delta_1, \ldots, \Delta_\lambda$ using:

$$\Delta_i = \text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}}) - \text{signum}\left(\text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}})\right). \tag{5.53}$$

4. Compute the uncertainty level based on the rank-changes

$$\begin{aligned} s \;=\; & \frac{1}{\lambda_{\text{reev}}} \sum_{i=1}^{\lambda_{\text{reev}}} \big(2|\Delta_i| \\ & -\Delta_\theta^{\lim}\left(\text{rank}\left(L_i^{\text{new}}\right) - \mathbb{I}\{L_i^{\text{new}} > L_i^{\text{old}}\}\right) \\ & -\ \Delta_\theta^{\lim}\left(\text{rank}\left(L_i^{\text{old}}\right) - \mathbb{I}\{L_i^{\text{old}} > L_i^{\text{new}}\}\right)\big). \end{aligned} \tag{5.54}$$

5. Update the sample size m_{eval} using the update rule

$$m_{\text{eval}} = \begin{cases} \alpha \cdot m_{\text{eval}} & ,\text{if } s > 0 \\ m_{\text{eval}} & ,\text{otherwise} \end{cases}. \tag{5.55}$$

6. Generate a ranking $\mathbf{x}_{1:\lambda}, \ldots, \mathbf{x}_{\lambda:\lambda}$ based on the sample means $(\bar{f}_{1,\text{old}} + \bar{f}_{1,\text{new}})/2, \ldots, (\bar{f}_{\lambda,\text{old}} + \bar{f}_{\lambda,\text{new}})/2$.

In this work we consider the algorithm as described in Algorithm 5.2 as an implementation of the rank-change based uncertainty handling scheme. It is an adapted version of the approach proposed in [HNGK09], which is done to allow for resampling, but also to make it such that this scheme differs from Algorithm 5.1 only in the uncertainty indicator. The latter is done for the sake of comparison of the two approaches.

Two issues that are deliberately left out are the decrement of m_{eval} when the uncertainty level is small (i.e., $s < 0$), and the upper limit on the sample size. The former is done under the assumption that the sample size should only increase during an optimization run. The latter is done in order test the uncertainty handling mechanism itself, eliminating side-effects that are due to sample size limits. To obtain an insight in the behavior of this rank-change based uncertainty handling scheme we perform the following experiment:

Experiment 5.4.2 (Performance of rank based adaptive averaging on the noisy sphere problem)**:** We perform 10 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) incorporating the evaluation procedure of Algorithm 5.2 (named UH-$(5/2_{DI}, 35)$-σSA-ES) on the 10-dimensional noisy sphere problem (see Appendix A.1). We take parameter setting $\alpha = 1.5$ and use varying $\theta = 0.5, 0.7, 0.9$. As benchmark, we include a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Each run uses a budget of $100,000$ objective function evaluations.

The results of Experiment 5.4.2 are presented in Figure 5.10 and Figure 5.11. Figure 5.10 shows the convergence dynamics of the three instances of this adaptive averaging approach, and as a benchmark the dynamics of a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Figure 5.11 shows the single run and average dynamics of the three instances of this adaptive averaging scheme. The left column shows the distance to the optimizer versus the number of generations, the middle column the development of sample size parameter m_{eval}, and the right column the development of the uncertainty indicator.

The results of Experiment 5.4.2 are similar to the results of Experiment 5.4.1. As can be seen in Figure 5.11, increasing the strictness of the uncertainty indicator yields a quicker growth of the uncertainty level and the sample size, allowing for fewer generations. Also in this case there is a trade-off between allowing uncertainty and depending on the averaging effects of multiple generations or requiring a strict confidence in order to obtain a high progress rate per generation.

5.4.6 Rank-Inversions Based Adaptive Averaging

An alternative to the rank-change based uncertainty measure (see Eq. 5.51 in Technical Note 5.9) is to count rank inversions [Mar01, KEB09a]. For this measure, the distribution is known, and normal for $\lambda \to \infty$. This allows for a better founded measure of uncertainty which could be argued to be simpler to implement as compared to the rank-change based measure. Technical

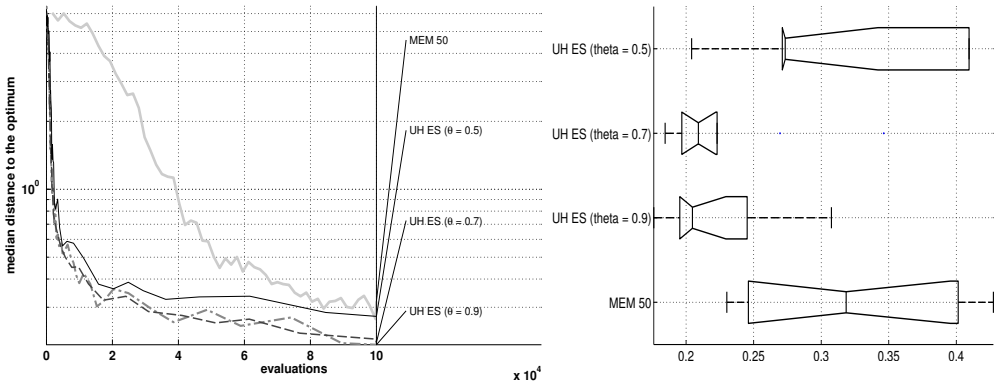


Figure 5.10: Left: The convergence dynamics (median over 10 runs) of the UH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem using $\alpha = 1.5$ and using varying $\theta = 0.5, 0.7, 0.9$, compared against a fixed sample size resampling scheme with $m = 50$. Right: boxplots of the final solution quality.

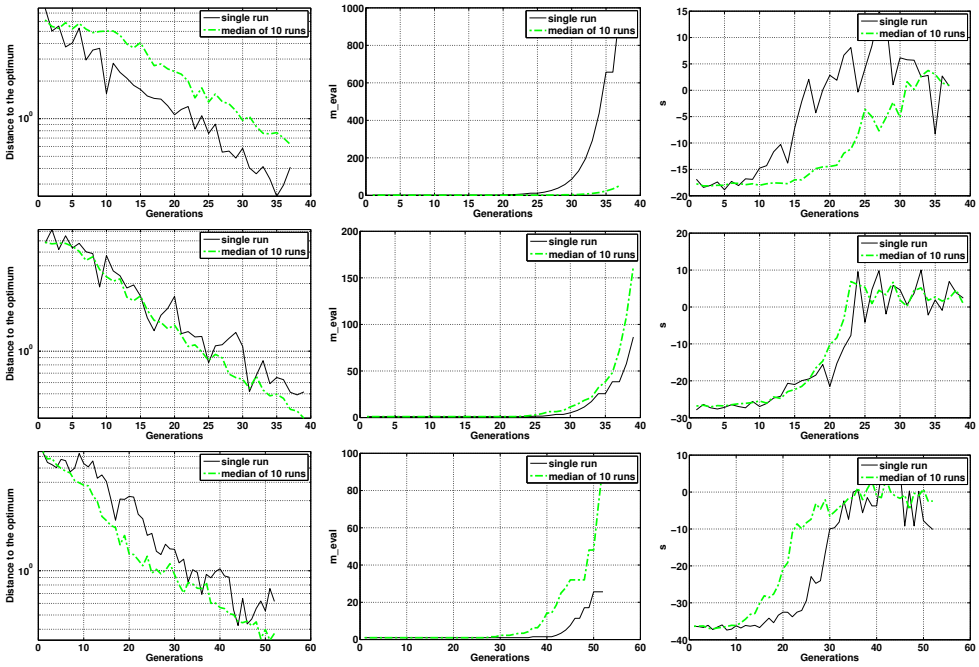


Figure 5.11: The dynamics (median over 10 runs) of the UH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem, using different confidence levels $\theta = 0.5$ (top row), $\theta = 0.7$ (middle row), $\theta = 0.9$ (bottom row). Left: the distance to the optimizer versus number of generations. Center: the development of m_{eval}. Right: the development of the uncertainty level (i.e., the rank-change based indicator).

Technical Note 5.10: Rank-Inversion Based Uncertainty Measure

Given a population of λ candidate solutions, let $\mathrm{rank}_i^{\mathrm{old}}$ denote the rank of solution i based on the fitness values of the first evaluation step and let $\mathrm{rank}_i^{\mathrm{new}}$ denote the rank of solution i based on the fitness values of the reevaluation step. The number of inversions is computed as

$$\mathrm{Inv}_\lambda = \sum_{(i,j)\in\{1,\ldots,\lambda\}^2} I(i,j), \tag{5.56}$$

$$I(i,j) = \begin{cases} 0 & \text{, if } (\mathrm{rank}_i^{\mathrm{old}} < \mathrm{rank}_i^{\mathrm{old}}) \wedge (\mathrm{rank}_i^{\mathrm{new}} > \mathrm{rank}_i^{\mathrm{new}}) \\ 1 & \text{, otherwise} \end{cases}. \tag{5.57}$$

Let ξ_λ denote a random variable measuring the number of inversions for a pure random ordering. The number of inversions for randomly generated perturbations follows a normal distribution for $\lambda \to \infty$ with mean $\mathbf{E}[\xi_\lambda] = \lambda(\lambda-1)/4$ and its variance is $\mathrm{Var}[\xi_\lambda] = (2\lambda^3 + 3\lambda^2 - 5\lambda)/72$ (see, [Mar01]).

Using this, an uncertainty measure s can be constructed as

$$s_{\mathrm{Inv}} = \mathrm{Inv}_\lambda - \left(\mu_{\mathrm{Inv}} + \sigma_{\mathrm{Inv}} \cdot \Phi^{-1}(\theta)\right), \tag{5.58}$$

with $\mu_{\mathrm{Inv}} = \lambda(\lambda-1)/4$, $\sigma_{\mathrm{Inv}} = \sqrt{(2\lambda^3 + 3\lambda^2 - 5\lambda)/72}$, and $\Phi^{-1}(.)$ being the inverse cumulative distribution function of the standard normal distribution.

Note 5.10 describes how this measure can be used to obtain a similar, but alternative uncertainty measure for the rank-change based uncertainty handling method of Algorithm 5.2.

In order to test this alternative uncertainty measure, we consider it to be incorporated in the procedure of Algorithm 5.2 and use it instead of the rank-change based uncertainty measure. For this adapted scheme we run the following experiment:

Experiment 5.4.3 (Performance of inversions based adaptive averaging on the noisy sphere problem)**:** We perform 10 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) incorporating the evaluation procedure of Algorithm 5.2 that uses the alternative uncertainty measure of Technical Note 5.10. This scheme is named IUH-$(5/2_{DI}, 35)$-σSA-ES, and the experiments are performed on the 10-dimensional noisy sphere problem (see Appendix A.1). We take parameter setting $\alpha = 1.5$ and use varying $\theta = 0.3, 0.5, 0.7$. As benchmark, we include a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Each run uses a budget of $100,000$ objective function evaluations.

The results of Experiment 5.4.3 are presented in Figure 5.12 and Figure 5.13. Figure 5.12 shows the convergence dynamics of the three instances of this adaptive averaging scheme compared to the convergence dynamics of a $(5/2_{DI}, 35)$-σSA-ES using a fixed sample size resampling scheme with $m = 50$. Figure 5.13 shows the single run and average dynamics of the three instances of this adaptive averaging scheme. The left column shows the distance to

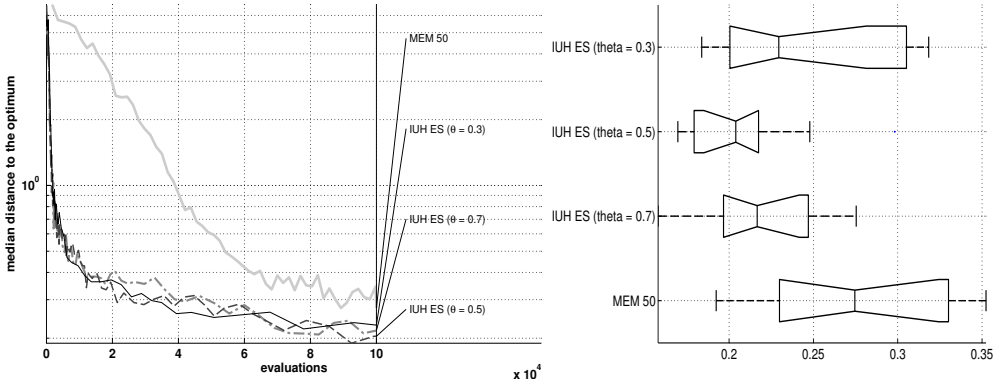


Figure 5.12: Left: The convergence dynamics (median over 10 runs) of the IUH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem using $\alpha = 1.5$ and using varying $\theta = 0.3, 0.5, 0.7$, compared against a fixed sample size resampling scheme with $m = 50$. Right: boxplots of the final solution quality.

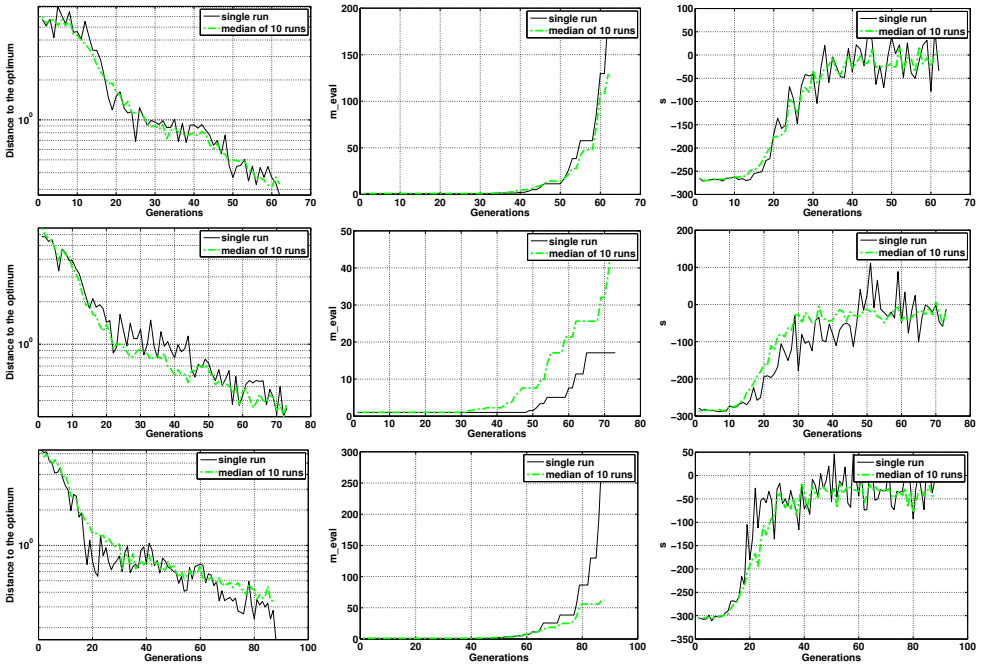


Figure 5.13: The dynamics (median over 10 runs) of the IUH-$(5/2_{DI}, 35)$-σSA-ES on the noisy sphere problem, using different confidence levels $\theta = 0.3$ (top row), $\theta = 0.5$ (middle row), $\theta = 0.7$ (bottom row). Left: the distance to the optimizer versus number of generations. Center: the development of m_{eval}. Right: the development of the uncertainty level (i.e., the inversions based indicator).

the optimizer versus the number of generations, the middle column the development of sample size parameter m_{eval}, and the right column the development of the uncertainty indicator.

The results of Experiment 5.4.3 are similar to the results of Experiment 5.4.2 of the rank-change based uncertainty handling approach. As can be seen in Figure 5.13, increasing the strictness of the uncertainty indicator yields a quicker growth of the uncertainty level and the sample size, allowing for fewer generations. The uncertainty indicator is, however, not so strict as the rank-change based uncertainty measure when using the same value for θ, though from these results this seems to be the only difference.

5.4.7 A Discussion on Adaptive Noise Handling Techniques

Comparing the adaptive averaging techniques discussed in this chapter, we observe the following:

Uncertainty quantification and uncertainty treatment: All schemes either implicitly or explicitly distinguish between uncertainty quantification and uncertainty handling. The term *uncertainty quantification* regards the decision mechanism that determines whether or not to increase the evaluation accuracy of the underlying noise treatment mechanism. The term *noise treatment* refers to the underlying noise handling mechanism. All except one of the adaptive averaging techniques that have been discussed perform uncertainty treatment using resampling (or explicit averaging). Interestingly, the alternative of increasing the population size is not considered in any of the adaptive averaging techniques.

In-generation and inter-generation mechanisms: There are two different types adaptation mechanisms; *in-generation* and *inter-generation* mechanisms. In-generation mechanisms are used in, e.g., the sample allocation scheme, t-test based adaptive resampling, and the races based approaches. Here, the uncertainty is targeted directly by continuing the resampling procedure until the uncertainty level is sufficiently reduced. In inter-generation mechanisms, present in, e.g., the duration scheduling and rank-change based uncertainty handling mechanism, the uncertainty treatment is adapted after each generation based on the previous uncertainty quantification. Inter-generation methods are based on trusting the evolution process to be partially robust against disturbances in the selection that are higher than the desired level indicated by the uncertainty quantification as long as the evaluation intensity of the following generations is increased. Although in-generation methods are more direct, an advantage of using inter-generation methods is that these are less sensitive to scenarios as observed in the t-test based approach where many samples are spend on trying to distinguish between two solutions while their difference might be of no importance in the perspective of the current stage of the optimization. Note that both mechanisms can be used within the same method. For instance, the races based approach uses both mechanisms.

Evaluation intensity limitations: Most adaptive averaging methods incorporate an absolute upper limit for the evaluation intensity. For instance, the races based approach, or the rank-

change based adaptive averaging method use a maximum sampling limit and a maximum evaluation time respectively. Moreover, a scheme in which the sample size is allowed to grow without limit, being the scheme discussed with the t-test based approach, suffers from a rapid sample size explosion. Apparently there are practical reasons for bounding the evaluation intensity. However, using such bounds is undesirable from a theoretical perspective, because it introduces limitations on the convergence accuracy.

Underlying assumptions: When looking at the assumptions on which the uncertainty handling methods are based (e.g., the type of noise), we see that there are only two uncertainty indicators that are not based on specific assumptions regarding the noise or the objective function; the rank-change and rank-inversions based uncertainty measure. Besides that, the assumption that the noise is Gaussian and the assumption of having the ability to establish confidence bounds take in a prominent place.

Parameters: The in-generation adaptive averaging schemes that were discussed require at least one parameter, namely an uncertainty quantification threshold (e.g., a confidence level). Inter-generation adaptive averaging mechanisms require at least two parameters, namely an uncertainty quantification threshold and a scaling factor for the evaluation intensity (e.g., a growth factor for the sample size in explicit averaging). Moreover, when including upper limits on the evaluation intensity or cumulation of the uncertainty quantification, this number increases rapidly. In the methods summarized so far, the parameters were mostly set based on empirical testing.

In conclusion, Table 5.2 shows the different adaptive averaging techniques, summarized with respect to the assumptions on which they are based, the used uncertainty quantification measure, and the used uncertainty handling method. Considering the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, the partial order based adaptive averaging (*PUH*, Section 5.4.3), rank-change based adaptive averaging (*UH*, Section 5.4.5), and rank-inversions based adaptive averaging or (*IUH*, Section 5.4.6) provide the most promising alternatives to explicit and implicit averaging. In the remainder of this work, these schemes will be studied in more depth for their practical applicability. Two essential, but yet unanswered questions are:

- For each uncertainty quantification measure, at what uncertainty level do Evolution Strategies still have a positive expected progress and which uncertainty level is optimal?
- For inter-generation adaptive averaging methods, at what rate should the evaluation intensity increase in order to allow Evolution Strategies to progress?

5.5 Metamodel Assisted Noise Handling

Another class of techniques for dealing with noisy objective functions in the context of Evolutionary Algorithms is formed by techniques that construct a surrogate model (or metamodel) of

Noise handling scheme	Assumptions	Uncertainty quantification	Uncertainty treatment
Duration scheduling [AW93, AW94]	The "real" underlying fitness values of the population are normally distributed and the noise is Gaussian	Ratio between population variance and approximation error	Resampling
Sample allocation [AW93, AW94]	The "real" underlying fitness values of the population are normally distributed and the noise is Gaussian	The probability of each individual to be the best	Resampling
t-Test based resampling [Sta98, CP04, KEB09a]	Gaussian noise	Ranking confidence among all/some pairs of individuals based on the t-test	Resampling
Partial order based selection [Rud01]	The noise is bounded or Gaussian noise is assumed, utilizing confidence bounds	Non-dominance ranking on interval orders / acceptance threshold	Resampling
Races based uncertainty handling [HMI09a]	The objective function values are bounded within known bounds	Ranking confidence among all individuals based on the Hoeffding or Bernstein bound	Resampling
Rank-change based uncertainty handling [HNGK09]	No assumptions	Rank changes after reevaluation of (part of) the individuals in the population	Increasing evaluation time
Rank-inversions based uncertainty handling	No assumptions	Rank inversions after reevaluation of the individuals in the population	Resampling

Table 5.2: The different adaptive averaging techniques, summarized with respect to the assumptions which they are based on, the uncertainty quantification measure that is used, and the uncertainty handling method that is used.

the underlying objective function based on the full history of noisy measurements, and perform optimization on this model. Such approaches aim to use all information that is available about the objective function. Methods that are based on this idea were proposed by Sano and Kita [SK00, SK02], and Branke et al. [BSS01]. Though these schemes are not the main focus of this work, this section will provide a brief technical summary.

5.5.1 Memory-Based Fitness Estimation

In [SK00, SK02], Sano and Kita propose an approach named Memory-Based Fitness Estimation (MFE) (see Technical Note 5.11). The core idea of the approach is that the objective function value of candidate solution $\mathbf{x}^*$, given fitness observation $\tilde{f}^*$, can be estimated by means of a maximum likelihood approach based on Gaussian model assumptions and the assumption that there is a spatial relation between $\mathbf{x}^*$ and an archive of previous objective function measurements $A = \{(\mathbf{x}_i, \tilde{f}_i) \,|\, i = 1, \ldots, L\}$. In the approach, the fitness of every individual in the population is computed following this method. The archive is in this approach filled with objective function measurements of the previous populations.

In [SK02], an adaptation is proposed to account for situations where the objective function estimate of a candidate solutions differed too much from the measured objective function value. In order to prevent these cases, it is proposed to reject the candidate solutions in a population of which the measured objective function values differed more than a threshold Z from the best measured objective function value. In [SK02], the threshold Z is recommended to be set such that the probability of such errors to occur is less than 0.3.

5.5.2 Local Regression Based Fitness Estimation

Branke et al. [BSS01] propose a similar approach, only they consider a model that is based on different assumptions as compared to the approach of Sano and Kita [SK00, SK02]. In their approach, they consider stationary Gaussian noise and optimization of the real underlying objective function.

Technical Note 5.12 describes the general approach proposed in [BSS01]. The modeling assumption is that the objective function can be locally approximated by a low order polynomial function. They propose to estimate the objective function value of each candidate solution by building a local model based on an archive of previously evaluated points, weighting each archive solution based on the distance to the to-be-evaluated solution. The choices that remain open are to determine the degree of the polynomial used for regression, the choice for the neighborhood parameter h that assigns a weighting contribution for each archive solution based on the distance to the to-be-evaluated solution, and the way in which the regression model is fitted. In [BSS01], a quadratic model is used. As an extension, it is also suggested to use the information of the local models more extensively by performing local hill-climbing on the model to locally improve candidate solutions.

Technical Note 5.11: Memory-Based Fitness Estimation

Assume that the objective function value of each candidate solution is normally distributed and consider the following modeling assumption:

$$f_j \sim \mathcal{N}(f_i, kd_{ij}), \tag{5.59}$$
$$\tilde{f}_j \sim \mathcal{N}(f_i, kd_{ij} + \sigma_\epsilon^2). \tag{5.60}$$

Here, f_i and f_j denote the true objective function values of candidate solutions $\mathbf{x}_i$ and $\mathbf{x}_j$, k is some constant, and d_{ij} denotes the distance d_{ij} between $\mathbf{x}_i$ and $\mathbf{x}_j$ (i.e., $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$). Hence, the true objective function value at $\mathbf{x}_j$ is assumed to be distributed normally random around the true objective function value of $\mathbf{x}_i$, proportional to the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Given known values of k and σ_ϵ^2, a maximum likelihood estimation approach can be used to estimate the true objective function value f^* of candidate solution $\mathbf{x}^*$ based on its own objective function measurement $\tilde{f}^*$ and an archive of previously observed measurements $A = \{(\mathbf{x}_i, \tilde{f}_i), i = 1, \ldots, L\}$. When given f^*, the probability of obtaining $\tilde{f}_1, \ldots, \tilde{f}_L$ is expressed by

$$\prod_{i=1}^{L} p(\tilde{f}_i, d_i), \text{ where } p(\tilde{f}_i, d_i) = \frac{1}{\sqrt{2\pi(kd_i + \sigma_\epsilon^2)}} \exp\left(-\frac{1}{2} \frac{\left(\tilde{f}_i - f^*\right)^2}{kd_i + \sigma_\epsilon^2} \right). \tag{5.61}$$

Here, d_i denotes the distance between $\mathbf{x}^*$ and $\mathbf{x}_i$. One can maximize this expression for f^*, from which we obtain

$$\hat{f}^* = \frac{\tilde{f}^* + \sum_{i=1}^{L} \frac{\sigma_\epsilon^2}{kd_i + \sigma_\epsilon^2} \tilde{f}_i}{1 + \sum_{i=1}^{L} \frac{\sigma_\epsilon^2}{kd_i + \sigma_\epsilon^2}}. \tag{5.62}$$

The model parameters k and σ_ϵ^2 can be estimated using Eq. 5.61 by maximization of the log-likelihood

$$\underset{k, \sigma_\epsilon^2}{\operatorname{argmax}} \left\{ -\frac{1}{2} \left(L \log 2\pi + \sum_{i=1}^{L} \log(kd_i + \sigma_\epsilon^2) + \sum_{i=1}^{L} \frac{(\tilde{f}_i - f^*)^2}{kd_i + \sigma_\epsilon^2} \right) \right\}. \tag{5.63}$$

That is, these estimates are taken from the perspective of one candidate solution $\mathbf{x}^*$, with true objective function value f^*. In [SK00, SK02], the best individual of the current population is recommended to be used for this estimation procedure and its true fitness is recommended to be estimated by averaging the objective function values of the five closest individuals. A hill-climbing method acting on a logarithmic space of k and σ_ϵ^2 should be used for maximization of the log-likelihood[a].

[a] In [SK02], an exact derivation for the maximum log-likelihood was suggested with respect to σ_ϵ^2, but this derivation is incorrect.

Technical Note 5.12: Local Regression Based Fitness Estimation

Assume that the fitness of individual i is normally distributed, i.e.,

$$\tilde{f}_i \sim \mathcal{N}(f_i, \sigma_\epsilon^2). \tag{5.64}$$

Furthermore, assume that the "real" underlying objective function $f(\mathbf{x})$ can be locally approaximated by means of a low order polynomial function.

For a candidate solution $\mathbf{x}^*$, construct a locally weighted regression model $g_{\mathbf{x}^*,h}(\mathbf{x})$ based on an archive of previously observed measurements $A = \{(\mathbf{x}_i, \tilde{f}_i), i = 1, \ldots, L\}$ and use as objective function approximation:

$$\hat{f}_{\exp_i} = g_{\mathbf{x}_i,h}(\mathbf{x}^*). \tag{5.65}$$

The locally weighted regression model $g_{\mathbf{x}_i,h}(\mathbf{x})$ is based on a weight function $w_h(d)$, assigning a weight to the contribution of each archive point based on the Euclidean distance d between the archive point and $\mathbf{x}^*$. The weight function considered in [BSS01] is the tri-cube function

$$w_h(d) = \begin{cases} (1 - d^3/h)^3 & ,\text{if } d < h \\ 0 & ,\text{otherwise} \end{cases}. \tag{5.66}$$

The parameter h is called the neighborhood parameter and reflects the size of the neighborhood for which the assumptions can be considered to be valid.

For setting h, two methods are considered: 1) choosing it such that 5% of the archive points are considered to be neighboring points, 2) choosing h such that the following cross-validation criterion is minimized (computed using a numerical hill-climber):

$$CV(\hat{g}_{\mathbf{x}*,h}) = \frac{\sum_{i=1}^{L} w_h(d_i) \left(\tilde{f}_i - \hat{g}^{-i}_{\mathbf{x}*,h}\right)^2}{\sum_{i=1}^{L} w_h(d_i)}. \tag{5.67}$$

.

5.5.3 A Discussion on Metamodeling Noise Handling Techniques

The validity of these metamodeling approaches depends on three key issues: 1) the extent to which the modeling assumptions are valid, 2) the quality of the archive with respect to the to-be-estimated objective function value, and 3) the accuracy of the tuning of the model parameters. Moreover, it is well-known that for higher dimensional search spaces, metamodeling becomes increasingly more difficult due to the *curse of dimensionality* (see, e.g., [FSK08, p. xvii]).

Regarding the validity of the modeling assumptions, we note that this depends purely on the optimization problem at hand. Assuming a Gaussian spatial correlation, as done by Sano and Kita [SK00, SK02], is not uncommon and used, for instance, also in Kriging (see, e.g., [SWMW89, JSW98, FSK08]). The same holds for the assumption that the objective function landscape can be locally approximated using a polynomial model, as done by Branke et al. [BSS01].

The quality of the archive is in this context an issue of a different kind. For the estimation of the objective function for a given candidate solution, ideally, the archive should contain data points that lie well-spread around the given candidate solution. However, when the archive is straightforwardly built up as the history of candidate solutions obtained by the evolutionary process of an Evolutionary Algorithm itself, this is not necessarily achieved. See [KEB10] for an example where straightforward utilization of an archive within an Evolutionary Algorithm is outperformed by a more careful archive maintenance approach. The approaches discussed here do not actively try to maintain a "proper" archive with respect to the to-be-estimated solution qualities, yet the improvements proposed in [SK02] are based on negative effects that can be related to a poor archive quality.

The accuracy of the tuning of the model parameters of the MFE approach of [SK00, SK02] is questionable. That is, these parameters are estimated from the perspective of the observed best candidate solution, using a crude estimate of its real objective function value. The approach of Branke et al. [BSS01] has a more solid mathematical basis, yet also requires to tune a correlation distance.

5.6 A General Discussion of Noise Handling Techniques

In the previous sections, the working mechanisms of a number of noise handling techniques have been summarized. In this review we have made the categorization of basic noise handling techniques, adaptive averaging techniques, and metamodel assisted noise handling techniques. The basic noise handling techniques provide the basic techniques on how to reduce the undesirable effects of noise. Adaptive averaging techniques aim to automatically adapt the parameters of static noise handling techniques. Metamodeling techniques, on the other hand, attempt to build a (local) surrogate model of the original objective function, therewith aiming to generate a near noise-free surrogate objective function.

Although implicit and explicit averaging are easier to implement than all other schemes presented, different noise handling techniques seem to be preferable for three reasons:

1. Maintaining arbitrary convergence accuracy.

2. Costly evaluations require efficient usage of the evaluation budget.

3. Eliminating the need to specify sensitive noise-handling parameters.

The first reason regards the drawback of implicit and explicit averaging to target the optimum with arbitrary precision. However, many adaptive averaging techniques have serious difficulties with a seemingly exploding number of required evaluations. The races and rank-change based uncertainty handling techniques therefore use upper bounds on the evaluation intensity, which in turn bounds the convergence accuracy.

The second, aiming for saving costly function evaluations, has a more practical basis. This can be seen as motivation for the races and rank-change based uncertainty handling techniques that by using upper limits on the evaluation effort, therewith aiming to achieve a better convergence accuracy within less time than a static noise handling technique. Metamodeling techniques are constructed for this practical purpose too. The computational demands of metamodeling techniques makes them suitable only in the cases where evaluations are costly, but the alleged gain is that a reduced number of objective function evaluations are acquired.

The third regards the ideal not to have to set parameters like the sample size for explicit resampling beforehand. When looking at the adaptive averaging and metamodeling techniques that have been discussed in the previous section, none of them manages to accomplish this. Yet, the parameters that are replaced, for instance, in the rank-change based uncertainty handling technique could be less sensitive than using a fixed sample size.

Based on the former observations, we conclude that for an advanced noise handling technique to be practically viable, it should be an improvement with respect to implicit or explicit averaging in either of the following scenarios:

1. **Arbitrary convergence accuracy:** the advanced noise handling technique can be proven to maintain the global convergence criterion.

2. **Sampling efficiency:** given an arbitrary, but fixed evaluation budget, the advanced noise handling technique should ideally outperform any static averaging technique.

3. **Parameter reduction:** the advanced noise handling technique should either be completely parameter-less (*strict parameter reduction*), or given canonical settings, it should outperform any fixed static averaging scheme on a majority of problems (*weak parameter reduction*).

Of the approaches presented in literature, we can say that *arbitrary convergence accuracy* does not hold for implicit and explicit averaging nor for adaptive averaging techniques that limit the evaluation accuracy. For the adaptive averaging techniques that do not limit the evaluation accuracy and for the metamodeling approaches, *arbitrary convergence accuracy* remains to be proven. For all adaptive averaging techniques considered in this chapter, *sampling efficiency* remains to be proven. For all adaptive averaging techniques and metamodeling noise handling techniques, strict *parameter reduction* is not achieved and *weak parameter reduction* remains to be proven.

5.7 Summary and Discussion

In this chapter we have reviewed the problem of optimization of noisy objective functions using Evolution Strategies from the perspective of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES.

There are different goals for optimization of noisy objective functions, which are modeled differently in an effective objective function formulation. Choosing an effective objective function is a design issue that depends on the problem at hand. In systems with intrinsic additive noise, optimization of the expected objective function is most customary. In systems in which the noise is due to errors in measurement, optimization of the real underlying objective function is more appropriate. When the noise is stationary, these two goals are equivalent. When the noise is non-stationary, other options are also reasonable.

Evolution Strategies are fairly robust against noise when considering the expected objective function as an optimization goal. However, noise limits the convergence accuracy of Evolution Strategies and countermeasures are needed when higher accuracy is desired.

The way in which to adapt Evolution Strategies in order to deal with noisy objective functions depends on what is aimed for. In the second part of this chapter, a number of noise handling techniques have been described, categorized as: basic noise handling, adaptive averaging, and metamodel assisted noise handling. Explicit and implicit averaging techniques can be used to improve the convergence accuracy of Evolution Strategies, but they also suffer from convergence accuracy limitations. When arbitrary convergence accuracy is aimed for, adaptive averaging techniques or metamodeling techniques should be used.

The question which of the techniques considered in this chapter is most suitable depends on the particular type of noise. Moreover, even when considering the specific case of stationary Gaussian noise, the question remains open which of the advanced noise handling schemes is the best. Suitable techniques for the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES are the partial order based adaptive averaging technique (PUH), the rank-change based adaptive averaging technique (UH), and the inversion-based adaptive averaging technique (IUH). We will consider these three adaptive averaging methods next to implicit and explicit averaging as the most appropriate candidates for noise handling. For these techniques, the questions that remain are: 1) How should the algorithmic parameters of these adaptive averaging methods be set? 2) How

do these techniques compare against each other and how do these techniques compare against their static counterparts; implicit and explicit averaging? In the next chapter we will study these issues in more detail.

Chapter 6

A Study on Noise Handling Schemes

In Chapter 5, three adaptive averaging techniques are identified as suitable and promising techniques for noise handling in the context of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. These techniques are: poset based adaptive averaging (PUH, Section 5.4.3), rank-change based adaptive averaging (UH, Section 5.4.5), and rank-inversions based adaptive averaging (IUH, Section 5.4.6). All three are based on the same inter-generation adaptive averaging framework and use the same noise treatment approach of explicit averaging (or resampling). The difference between them is that they are based on different uncertainty quantification methods.

In this chapter, we aim to study these approaches in more depth. In particular, we will aim to answer the following questions: 1) At what rate should the sample size grow when using adaptive resampling for optimization of noisy objective functions? 2) What parameter settings are appropriate for the adaptive averaging techniques considered in this work? 3) Does using the adaptive averaging techniques considered in this chapter yield better results than using explicit or implicit averaging?

This chapter is structured as follows: Section 6.1 presents a theoretical study on the ideal growth of the sample size on a simple test problem to gain insight in the dynamics of an optimally functioning adaptive averaging scheme. Section 6.2 presents the results of an empirical study that aims to gain insight into the algorithmic parameters of the adaptive averaging techniques. Section 6.3 presents the results of an empirical study in which the adaptive averaging techniques are compared to the standard techniques of explicit and implicit averaging. Section 6.4 closes with a summary and discussion.

6.1 The Growth Rate of the Sample Size

The inter-generation adaptive averaging techniques considered in this chapter all use a multiplicative update rule for adapting the number of samples when the uncertainty level is too high. In literature, the increment factor for updating the sample size is regarded as an algorithmic parameter that is to be tuned manually. However, two questions that remain are:

should the update rule indeed be exponential and, if so, at what rate should it ideally grow? This section studies how the sample size should optimally develop within an Evolution Strategy, the $(\mu/\mu_I, \lambda)$-ES, on a simple artificial test problem, the noisy sphere problem. This section builds forth on the results obtained by Arnold and Beyer [AB02].

The motivation behind using an adaptive sample size for resampling strategies is to adapt the number of samples used for evaluating each individual such that it is sufficient for the Evolutionary Algorithm to progress. On the other hand, it should not be too high, because then samples will be wasted. This way of formulating the problem of adaptive averaging suggests that, when an Evolution Strategy is in a certain state of the evolution process, there is an optimal sample size. Ideally, an adaptive averaging scheme is able to follow this optimal sample size over the course of evolution.

6.1.1 The Progress Rate of the $(\mu/\mu_I, \lambda)$-ES on the Noisy Sphere

In the analysis of Evolution Strategies, the *progress rate* is an often used performance measure. The progress rate, denoted φ, is defined as the expected distance covered by the centroid of the population towards the location of the optimum within one generation. That is, when R denotes the distance to the optimum location of the centroid of the parents, and when r denotes the distance to the optimum location of the centroid of the selected offspring, then the progress rate reads $\varphi = \mathbf{E}[R - r]$. The progress rate can be used to determine the *efficiency per evaluation* when dividing it by the number of evaluations used for the current generation. By optimizing the efficiency per evaluation with respect to the sample size, we can derive the optimal sample size for an Evolution Strategy at a given stage in the optimization.

The noisy sphere problem is given by the function

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + \sigma_\epsilon z_\epsilon \to \min\ ,\ z_\epsilon \sim \mathcal{N}(0,1), \tag{6.1}$$

mapping n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ to a noisy objective function value with Gaussian noise with standard deviation σ_ϵ. For this model, Arnold and Beyer [AB02] derived for the $(\mu/\mu_I, \lambda)$-ES the normalized progress rate

$$\varphi^*_{\mu/\mu,\lambda} \simeq \frac{c_{\mu/\mu,\lambda}\sigma^* \left(1 + \sigma^{*2}/(2\mu n)\right)}{\sqrt{1 + \sigma^{*2}/(\mu n)}\sqrt{1 + \upsilon^2 + \sigma^{*2}/(2n)}} - n\left[\sqrt{1 + \frac{\sigma^{*2}}{\mu n}} - 1\right], \tag{6.2}$$

where $c_{\mu/\mu,\lambda}$ is the $(\mu/\mu_I, \lambda)$-progress coefficient and φ^* is the normalized progress rate

$$\varphi^* = \varphi\frac{n}{R}. \tag{6.3}$$

Furthermore, σ^*_ϵ is a normalized version of σ_ϵ, according to normalization

$$\sigma^*_\epsilon = \sigma_\epsilon \frac{n}{2R^2}, \tag{6.4}$$

σ^* is a normalized version of the stepsize parameter σ, reading

$$\sigma^* = \sigma \frac{n}{R}, \tag{6.5}$$

and υ is the noise-to-signal ratio, defined as

$$\upsilon = \frac{\sigma^*_\epsilon}{\sigma^*}. \tag{6.6}$$

6.1.2 The Efficiency of Resampling

Resampling with m samples reduces the error of the objective function approximations by a factor of $\sqrt{m}$, that is, when using m samples, we have an effective approximation error of

$$\hat{\sigma}_\epsilon = \sigma_\epsilon / \sqrt{m}. \tag{6.7}$$

Hence, resampling increases the progress rate φ^* (see Eq. 6.2), however, at the cost of requiring more evaluations. In order to determine the efficiency, the progress rate should be divided by the number of evaluations used for the current generation, i.e., by λm.

When letting $\varphi_{\hat{\sigma}_\epsilon}$ denote the progress rate for noise factor $\hat{\sigma_\epsilon}$, we can state the efficiency η_m of a certain sample size m as

$$\eta_m = \frac{\varphi_{\hat{\sigma}_\epsilon}}{\lambda m}. \tag{6.8}$$

For the $(\mu/\mu_I, \lambda)$-ES on the noisy sphere problem, when substituting Eq. 6.2 into this equation and using $\upsilon = \sigma^*_\epsilon / (\sqrt{m}\sigma^*)$ (i.e., using the effective approximation error due to resampling), this yields a normalized efficiency

$$\begin{aligned} \eta^*_m &= \frac{\frac{c_{\mu/\mu,\lambda}\sigma^*\left(1+{\sigma^*}^2/(2\mu n)\right)}{\sqrt{1+{\sigma^*}^2/(\mu n)}\sqrt{1+\sigma^{*2}_\epsilon/(m{\sigma^*}^2)+{\sigma^*}^2/(2n)}} - n\left[\sqrt{1+\frac{{\sigma^*}^2}{\mu n}}-1\right]}{\lambda m} && (6.9) \\ &= \frac{c_{\mu/\mu,\lambda}\sigma^*\left(1+{\sigma^*}^2/(2\mu n)\right)}{\lambda m\sqrt{1+{\sigma^*}^2/(\mu n)}\sqrt{1+\sigma^{*2}_\epsilon/(m{\sigma^*}^2)+{\sigma^*}^2/(2n)}} \\ &\quad - \frac{n\left[\sqrt{1+\frac{{\sigma^*}^2}{\mu n}}-1\right]}{\lambda m}. && (6.10) \end{aligned}$$

6.1.3 The Optimal Sample Size

The optimal sample size for Evolution Strategies at a given stage of the optimization can be determined by maximizing the efficiency with respect to m. That is,

$$m_{\text{opt}} = \text{argmax}_m \eta^*_m. \tag{6.11}$$

When omitting the fact that in practice m can only assume positive integer values, this can be done by solving

$$\frac{\partial \eta^*_m}{\partial m} = 0. \tag{6.12}$$

In order to do so, we apply the substitutions

$$c_1 = c_{\mu/\mu,\lambda}\,\sigma^* \left(\frac{\sigma^{*2}}{2\,\mu\,n} + 1\right), \tag{6.13}$$

$$c_2 = \lambda \sqrt{\frac{\sigma^{*2}}{\mu\,n} + 1}, \tag{6.14}$$

$$c_3 = 1 + \frac{\sigma^{*2}}{2n}, \tag{6.15}$$

$$c_4 = \sigma_\epsilon^{*2}/\sigma^{*2}, \tag{6.16}$$

$$c_5 = \frac{n\left[\sqrt{1+\frac{\sigma^{*2}}{\mu n}} - 1\right]}{\lambda}. \tag{6.17}$$

This yields

$$\eta_m^* = \frac{c_1}{m c_2 \sqrt{c_3 + c_4/m}} - \frac{c_5}{m}, \tag{6.18}$$

and

$$\frac{\partial \eta_m^*}{\partial m} = \frac{2\,c_2\,c_5\,m\left(\frac{c_3\,m+c_4}{m}\right)^{3/2} - 2\,c_1\,c_3\,m - c_1\,c_4}{2\,c_2\,m^3\left(\frac{c_3\,m+c_4}{m}\right)^{3/2}}. \tag{6.19}$$

Hence, to solve $\frac{\partial \eta_m^*}{\partial m} = 0$ for $m \geq 1$, we need to solve

$$2\,c_2\,c_5\,m\left(\frac{c_3\,m+c_4}{m}\right)^{\frac{3}{2}} - 2\,c_1\,c_3\,m - c_1\,c_4 = 0. \tag{6.20}$$

This equation has only one real solution, namely

$$m_{\text{opt}} = -\frac{c_4(3c_2^2c_3c_5^2 - c_1^2)}{3c_3(c_2^2c_3c_5^2 - c_1^2)} - \frac{(c_1^6c_3^3c_4^3 + 36c_1^4c_2^2c_3^4c_4^3c_5^2 + 27c_1^2c_2^4c_3^5c_4^3c_5^4 + c_6)^{1/3}}{6c_3^2(c_2^2c_3c_5^2 - c_1^2)} \tag{6.21}$$

$$+\frac{(-4c_1^4c_3^2c_4^2 - 60c_1^2c_2^2c_3^3c_4^2c_5^2)}{24c_3^2(c_2^2c_3c_5^2 - c_1^2)(c_1^6c_3^3c_4^3 + 36c_1^4c_2^2c_3^4c_4^3c_5^2 + 27c_1^2c_2^4c_3^5c_4^3c_5^4 + c_6)^{1/3}}, \tag{6.22}$$

with

$$c_6 = 3\sqrt{3}\sqrt{c_1^{10}c_2^2c_3^7c_4^6c_5^2 + 25c_1^8c_2^4c_3^8c_4^6c_5^4 - 53c_1^6c_2^6c_3^9c_4^6c_5^6 + 27c_1^4c_2^8c_3^{10}c_4^6c_5^8}. \tag{6.23}$$

As can be seen, this expression is quite involved and hard to simplify any further. Yet, it is an exact derivation with respect to Eq. 6.20. Moreover, when assuming that σ^* is constant (which is realistic when assuming a properly functioning adaptation mechanism of the stepsize) and using the substitutions Eq. 6.17 and Eq. 6.4, it can be shown that

$$m_{\text{opt}} \propto \Theta\left(\frac{1}{R^4}\right). \tag{6.24}$$

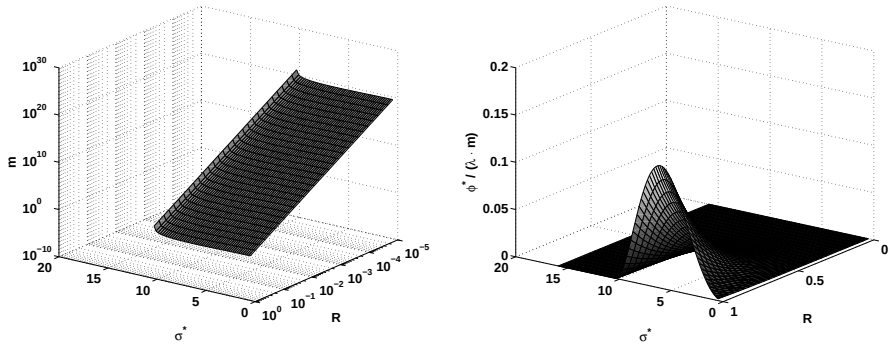


Figure 6.1: The derived optimal sample size (left) and the efficiency $\eta^* = \varphi^*/(\lambda m)$ obtained for the optimal sample size (right) for the $(\mu/\mu_I, \lambda)$-ES for different values of R and σ^*. These results are obtained with $\mu = 8$, $\lambda = 32$, $\sigma_\epsilon = 1$, $n = 10$.

That is, the sample size should grow quartically with the inverse distance to the optimum. Intuitively, this result is plausible, because the resampling reduces the noise with a factor of $\sqrt{m}$ and for the sphere problem the local fitness differences decrease quadratically with R.

In order to get a picture of the development of m_{opt} for different values of R and σ^*, Figure 6.1 shows the derived optimal sample size and the efficiency $\eta^* = \varphi^*/(\lambda m)$ obtained for the optimal sample size for the $(\mu/\mu_I, \lambda)$-ES for different values of R and σ^*. These results are obtained with $\mu = 8$, $\lambda = 32$, $\sigma_\epsilon = 1$, $n = 10$. The slope of the log-log plot of m_{opt} in the direction of R is approximately 4, which is an empirical indicator to support the result of Eq. 6.24. Furthermore, we see that even with an optimally adapted sample size, the efficiency drops quickly for decreasing R (which can largely be attributed to the fast growth of m_{opt}), and seems optimal for $\sigma^* \approx 5$.

6.1.4 The Minimally Required Sample Size

Similarly, a lower bound for m can be derived. That is, the minimally required number of samples to maintain positive progress. This requires solving $\eta^*_m = 0$ for m. For this we obtain

$$m_{\text{least}} = -\frac{c_2^2 c_4 c_5^2}{c_2^2 c_3 c_5^2 - c_1^2} \tag{6.25}$$

$$= \frac{-{\sigma_\epsilon}^2 n^2 \left(\sqrt{\frac{\sigma n^2}{\mu R}+1}-1\right)^2 \left(\frac{\sigma^2 n}{\mu R^2}+1\right)}{4\sigma^2 \left(n^2 \left(\sqrt{\frac{\sigma n^2}{\mu R}+1}-1\right)^2 \left(\frac{\sigma^2 n}{2R^2}+1\right)\left(\frac{\sigma^2 n}{\mu R^2}+1\right) - \frac{\log\left(\frac{\lambda}{\mu}\right)\sigma^2 n^2 \left(\frac{\sigma^2 n}{2\mu R^2}+1\right)^2}{R^2}\right) R^2}. \tag{6.26}$$

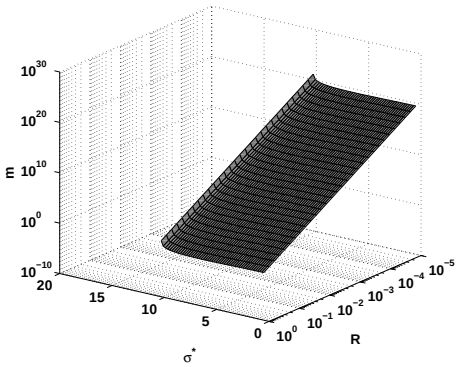


Figure 6.2: The derived minimally required number of samples for the $(\mu/\mu_I, \lambda)$-ES for different values of R and σ^*. These results are obtained with $\mu = 8$, $\lambda = 32$, $\sigma_\epsilon = 1$, $n = 10$.

When assuming that σ^* is constant then we obtain

$$m_{\text{least}} = -\frac{{\sigma_\epsilon}^2 \left(\frac{\sigma^{*2}}{\mu n} + 1\right) n^4 \left(\sqrt{\frac{\sigma^* n}{\mu} + 1} - 1\right)^2}{4\sigma^{*2} \left(\left(\frac{\sigma^{*2}}{2n} + 1\right)\left(\frac{\sigma^{*2}}{\mu n} + 1\right) n^2 \left(\sqrt{\frac{\sigma^* n}{\mu} + 1} - 1\right)^2 - {c_{\mu/\mu,\lambda}}^2 \sigma^{*2} \left(\frac{\sigma^{*2}}{2\mu n} + 1\right)^2\right) R^4}. \tag{6.27}$$

Hence, also for m_{least} we can conclude that, when σ^* is constant,

$$m_{\text{least}} \propto \Theta\left(\frac{1}{R^4}\right). \tag{6.28}$$

Hence, the sample size should grow at least quartically with the inverse distance to the optimum to maintain positive progress. In order to get an impression of the development for different R and σ^*, Figure 6.2 shows the derived minimal required number of samples for the $(\mu/\mu_I, \lambda)$-ES (see, Eq. 6.26) for different values of R and σ^*. These results are obtained with $\mu = 8$, $\lambda = 32$, $\sigma_\epsilon = 1$, $n = 10$. Comparing Figure 6.2 to Figure 6.1, we observe practically the same results. This is not surprising, given that $\frac{1}{R^4}$ is the dominating term of both m_{opt} and m_{least} when n, μ, λ, and σ^* are relatively small.

6.1.5 The Growth Rate of the Sample Size

We consider $\varphi^* = (n\varphi)/R$ and take the optimistic assumption that $\varphi*$ is constant (obtained when the stepsize adaptation mechanism functions optimally). Let R_t denote the distance to the optimum of the centroid of the population at time t. We can express the expected distance to the optimum $\mathbf{E}[R_{t+1}]$ of the centroid of the next generation, at time $t+1$ in terms of φ. I.e.,

$$\mathbf{E}[R_{t+1}] = R_t - \varphi = R_t - \frac{R_t \varphi^*}{n} = R_t - kR_t, \tag{6.29}$$

where $0 < k < 1$ is some constant. We can derive the *generation-wise* limit behavior as

$$\lim_{R_t \to 0} \frac{\mathbf{E}[R_{t+1}]}{R_t} = \lim_{R_t \to 0} \frac{R_t - kR_t}{R_t} = 1 - k < 1. \tag{6.30}$$

Hence, the convergence behavior is linear with respect to the number of generations.

When considering the progress per evaluation using adaptive resampling, we use the efficiency, defined as $\eta_m = \varphi/(\lambda m)$. Considering an optimally tuned adaptive resampling mechanism with $m = k_m/R^4$, k_m being some constant. Still assuming that $\varphi*$ is constant, we can use $m = k_m/R^4$ and obtain

$$\eta_m = \frac{\varphi}{\lambda(k_m/R^4)} = \frac{R^5\varphi^*}{n\lambda k_m} = kR^5. \tag{6.31}$$

where $0 < k < 1$ is some constant. We can express the expected distance to the optimum $\mathbf{E}[R_{t+1}]$ of the centroid of the next *evaluation*, at time $t + 1$, in terms of η as

$$\mathbf{E}[R_{t+1}] = R_t - \eta_m = R_t - kR_t^5. \tag{6.32}$$

For this, we can derive the *evaluation-wise* limit behavior as

$$\lim_{R_t \to 0} \frac{\mathbf{E}[R_{t+1}]}{R_t} = \lim_{R_t \to 0} \frac{R_t - kR_t^5}{R_t} = 1. \tag{6.33}$$

Hence, for an optimally adapted sample size, the convergence rate of a $(\mu/\mu_I, \lambda)$-ES on the noisy sphere is sublinear with respect to the number of objective function evaluations.

Finally, using $\mathbf{E}[R_{t+1}] = (1 - k)R_t$, $m_t = c/R_t$, and $m_{t+1} = k/\mathbf{E}[R_{t+1}]^4$, we obtain the following growth of the sample size when assuming linear convergence of R with respect to the number of generations:

$$\lim_{R_t \to 0} \frac{m_{t+1}}{m_t} = \lim_{R_t \to 0} \frac{R_t^4}{\mathbf{E}[R_{t+1}]^4} = \lim_{R_t \to 0} \frac{R_t^4}{((1-k)R_t)^4} = \frac{1}{(1-k)^4} > 1. \tag{6.34}$$

In conclusion, the sample size should indeed grow exponentially with respect to the number of generations to achieve a generation-wise linear convergence rate.

6.2 Tuning the Adaptive Averaging Methods

The previous analysis confirmed that the sample size within an adaptive resampling scheme should grow exponentially when generation-wise linear convergence is desired. However, the rate at which it should grow is not yet determined and also the uncertainty threshold for the adaptive averaging techniques remains to be set. To gain insight into how these parameters should be set, we perform an empirical study on each of the adaptive averaging techniques (the PUH, the UH, and the IUH scheme) incorporated in the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. We consider different instances of these schemes, varying the uncertainty thresholds δ or θ and the resampling factor α.

The experiments are performed on the 10-dimensional noisy sphere problem and a large evaluation budget of $100,000$ function evaluations is considered. For the uncertainty handling schemes, we consider for the uncertainty thresholds δ or θ the values $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and for the uncertainty treatment parameter we consider the values $\alpha \in \{1.1, 1.3, 1.5, 1.7, 1.9\}$. This yields 25 combinations of settings for each scheme for both the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. Each run for each instance is repeated 10 times.

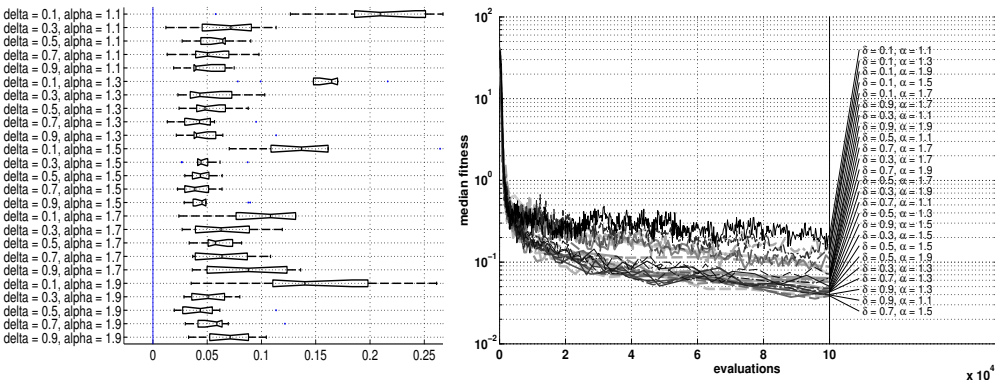
The results of the experiments are shown in Section 6.2.1 and Section 6.2.2 for the PUH-scheme, Section 6.2.3 and Section 6.2.4 show the results of the UH-scheme, and Section 6.2.5 and Section 6.2.6 show the results of the IUH-scheme. For each scheme, the boxplot shows the final solution quality after $100,000$ objective function evaluations, which is computed a posteriori using the noise-free signal function as measure of the expected objective function. The line in the boxplot indicates the objective function value of the optimal solution. The other plot shows the convergence dynamics in terms of the median of the real (noise-free) objective function value of the best individual of the current population versus the number of evaluations. The table summarizes the statistics of the final solution quality. The best instance is determined based on the rank sum.

The results show that all adaptive resampling schemes are fairly robust with respect to different settings of δ/θ and α. Except for a few outlier-settings, such as $\delta = 0.1$ for the PUH approach or $\theta = 0.1$ for the UH approach, most of the tested settings yield comparable results. The optimal settings that can be derived from these results differ for each uncertainty quantification scheme and also for the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. The results of these experiments are summarized in Table 6.1.

	$(5/2_{DI}, 35)$-σ**SA-ES**	**CMA-ES**
PUH	$\delta = 0.7$, $\alpha = 1.5$	$\delta = 0.9$, $\alpha = 1.3$
UH	$\theta = 0.9$, $\alpha = 1.1$	$\theta = 0.9$, $\alpha = 1.9$
IUH	$\theta = 0.3$, $\alpha = 1.9$	$\theta = 0.1$, $\alpha = 1.3$

Table 6.1: The optimal settings (with an approximate error of ± 0.1) of the uncertainty threshold δ/θ and the resampling rate α to achieve the best convergence accuracy on a budget of $100,000$ function evaluations.

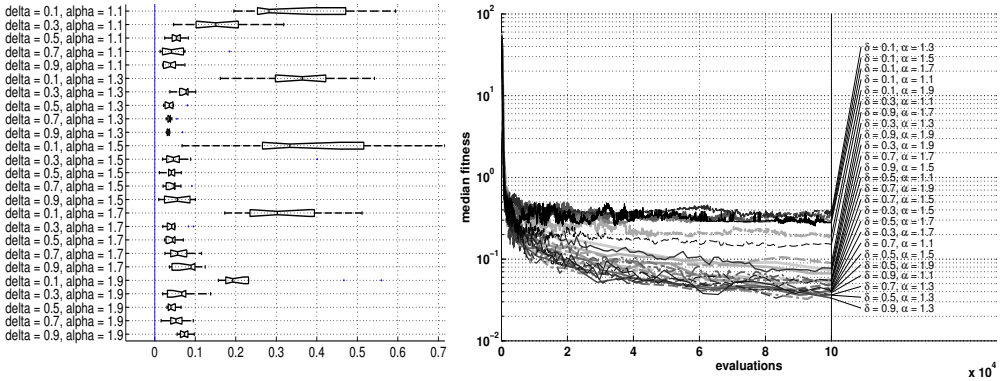
6.2.1 Results Tuning PUH-$(5/2_{DI}, 35)$-σSA-ES on the Noisy Sphere



NOISY SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
delta = 0.1, alpha = 1.1	5.29	16.12	0.21	2235	25
delta = 0.3, alpha = 1.1	0.07	0.03	0.07	1312	17
delta = 0.5, alpha = 1.1	4.19	13.08	0.06	1302	16
delta = 0.7, alpha = 1.1	0.05	0.02	0.05	1034	12
delta = 0.9, alpha = 1.1	0.05	0.02	0.04	852	6
delta = 0.1, alpha = 1.3	0.17	0.07	0.16	2200	24
delta = 0.3, alpha = 1.3	0.05	0.03	0.04	895	9
delta = 0.5, alpha = 1.3	0.05	0.02	0.05	1016	11
delta = 0.7, alpha = 1.3	0.04	0.02	0.04	752	2
delta = 0.9, alpha = 1.3	0.05	0.03	0.04	858	7
delta = 0.1, alpha = 1.5	0.15	0.07	0.14	2133	23
delta = 0.3, alpha = 1.5	0.05	0.02	0.04	829	5
delta = 0.5, alpha = 1.5	0.04	0.01	0.04	783	3
delta = 0.7, alpha = 1.5	**0.04**	**0.01**	**0.04**	**654**	**1**
delta = 0.9, alpha = 1.5	0.05	0.02	0.05	893	8
delta = 0.1, alpha = 1.7	0.14	0.12	0.11	1839	21
delta = 0.3, alpha = 1.7	0.07	0.03	0.06	1237	14
delta = 0.5, alpha = 1.7	0.06	0.02	0.06	1249	15
delta = 0.7, alpha = 1.7	4.37	13.61	0.06	1357	18
delta = 0.9, alpha = 1.7	0.09	0.04	0.09	1552	20
delta = 0.1, alpha = 1.9	0.17	0.09	0.14	2038	22
delta = 0.3, alpha = 1.9	0.05	0.02	0.05	988	10
delta = 0.5, alpha = 1.9	0.05	0.03	0.04	809	4
delta = 0.7, alpha = 1.9	0.06	0.03	0.06	1106	13
delta = 0.9, alpha = 1.9	4.65	14.48	0.07	1452	19

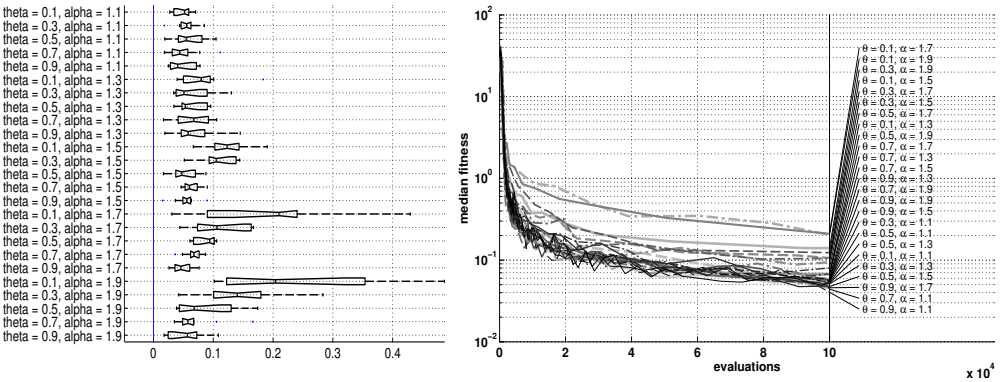
6.2.2 Results Tuning PUH-CMA-ES on the Noisy Sphere



NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
delta = 0.1, alpha = 1.1	0.35	0.14	0.28	2260	24
delta = 0.3, alpha = 1.1	0.16	0.09	0.15	1843	20
delta = 0.5, alpha = 1.1	0.05	0.02	0.05	1044	14
delta = 0.7, alpha = 1.1	0.05	0.05	0.04	839	8
delta = 0.9, alpha = 1.1	0.04	0.02	0.04	707	4
delta = 0.1, alpha = 1.3	0.36	0.12	0.36	2274	25
delta = 0.3, alpha = 1.3	0.07	0.02	0.08	1456	17
delta = 0.5, alpha = 1.3	0.04	0.02	0.04	643	3
delta = 0.7, alpha = 1.3	0.04	0.01	0.04	641	2
delta = 0.9, alpha = 1.3	**0.04**	**0.01**	**0.03**	**589**	**1**
delta = 0.1, alpha = 1.5	0.41	0.25	0.33	2231	22
delta = 0.3, alpha = 1.5	0.08	0.11	0.05	998	11
delta = 0.5, alpha = 1.5	0.04	0.01	0.04	757	5
delta = 0.7, alpha = 1.5	0.05	0.02	0.05	852	9
delta = 0.9, alpha = 1.5	0.06	0.03	0.06	1035	12
delta = 0.1, alpha = 1.7	0.32	0.11	0.30	2231	23
delta = 0.3, alpha = 1.7	0.05	0.02	0.04	855	10
delta = 0.5, alpha = 1.7	0.04	0.02	0.04	801	6
delta = 0.7, alpha = 1.7	0.06	0.03	0.06	1160	15
delta = 0.9, alpha = 1.7	0.08	0.03	0.09	1482	18
delta = 0.1, alpha = 1.9	0.25	0.14	0.19	2112	21
delta = 0.3, alpha = 1.9	0.07	0.04	0.07	1184	16
delta = 0.5, alpha = 1.9	0.04	0.01	0.04	824	7
delta = 0.7, alpha = 1.9	0.05	0.02	0.05	1042	13
delta = 0.9, alpha = 1.9	0.07	0.01	0.07	1515	19

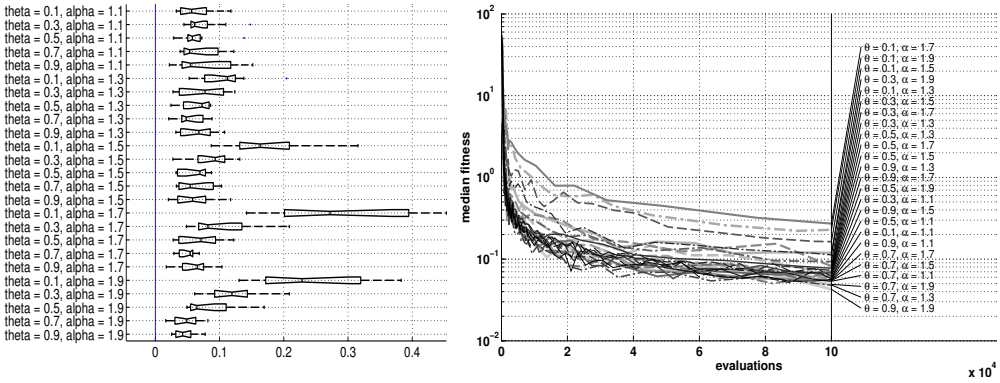
6.2.3 Results Tuning UH-$(5/2_{DI}, 35)$-σSA-ES on the Noisy Sphere



NOISY SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
theta = 0.1, alpha = 1.1	2.58	8.00	0.05	872	6
theta = 0.3, alpha = 1.1	0.06	0.02	0.06	944	8
theta = 0.5, alpha = 1.1	4.56	14.26	0.05	996	9
theta = 0.7, alpha = 1.1	0.05	0.03	0.04	730	3
theta = 0.9, alpha = 1.1	**0.05**	**0.02**	**0.04**	**659**	**1**
theta = 0.1, alpha = 1.3	0.08	0.04	0.08	1373	18
theta = 0.3, alpha = 1.3	4.40	13.73	0.05	1102	12
theta = 0.5, alpha = 1.3	0.06	0.02	0.05	1020	10
theta = 0.7, alpha = 1.3	3.01	9.31	0.07	1151	14
theta = 0.9, alpha = 1.3	0.07	0.04	0.06	1084	11
theta = 0.1, alpha = 1.5	0.12	0.04	0.12	1946	22
theta = 0.3, alpha = 1.5	4.72	14.58	0.10	1880	21
theta = 0.5, alpha = 1.5	0.05	0.02	0.05	799	4
theta = 0.7, alpha = 1.5	0.06	0.02	0.06	1154	15
theta = 0.9, alpha = 1.5	0.05	0.02	0.06	901	7
theta = 0.1, alpha = 1.7	0.20	0.12	0.21	1974	24
theta = 0.3, alpha = 1.7	3.38	10.36	0.11	1747	20
theta = 0.5, alpha = 1.7	2.81	8.62	0.09	1685	19
theta = 0.7, alpha = 1.7	0.07	0.01	0.07	1202	16
theta = 0.9, alpha = 1.7	0.05	0.02	0.05	691	2
theta = 0.1, alpha = 1.9	0.25	0.14	0.20	2235	25
theta = 0.3, alpha = 1.9	0.15	0.07	0.14	1946	23
theta = 0.5, alpha = 1.9	4.76	14.82	0.07	1292	17
theta = 0.7, alpha = 1.9	0.07	0.04	0.06	1132	13
theta = 0.9, alpha = 1.9	0.05	0.03	0.06	860	5

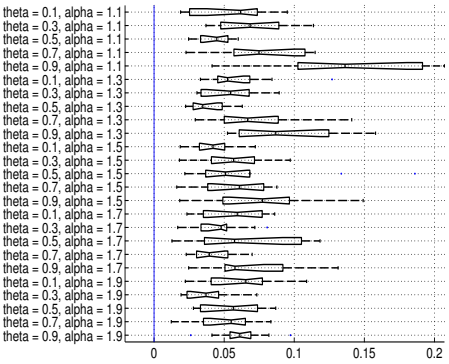
6.2.4 Results Tuning UH-CMA-ES on the Noisy Sphere

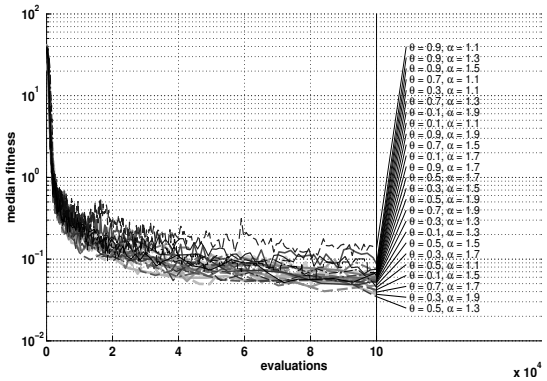


NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
theta = 0.1, alpha = 1.1	0.06	0.03	0.06	994	9
theta = 0.3, alpha = 1.1	0.07	0.03	0.06	1207	16
theta = 0.5, alpha = 1.1	0.06	0.03	0.06	958	6
theta = 0.7, alpha = 1.1	0.07	0.03	0.05	1046	11
theta = 0.9, alpha = 1.1	0.08	0.05	0.06	1118	15
theta = 0.1, alpha = 1.3	0.11	0.04	0.11	1694	21
theta = 0.3, alpha = 1.3	0.17	0.31	0.08	1265	17
theta = 0.5, alpha = 1.3	0.07	0.02	0.07	1101	14
theta = 0.7, alpha = 1.3	0.05	0.02	0.05	827	4
theta = 0.9, alpha = 1.3	0.07	0.03	0.07	1097	13
theta = 0.1, alpha = 1.5	0.18	0.07	0.16	2167	23
theta = 0.3, alpha = 1.5	0.09	0.04	0.09	1435	19
theta = 0.5, alpha = 1.5	0.06	0.02	0.07	978	7
theta = 0.7, alpha = 1.5	0.06	0.03	0.05	998	10
theta = 0.9, alpha = 1.5	0.06	0.03	0.06	912	5
theta = 0.1, alpha = 1.7	0.34	0.24	0.27	2355	25
theta = 0.3, alpha = 1.7	0.10	0.05	0.08	1573	20
theta = 0.5, alpha = 1.7	0.07	0.03	0.07	1081	12
theta = 0.7, alpha = 1.7	0.05	0.01	0.05	715	3
theta = 0.9, alpha = 1.7	0.06	0.03	0.07	979	8
theta = 0.1, alpha = 1.9	0.27	0.14	0.23	2314	24
theta = 0.3, alpha = 1.9	0.12	0.05	0.12	1875	22
theta = 0.5, alpha = 1.9	0.26	0.56	0.06	1385	18
theta = 0.7, alpha = 1.9	0.05	0.02	0.05	693	2
theta = 0.9, alpha = 1.9	**0.05**	**0.02**	**0.04**	**608**	**1**

6.2.5 Results Tuning IUH-$(5/2_{DI}, 35)$-σSA-ES on the Noisy Sphere

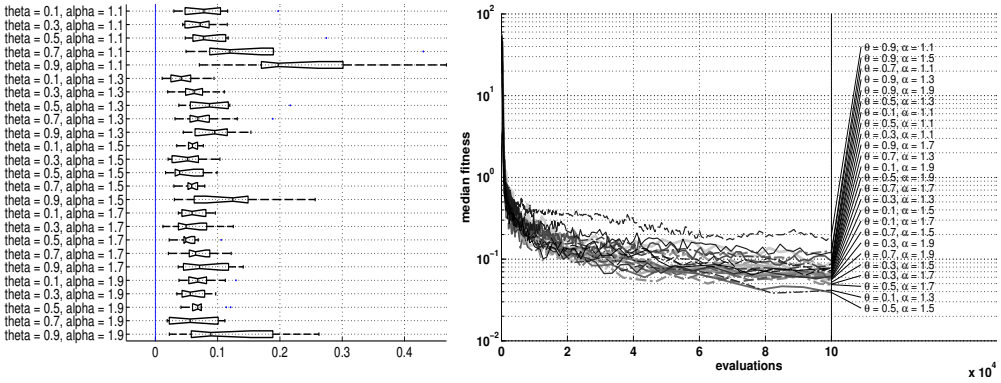




NOISY SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
theta = 0.1, alpha = 1.1	0.05	0.03	0.06	1104	7
theta = 0.3, alpha = 1.1	3.17	9.82	0.07	1517	21
theta = 0.5, alpha = 1.1	3.42	10.68	0.04	918	5
theta = 0.7, alpha = 1.1	0.08	0.03	0.07	1610	23
theta = 0.9, alpha = 1.1	0.14	0.06	0.14	2149	25
theta = 0.1, alpha = 1.3	0.06	0.03	0.05	1243	12
theta = 0.3, alpha = 1.3	0.06	0.02	0.05	1168	9
theta = 0.5, alpha = 1.3	0.04	0.01	0.03	673	2
theta = 0.7, alpha = 1.3	0.07	0.04	0.07	1508	20
theta = 0.9, alpha = 1.3	0.09	0.04	0.09	1864	24
theta = 0.1, alpha = 1.5	0.04	0.02	0.04	833	4
theta = 0.3, alpha = 1.5	4.26	13.31	0.06	1288	16
theta = 0.5, alpha = 1.5	0.07	0.05	0.05	1211	10
theta = 0.7, alpha = 1.5	0.06	0.02	0.06	1258	13
theta = 0.9, alpha = 1.5	0.08	0.04	0.08	1572	22
theta = 0.1, alpha = 1.7	0.06	0.02	0.06	1239	11
theta = 0.3, alpha = 1.7	0.05	0.02	0.05	950	6
theta = 0.5, alpha = 1.7	3.51	10.91	0.06	1336	17
theta = 0.7, alpha = 1.7	0.04	0.01	0.04	744	3
theta = 0.9, alpha = 1.7	0.07	0.03	0.06	1455	19
theta = 0.1, alpha = 1.9	0.06	0.03	0.07	1272	14
theta = 0.3, alpha = 1.9	**0.04**	**0.02**	**0.04**	**661**	**1**
theta = 0.5, alpha = 1.9	3.35	10.43	0.06	1277	15
theta = 0.7, alpha = 1.9	3.80	11.85	0.06	1149	8
theta = 0.9, alpha = 1.9	0.06	0.02	0.06	1376	18

6.2.6 Results Tuning IUH-CMA-ES on the Noisy Sphere



NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
theta = 0.1, alpha = 1.1	0.08	0.05	0.08	1367	18
theta = 0.3, alpha = 1.1	0.07	0.02	0.07	1275	13
theta = 0.5, alpha = 1.1	0.10	0.07	0.08	1524	19
theta = 0.7, alpha = 1.1	0.15	0.11	0.12	1947	24
theta = 0.9, alpha = 1.1	0.23	0.13	0.20	2241	25
theta = 0.1, alpha = 1.3	**0.04**	**0.02**	**0.04**	**578**	**1**
theta = 0.3, alpha = 1.3	0.06	0.03	0.06	1099	11
theta = 0.5, alpha = 1.3	0.10	0.05	0.09	1532	20
theta = 0.7, alpha = 1.3	0.08	0.05	0.07	1348	17
theta = 0.9, alpha = 1.3	0.14	0.18	0.10	1610	22
theta = 0.1, alpha = 1.5	0.06	0.01	0.06	1006	8
theta = 0.3, alpha = 1.5	0.05	0.03	0.05	807	3
theta = 0.5, alpha = 1.5	0.05	0.03	0.04	734	2
theta = 0.7, alpha = 1.5	0.06	0.01	0.06	985	6
theta = 0.9, alpha = 1.5	0.12	0.07	0.12	1704	23
theta = 0.1, alpha = 1.7	0.06	0.02	0.06	1043	9
theta = 0.3, alpha = 1.7	0.06	0.03	0.05	935	5
theta = 0.5, alpha = 1.7	0.06	0.02	0.05	842	4
theta = 0.7, alpha = 1.7	0.07	0.03	0.06	1211	12
theta = 0.9, alpha = 1.7	0.12	0.14	0.07	1322	15
theta = 0.1, alpha = 1.9	0.07	0.03	0.07	1295	14
theta = 0.3, alpha = 1.9	0.10	0.15	0.06	1075	10
theta = 0.5, alpha = 1.9	0.07	0.02	0.07	1335	16
theta = 0.7, alpha = 1.9	0.06	0.04	0.06	994	7
theta = 0.9, alpha = 1.9	0.15	0.15	0.09	1566	21

6.3 Adaptive Versus Non-Adaptive Averaging

Given the tuned adaptive averaging techniques, the final question that we aim to answer is whether adaptive averaging is better than explicit resampling, implicit averaging, or even a canonical implementation. For the algorithmic schemes considered in this work we will attempt to answer this question based on empirical comparison on a number of artificial test problems in the context of Gaussian additive noise.

Table 6.2 shows the general setup adopted in the experiments. Table 6.3 shows the set of test problems used in the comparison. This set is partially based on the test problems used in [SHL$^+$05, HFRA09b, HFRA09a, HFRA10]. Full descriptions of these test problems are given in Appendix A. Table 6.4 shows the noise handling schemes that are considered for comparison. For implicit and explicit averaging we consider the optimal sample size or population size for each test problem, therewith aiming to compare the adaptive averaging techniques to optimally tuned non-adaptive schemes. For this, Section 6.3.1 and Section 6.3.2 present the results of an empirical study to find for each test problem the optimal sample size or population size. In Section 6.3.3 these results are used for the full empirical comparison.

General experimental settings	
Search space dimension size	$n = 10$
Evaluation budget per run	10,000
Runs per algorithmic scheme	100
Performance indicators	Final solution quality w.r.t. the underlying signal function (mean, std, median) over all runs, and rank sum for ranking of the algorithmic schemes

Table 6.2: The general experimental setup.

Test problem	**Properties of the underlying signal function**		
Noisy Sphere Problem	unimodal	separable	well-conditioned
Noisy Ellipsoid Problem	unimodal	non-separable	ill-conditioned
Noisy Step Ellipsoid Problem	unimodal	non-separable	ill-conditioned
Noisy Rosenbrock Problem	unimodal	non-separable	well-conditioned
Noisy Ackley Problem	multimodal	separable	well-conditioned
Noisy Griewank Problem	multimodal	separable	well-conditioned
Noisy Rastrigin Problem	multimodal	separable	well-conditioned
Noisy Schaffer's F7 Problem	multimodal	separable	well-conditioned
Noisy Branke's Multipeak Problem	multimodal	separable	well-conditioned
Noisy Keane's Bump Problem	multimodal	non-separable	well-conditioned

Table 6.3: The test problems used for empirical comparison.

Noise handling schemes used for empirical comparison	
Canonical	A canonical $(5/2_{DI}, 35)$-σSA-ES and CMA-ES.
MEM	Explicit resampling, optimally tuned for each test problem.
MPM	Implicit averaging, optimally tuned for each test problem.
PUH	The poset-based adaptive averaging method.
UH	The rank-based adaptive averaging method.
IUH	The inversions-based adaptive averaging method.

Table 6.4: The techniques considered in the empirical study on noise handling schemes.

6.3.1 The Optimal Sample Size for Explicit Averaging

This experiment is done in order to determine, for each test problem, the optimal sample size for explicit averaging. Different instances of the MEM-$(5/2_{DI}, 35)$-σSA-ES and the MEM-CMA-ES are considered with varying sample sizes: $m = 2, 4, \ldots, 16$. These sample sizes are compared on the test problems listed in Table 6.3 using the experimental setup shown in Table 6.2. The results of these experiments are shown in the tables and figures of Section 6.3.1.1 and Section 6.3.1.2 for the MEM-$(5/2_{DI}, 35)$-σSA-ES and the MEM-CMA-ES respectively.

The results show the trade-off between taking too few samples, leading to early stagnation, and too many samples, leading to slow convergence. In between lies an optimal sample size for the considered evaluation budget of $10,000$ function evaluations. For the explicit averaging schemes for each of the test problems with respect to the general experimental setup the optimal sample sizes lie, with an approximate error of ± 1, at the values shown in Table 6.5. From this table we see that the sample sizes for the MEM-$(5/2_{DI}, 35)$-σSA-ES are generally low and for the MEM-CMA-ES it seems that slightly higher sample sizes should be used.

Test problem	MEM-$(5, 35)$-σSA-ES	MEM-CMA-ES
Noisy Sphere Problem	m = 4	m = 12
Noisy Ellipsoid Problem	m = 4	m = 8
Noisy Step Ellipsoid Problem	m = 2	m = 2
Noisy Rosenbrock Problem	m = 2	m = 4
Noisy Ackley Problem	m = 4	m = 6
Noisy Griewank Problem	m = 6	m = 14
Noisy Rastrigin Problem	m = 2	m = 2
Noisy Schaffer's F7 Problem	m = 2	m = 4
Noisy Branke's Multipeak Problem	m = 4	m = 2
Noisy Keane's Bump Problem	m = 4	m = 2

Table 6.5: The optimal sample size for the MEM approach with an approximate error of ± 1 to achieve best convergence accuracy on a budget of $10,000$ function evaluations.

6.3.1.1 Results MEM-$(5/2_{DI}, 35)$-σSA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	2.65	9.46	0.37	27616	3
m = 4	**1.12**	**5.45**	**0.31**	**21548**	**1**
m = 6	1.30	6.54	0.33	22462	2
m = 8	2.77	8.84	0.41	31286	4
m = 10	2.85	8.96	0.54	40452	5
m = 12	2.52	5.56	0.88	50776	6
m = 14	7.33	12.70	1.86	60611	7
m = 16	5.61	8.47	2.77	65649	8

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	1.95	4.00	0.85	22374	2
m = 4	**1.86**	**4.32**	**0.81**	**20584**	**1**
m = 6	3.09	5.93	0.97	27054	3
m = 8	2.63	3.71	1.37	33075	4
m = 10	4.68	5.38	2.23	46163	5
m = 12	5.54	5.33	3.91	52517	6
m = 14	6.72	6.13	4.58	56617	7
m = 16	8.87	7.30	5.90	62016	8

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	**1.18**	**3.77**	**0.00**	**13133**	**1**
m = 4	1.97	4.52	1.00	23326	2
m = 6	1.19	2.56	1.00	24521	3
m = 8	3.28	5.48	1.00	37849	4
m = 10	3.67	4.35	2.00	45768	5
m = 12	5.39	5.32	4.00	54037	6
m = 14	6.12	5.05	4.00	57872	7
m = 16	8.39	6.02	7.00	63894	8

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	**8.63**	**1.10**	**8.74**	**15838**	**1**
m = 4	8.79	1.15	8.91	17670	2
m = 6	9.11	1.00	9.10	20403	3
m = 8	12.04	8.18	10.16	31062	4
m = 10	19.52	14.37	14.22	45578	5
m = 12	38.13	30.13	27.73	57724	6
m = 14	53.66	46.84	39.92	62745	7
m = 16	87.22	75.31	66.17	69380	8

NOISY ACKLEY PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	1.18	1.99	0.60	18224	2
m = 4	**1.13**	**1.93**	**0.66**	**18101**	**1**
m = 6	1.23	1.54	0.77	22770	3
m = 8	2.23	2.01	1.53	38075	4
m = 10	2.69	1.65	2.27	45781	5
m = 12	3.24	1.54	3.09	52847	6
m = 14	3.82	1.25	3.58	59927	7
m = 16	4.43	1.42	4.01	64675	8

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	1.29	0.40	1.20	44960	6
m = 4	1.27	0.44	1.15	37310	4
m = 6	**1.19**	**0.27**	**1.14**	**33226**	**1**
m = 8	1.18	0.19	1.13	33330	2
m = 10	1.22	0.32	1.15	36458	3
m = 12	1.23	0.29	1.17	39513	5
m = 14	1.25	0.23	1.20	45903	7
m = 16	1.36	0.38	1.21	49700	8

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	**27.79**	**37.93**	**12.06**	**16549**	**1**
m = 4	49.94	42.88	31.42	26768	2
m = 6	66.15	33.99	70.44	32571	3
m = 8	82.83	22.06	78.51	41122	4
m = 10	90.65	23.09	87.51	47240	5
m = 12	94.16	19.57	93.40	50716	7
m = 14	93.36	19.08	89.77	50053	6
m = 16	100.06	20.33	100.72	55381	8

NOISY SCHAFFERS F7 PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	**6.11**	**13.90**	**2.40**	**9634**	**1**
m = 4	11.05	16.89	4.72	20150	2
m = 6	18.86	17.13	12.19	31284	3
m = 8	22.65	13.39	18.31	37013	4
m = 10	32.46	13.82	26.99	49204	5
m = 12	37.37	14.42	34.08	54709	6
m = 14	39.45	12.61	35.54	57584	7
m = 16	42.26	11.18	41.26	60822	8

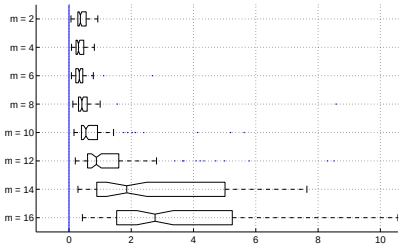
NOISY BRANKE MULTIPEAK PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	0.37	0.10	0.34	30149	3
m = 4	**0.35**	**0.06**	**0.33**	**23717**	**1**
m = 6	0.35	0.07	0.33	27505	2
m = 8	0.38	0.09	0.34	34868	4
m = 10	0.41	0.09	0.37	44727	5
m = 12	0.42	0.09	0.38	47615	6
m = 14	0.44	0.09	0.42	53656	7
m = 16	0.46	0.09	0.44	58163	8

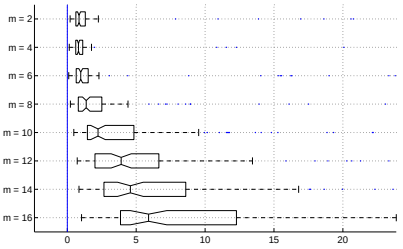
NOISY KEANE BUMP PROBLEM

	Mean	Std	Med	$\sum$#	#
m = 2	-0.59	0.14	-0.64	17807	2
m = 4	**-0.60**	**0.07**	**-0.61**	**17351**	**1**
m = 6	-0.55	0.10	-0.56	26485	3
m = 8	-0.48	0.11	-0.52	37184	4
m = 10	-0.43	0.11	-0.45	46214	5
m = 12	-0.39	0.09	-0.39	53539	6
m = 14	-0.35	0.08	-0.35	59281	7
m = 16	-0.33	0.09	-0.33	62539	8

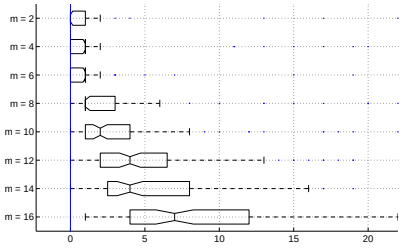
NOISY SPHERE PROBLEM



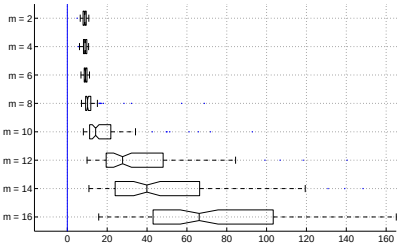
NOISY ELLIPSOID PROBLEM



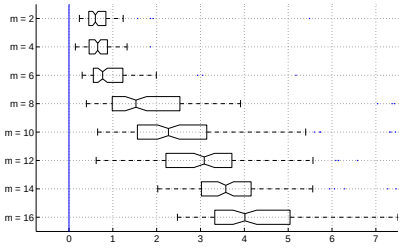
NOISY STEP ELLIPSOID PROBLEM



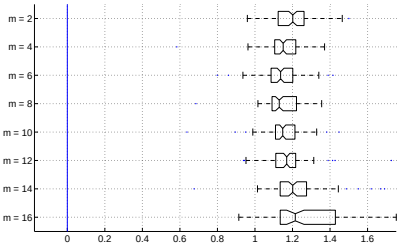
NOISY ROSENBROCK PROBLEM



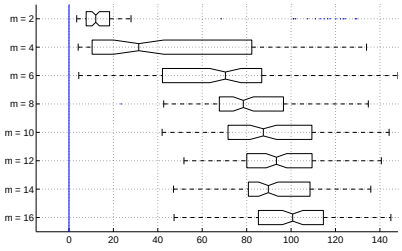
NOISY ACKLEY PROBLEM



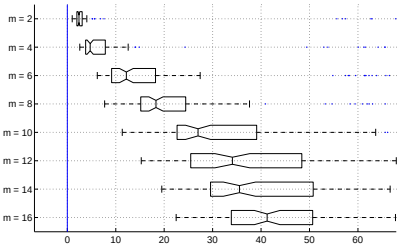
NOISY GRIEWANK PROBLEM



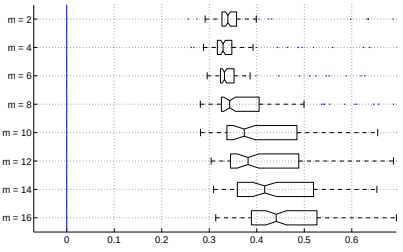
NOISY RASTRIGIN PROBLEM



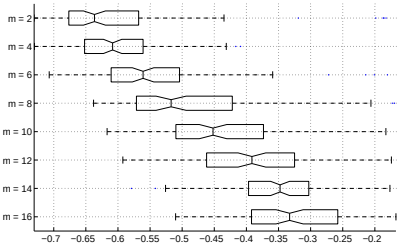
NOISY SCHAFFER'S F7 PROBLEM



NOISY BRANKE'S MULTIPEAK PROBLEM



NOISY KEANE'S BUMP PROBLEM



6.3.1.2 Results MEM-CMA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
m = 2	0.44	0.18	0.42	61210	8
m = 4	0.32	0.14	0.30	49041	7
m = 6	0.85	6.02	0.23	37108	6
m = 8	2.46	9.91	0.20	36067	5
m = 10	0.71	4.74	0.22	34633	3
m = 12	**1.36**	**7.88**	**0.21**	**32802**	**1**
m = 14	1.06	5.81	0.21	33597	2
m = 16	1.95	9.80	0.21	35942	4

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
m = 2	1.31	1.70	1.07	50014	8
m = 4	1.14	2.53	0.86	38460	5
m = 6	1.55	5.31	0.70	33460	2
m = 8	**1.80**	**4.81**	**0.67**	**31769**	**1**
m = 10	1.87	4.37	0.75	36178	3
m = 12	2.35	7.13	0.78	37697	4
m = 14	3.51	9.52	0.88	44445	6
m = 16	3.86	7.93	1.11	48377	7

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
m = 2	**1.27**	**3.44**	**0.00**	**22837**	**1**
m = 4	1.05	1.18	1.00	29963	3
m = 6	1.50	3.97	1.00	29819	2
m = 8	2.19	8.57	1.00	35177	4
m = 10	2.73	6.74	1.00	45646	5
m = 12	2.11	4.29	1.00	46145	6
m = 14	3.36	6.47	1.00	53768	7
m = 16	5.04	8.48	2.00	57045	8

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	∑#	#
m = 2	38.99	261.44	9.63	31838	2
m = 4	**71.43**	**408.26**	**9.25**	**28336**	**1**
m = 6	86.79	404.63	9.52	32392	3
m = 8	217.35	639.96	9.78	38770	5
m = 10	210.64	636.02	9.54	36780	4
m = 12	283.76	832.69	9.98	43746	6
m = 14	310.96	766.21	12.08	50537	7
m = 16	431.76	1015.12	20.58	58001	8

NOISY ACKLEY PROBLEM

	Mean	Std	Med	∑#	#
m = 2	1.10	0.68	0.86	56409	8
m = 4	0.90	1.23	0.53	43022	6
m = 6	**0.69**	**1.14**	**0.39**	**29784**	**1**
m = 8	0.90	1.80	0.42	30780	2
m = 10	0.78	1.49	0.41	31160	3
m = 12	0.63	0.80	0.47	34489	4
m = 14	0.72	0.61	0.53	40276	5
m = 16	1.24	1.31	0.81	54480	7

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	∑#	#
m = 2	1.08	0.32	1.17	46503	7
m = 4	1.11	0.18	1.14	47223	8
m = 6	1.09	0.20	1.13	45640	6
m = 8	1.14	0.32	1.10	40570	5
m = 10	1.08	0.13	1.10	35794	4
m = 12	1.13	0.23	1.09	35078	3
m = 14	**1.09**	**0.08**	**1.09**	**34535**	**1**
m = 16	1.15	0.30	1.08	35057	2

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	∑#	#
m = 2	**16.09**	**14.75**	**13.54**	**17294**	**1**
m = 4	21.95	25.06	14.75	21551	2
m = 6	26.91	25.97	16.50	24969	3
m = 8	46.89	36.19	28.91	37678	4
m = 10	64.43	35.16	67.65	48504	5
m = 12	71.29	28.52	77.92	53043	6
m = 14	81.88	22.42	82.71	59232	8
m = 16	81.21	21.02	79.88	58129	7

NOISY SCHAFFERS F7 PROBLEM

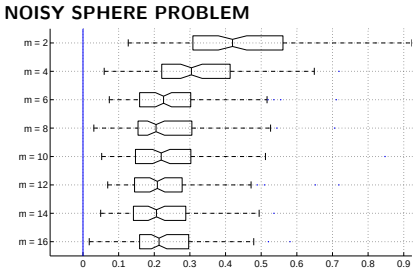
	Mean	Std	Med	∑#	#
m = 2	5.06	6.53	3.72	20228	3
m = 4	**5.37**	**8.95**	**3.30**	**19009**	**1**
m = 6	4.62	3.40	3.30	20120	2
m = 8	9.31	12.27	5.99	33921	4
m = 10	12.20	11.78	8.91	44711	5
m = 12	16.42	13.37	12.14	54832	6
m = 14	19.27	13.63	15.17	60625	7
m = 16	21.14	8.90	19.39	66954	8

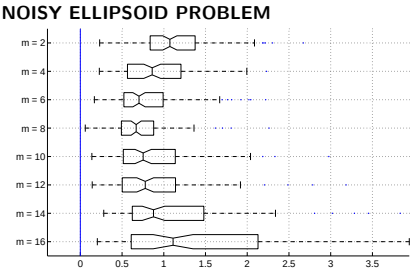
NOISY BRANKE MULTIPEAK PROBLEM

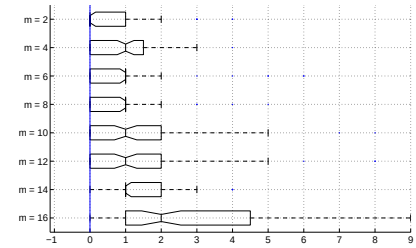
	Mean	Std	Med	∑#	#
m = 2	**0.36**	**0.11**	**0.33**	**31570**	**1**
m = 4	0.40	0.16	0.33	33160	2
m = 6	0.44	0.18	0.33	36489	3
m = 8	0.44	0.16	0.34	37489	4
m = 10	0.45	0.17	0.35	39659	5
m = 12	0.48	0.16	0.47	44608	6
m = 14	0.50	0.15	0.54	48942	8
m = 16	0.50	0.15	0.53	48483	7

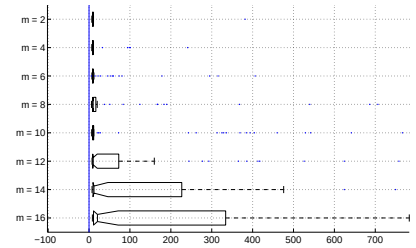
NOISY KEANE BUMP PROBLEM

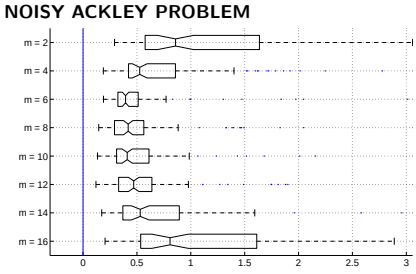
	Mean	Std	Med	∑#	#
m = 2	**-0.42**	**0.17**	**-0.47**	**27358**	**1**
m = 4	-0.41	0.16	-0.44	27921	2
m = 6	-0.37	0.16	-0.35	34292	3
m = 8	-0.34	0.15	-0.33	37736	4
m = 10	-0.29	0.13	-0.27	45434	5
m = 12	-0.27	0.11	-0.26	48101	7
m = 14	-0.28	0.10	-0.28	45571	6
m = 16	-0.24	0.09	-0.23	53987	8

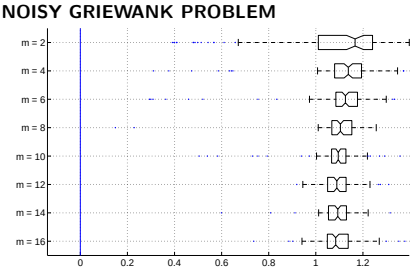

NOISY SPHERE PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

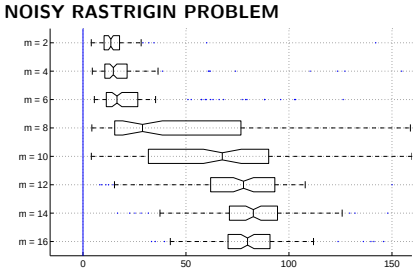

NOISY ELLIPSOID PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 0.5 1 1.5 2 2.5 3 3.5

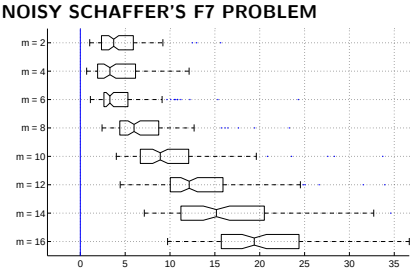

NOISY STEP ELLIPSOID PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
−1 0 1 2 3 4 5 6 7 8 9

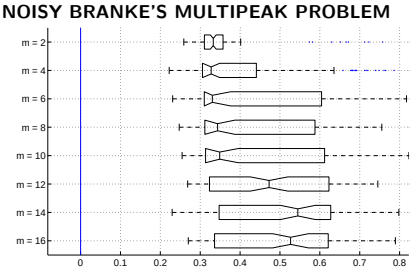

NOISY ROSENBROCK PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
−100 0 100 200 300 400 500 600 700

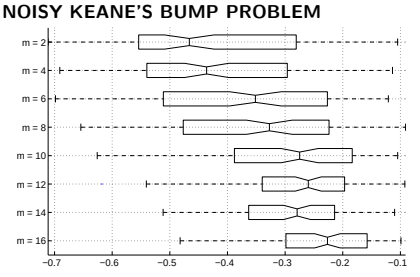

NOISY ACKLEY PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 0.5 1 1.5 2 2.5 3


NOISY GRIEWANK PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 0.2 0.4 0.6 0.8 1 1.2


NOISY RASTRIGIN PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 50 100 150


NOISY SCHAFFER'S F7 PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 5 10 15 20 25 30 35


NOISY BRANKE'S MULTIPEAK PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8


NOISY KEANE'S BUMP PROBLEM
m = 2
m = 4
m = 6
m = 8
m = 10
m = 12
m = 14
m = 16
−0.7 −0.6 −0.5 −0.4 −0.3 −0.2 −0.1

6.3.2 The Optimal Sample Size for Implicit Averaging

This experiment is done in order to determine, for each test problem, the optimal population size for implicit averaging. Different instances of the $(\mu/2_{DI}, \lambda)$-σSA-ES and the CMA-ES with varying population sizes are considered. For the $(\mu/2_{DI}, \lambda)$-σSA-ES: $(5, 35)$, $(10, 70)$, $(20, 140)$, $(30, 210)$, $(40, 280)$, $(50, 350)$, $(60, 420)$. For the CMA-ES: $(5, 10)$, $(10, 20)$, $(20, 40)$, $(30, 60)$, $(40, 80)$, $(50, 100)$, $(60, 120)$, $(70, 140)$. These different population sizes are compared on the test problems listed in Table 6.3 using the experimental setup as displayed in Table 6.2. The results of these experiments are shown in the tables and figures of Section 6.3.2.1 and Section 6.3.2.2 for the MPM-$(\mu/2_{DI}, \lambda)$-σSA-ES and the MPM-CMA-ES respectively.

Similar to the results of explicit averaging, the results show the trade-off that exists between taking too small population sizes that leads to early stagnation and too high population sizes that leads to slow convergence. In between lies an optimal population size for the considered evaluation budget of $10,000$ function evaluations. Based on these results, we can conclude that for the implicit averaging schemes, the MPM-$(\mu/2_{DI}, \lambda)$-σSA-ES and the MPM-CMA-ES, given the experimental setup, the optimal sample sizes lie at the values shown in Table 6.6. When taking α as the scaling factor of the default population sizes for both schemes, the error of these results is ± 10 for μ, maintaining the default ratios between μ and λ. From Table 6.6 we observe that for the MPM-$(\mu/2_{DI}, \lambda)$-σSA-ES, the implicit averaging factor is generally low and for the CMA-ES, slightly higher factors are optimal. This observation is similar to what is observed in the experiments of Section 6.3.1 in the comparison of different sample size for explicit averaging.

Test problem	**MPM-$(\mu/2_{DI}, \lambda)$-σSA-ES**	**MPM-CMA-ES**
Noisy Sphere Problem	(10,70)	(50,100)
Noisy Ellipsoid Problem	(20,140)	(40,80)
Noisy Step Ellipsoid Problem	(20,140)	(30,60)
Noisy Rosenbrock Problem	(20,140)	(40,80)
Noisy Ackley Problem	(10,70)	(40,80)
Noisy Griewank Problem	(20,140)	(40,80)
Noisy Rastrigin Problem	(20,140)	(30,60)
Noisy Schaffer's F7 Problem	(10,70)	(20,40)
Noisy Branke's Multipeak Problem	(20,140)	(30,60)
Noisy Keane's Bump Problem	(10,70)	(30,60)

Table 6.6: For each multi-population scheme for each test problem the optimal population size to achieve best convergence accuracy on a budget of $10,000$ function evaluations. When taking α as the scaling factor of the default population sizes for both schemes, the error of these results is ± 10 for μ, maintaining the default ratios between μ and λ.

6.3.2.1 Results MPM-$(\mu/2_{DI},\lambda)$-σSA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	2.80	8.95	0.59	40598	5
(10,70)	**0.41**	**0.19**	**0.37**	**23013**	**1**
(20,140)	0.42	0.23	0.35	23713	2
(30,210)	0.52	0.23	0.51	33364	3
(40,280)	0.59	0.30	0.52	36746	4
(50,350)	0.68	0.31	0.65	44069	6
(60,420)	0.87	0.34	0.84	55173	7
(70,490)	1.14	0.47	1.09	63724	8

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	1.88	3.64	1.02	31179	4
(10,70)	1.40	3.05	0.94	25903	2
(20,140)	**1.07**	**1.12**	**0.82**	**25236**	**1**
(30,210)	1.09	0.57	0.97	28911	3
(40,280)	1.49	0.69	1.41	41165	5
(50,350)	1.74	0.74	1.73	47970	6
(60,420)	2.17	0.96	2.11	55671	7
(70,490)	2.85	1.23	2.56	64365	8

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	1.91	5.25	1.00	23880	3
(10,70)	1.16	3.44	0.00	23053	2
(20,140)	**0.38**	**0.58**	**0.00**	**21802**	**1**
(30,210)	0.58	0.73	0.00	30540	4
(40,280)	1.00	0.92	1.00	41283	5
(50,350)	1.46	1.04	1.00	51105	6
(60,420)	2.12	1.20	2.00	61149	7
(70,490)	2.68	1.38	3.00	67588	8

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	9.36	6.42	8.70	26691	3
(10,70)	8.55	0.99	8.57	22443	2
(20,140)	**8.54**	**0.75**	**8.53**	**21091**	**1**
(30,210)	8.91	0.80	8.84	28526	4
(40,280)	9.38	0.83	9.20	36718	5
(50,350)	10.17	1.00	10.19	48650	6
(60,420)	12.00	1.46	12.17	64043	7
(70,490)	14.10	2.19	13.98	72238	8

NOISY ACKLEY PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	1.98	2.76	0.85	24915	3
(10,70)	**1.27**	**2.06**	**0.65**	**15895**	**1**
(20,140)	1.03	0.80	0.87	18688	2
(30,210)	1.72	0.51	1.71	32300	4
(40,280)	2.66	0.72	2.67	45007	5
(50,350)	3.28	0.61	3.27	54289	6
(60,420)	3.85	0.58	3.85	63504	7
(70,490)	4.04	0.66	4.01	65802	8

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	1.40	0.46	1.28	46229	7
(10,70)	1.21	0.11	1.20	32629	3
(20,140)	**1.19**	**0.11**	**1.18**	**28874**	**1**
(30,210)	1.21	0.10	1.19	31370	2
(40,280)	1.26	0.11	1.25	42578	5
(50,350)	1.26	0.13	1.25	41984	4
(60,420)	1.28	0.15	1.27	45009	6
(70,490)	1.33	0.16	1.32	51727	8

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	$\sum$#	#
(5,35)	25.66	36.95	11.27	33009	4
(10,70)	15.12	29.17	5.75	20476	2
(20,140)	**7.16**	**16.56**	**3.94**	**10749**	**1**
(30,210)	20.52	19.30	12.32	30308	3
(40,280)	44.92	12.00	46.97	48249	5
(50,350)	51.65	9.00	51.81	54729	6
(60,420)	56.01	10.28	56.51	59848	7
(70,490)	58.29	7.27	57.89	63032	8

NOISY SCHAFFERS F7 PROBLEM

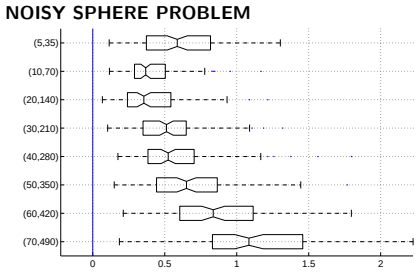
	Mean	Std	Med	$\sum$#	#
(5,35)	5.12	11.47	2.28	15876	2
(10,70)	**2.68**	**5.17**	**2.08**	**10141**	**1**
(20,140)	3.44	0.65	3.45	22667	3
(30,210)	6.22	1.20	6.23	34166	4
(40,280)	10.26	1.69	10.26	44882	5
(50,350)	14.33	2.23	13.99	55410	6
(60,420)	17.77	2.44	17.73	64592	7
(70,490)	21.17	2.68	21.10	72666	8

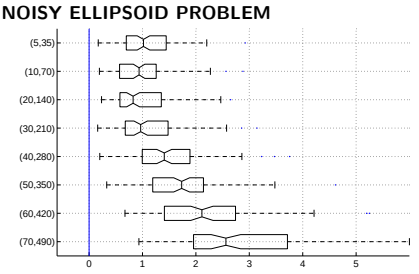
NOISY BRANKE MULTIPEAK PROBLEM

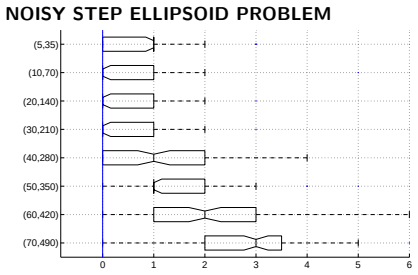
	Mean	Std	Med	$\sum$#	#
(5,35)	0.38	0.09	0.35	31124	4
(10,70)	0.36	0.06	0.34	25428	2
(20,140)	**0.34**	**0.02**	**0.34**	**20688**	**1**
(30,210)	0.36	0.03	0.35	29870	3
(40,280)	0.38	0.05	0.37	40128	5
(50,350)	0.41	0.06	0.40	52021	6
(60,420)	0.43	0.06	0.42	59024	7
(70,490)	0.45	0.06	0.45	62117	8

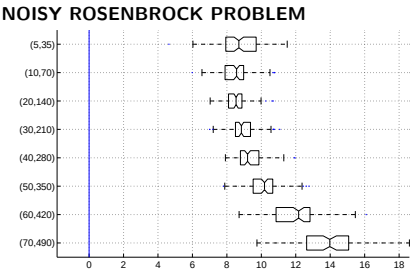
NOISY KEANE BUMP PROBLEM

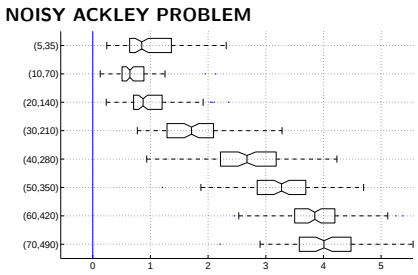
	Mean	Std	Med	$\sum$#	#
(5,35)	-0.53	0.19	-0.59	37524	4
(10,70)	**-0.67**	**0.07**	**-0.68**	**14122**	**1**
(20,140)	-0.66	0.07	-0.67	16270	2
(30,210)	-0.61	0.09	-0.62	28437	3
(40,280)	-0.54	0.09	-0.56	42853	5
(50,350)	-0.48	0.07	-0.49	53171	6
(60,420)	-0.42	0.07	-0.42	61586	7
(70,490)	-0.39	0.06	-0.39	66437	8

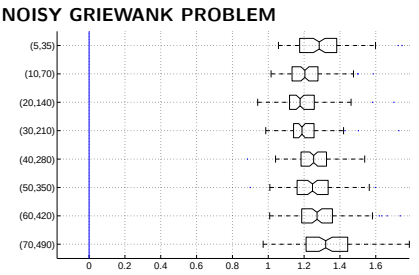

NOISY SPHERE PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
0.5
1
1.5
2

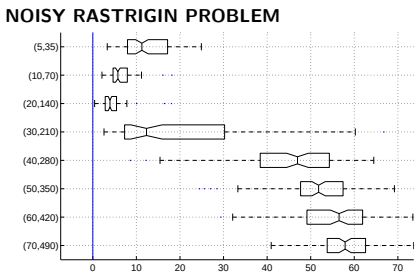

NOISY ELLIPSOID PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
1
2
3
4
5

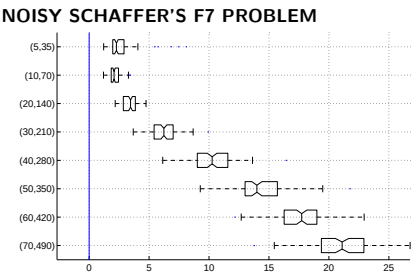

NOISY STEP ELLIPSOID PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
1
2
3
4
5
6

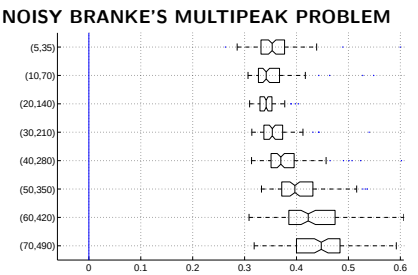

NOISY ROSENBROCK PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
2
4
6
8
10
12
14
16
18

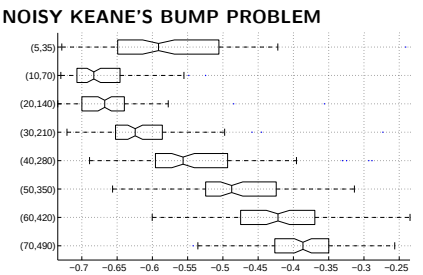

NOISY ACKLEY PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
1
2
3
4
5


NOISY GRIEWANK PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


NOISY RASTRIGIN PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
10
20
30
40
50
60
70


NOISY SCHAFFER'S F7 PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
5
10
15
20
25


NOISY BRANKE'S MULTIPEAK PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
0
0.1
0.2
0.3
0.4
0.5
0.6


NOISY KEANE'S BUMP PROBLEM
(5,35)
(10,70)
(20,140)
(30,210)
(40,280)
(50,350)
(60,420)
(70,490)
−0.7
−0.65
−0.6
−0.55
−0.5
−0.45
−0.4
−0.35
−0.3
−0.25

6.3.2.2 Results MPM-CMA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	0.64	0.30	0.57	72163	8
(10,20)	0.32	0.13	0.30	58550	7
(20,40)	0.41	2.24	0.16	39097	6
(30,60)	0.15	0.08	0.13	30397	3
(40,80)	1.30	5.95	0.13	31001	4
(50,100)	**0.13**	**0.07**	**0.12**	**27863**	**1**
(60,120)	0.14	0.09	0.13	29096	2
(70,140)	0.16	0.11	0.13	32233	5

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	1.88	4.43	1.29	67558	8
(10,20)	0.98	0.47	0.83	61041	7
(20,40)	1.24	3.51	0.45	42527	6
(30,60)	0.50	1.43	0.27	27724	2
(40,80)	**0.35**	**0.24**	**0.26**	**26619**	**1**
(50,100)	0.40	0.29	0.34	31036	4
(60,120)	0.42	0.44	0.31	29705	3
(70,140)	0.49	0.58	0.38	34190	5

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	1.25	1.34	1.00	42626	6
(10,20)	0.78	1.15	0.00	33268	2
(20,40)	0.93	2.27	0.00	35117	3
(30,60)	**0.66**	**2.56**	**0.00**	**30781**	**1**
(40,80)	0.39	0.94	0.00	35860	4
(50,100)	0.35	0.95	0.00	40772	5
(60,120)	0.19	0.60	0.00	46332	7
(70,140)	0.33	0.77	0.00	55644	8

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	72.01	616.49	9.76	54698	8
(10,20)	9.26	0.77	9.31	45337	6
(20,40)	36.26	201.84	9.07	40242	4
(30,60)	14.37	56.67	8.65	29441	2
(40,80)	**8.54**	**0.63**	**8.50**	**24091**	**1**
(50,100)	8.91	0.95	8.80	32494	3
(60,120)	9.16	0.78	9.07	40896	5
(70,140)	10.03	1.79	9.57	53201	7

NOISY ACKLEY PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	1.98	1.17	1.89	71112	8
(10,20)	0.74	0.89	0.53	50352	6
(20,40)	0.60	1.56	0.29	27570	3
(30,60)	0.56	1.50	0.25	23687	2
(40,80)	**0.27**	**0.10**	**0.25**	**20831**	**1**
(50,100)	0.40	0.52	0.31	29462	4
(60,120)	0.72	1.13	0.47	45430	5
(70,140)	0.73	0.58	0.55	51956	7

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	1.27	0.18	1.27	64831	8
(10,20)	1.10	0.20	1.14	54192	7
(20,40)	1.02	0.20	1.07	34586	5
(30,60)	1.05	0.17	1.06	33716	4
(40,80)	**1.05**	**0.08**	**1.05**	**31175**	**1**
(50,100)	1.08	0.17	1.05	32149	2
(60,120)	1.06	0.06	1.06	32849	3
(70,140)	1.10	0.20	1.06	36902	6

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	∑#	#
(5,10)	14.92	5.02	14.79	56521	7
(10,20)	7.56	2.83	6.99	41094	5
(20,40)	7.32	16.49	4.31	27918	3
(30,60)	**6.12**	**14.81**	**3.29**	**20923**	**1**
(40,80)	9.16	18.17	3.66	25880	2
(50,100)	13.06	16.23	4.54	35720	4
(60,120)	19.77	17.00	13.59	49587	6
(70,140)	30.88	20.04	30.56	62757	8

NOISY SCHAFFERS F7 PROBLEM

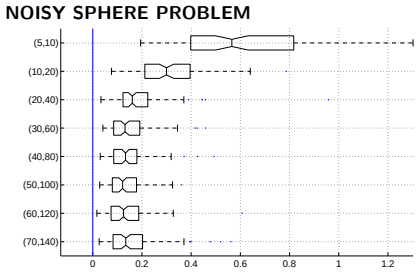
	Mean	Std	Med	∑#	#
(5,10)	5.91	6.55	4.49	64893	8
(10,20)	2.17	1.35	1.82	31858	4
(20,40)	**2.83**	**8.60**	**1.27**	**16479**	**1**
(30,60)	2.02	5.77	1.38	17063	2
(40,80)	2.58	5.96	1.77	28836	3
(50,100)	2.42	0.60	2.33	42748	5
(60,120)	3.09	0.70	3.05	54625	6
(70,140)	4.12	1.51	3.84	63898	7

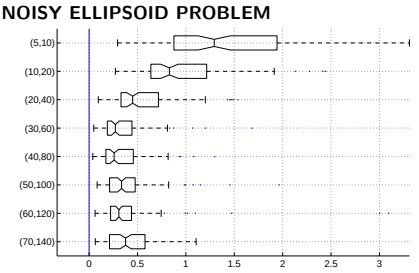
NOISY BRANKE MULTIPEAK PROBLEM

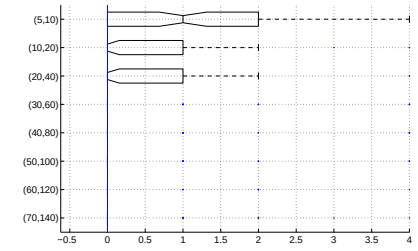
	Mean	Std	Med	∑#	#
(5,10)	0.37	0.11	0.34	49302	7
(10,20)	0.34	0.06	0.33	40695	5
(20,40)	0.33	0.07	0.32	30137	3
(30,60)	**0.33**	**0.06**	**0.31**	**25950**	**1**
(40,80)	0.32	0.04	0.32	30013	2
(50,100)	0.33	0.03	0.32	38527	4
(60,120)	0.34	0.04	0.34	48637	6
(70,140)	0.36	0.05	0.35	57139	8

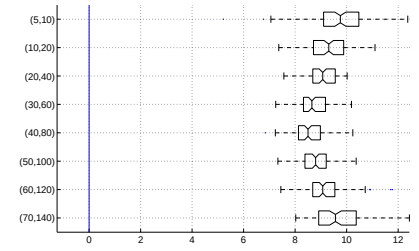
NOISY KEANE BUMP PROBLEM

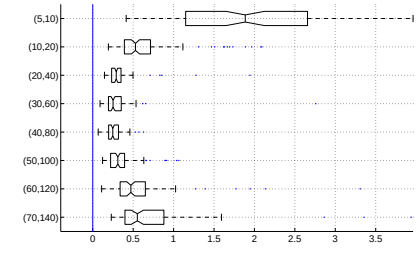
	Mean	Std	Med	∑#	#
(5,10)	-0.48	0.13	-0.50	57303	8
(10,20)	-0.54	0.15	-0.58	43049	5
(20,40)	-0.60	0.10	-0.62	30184	3
(30,60)	**-0.62**	**0.08**	**-0.64**	**25179**	**1**
(40,80)	-0.62	0.05	-0.62	28369	2
(50,100)	-0.58	0.08	-0.59	38621	4
(60,120)	-0.56	0.07	-0.57	45886	6
(70,140)	-0.54	0.08	-0.54	51809	7

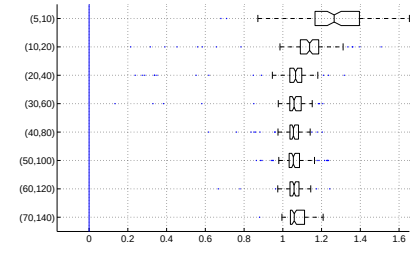

NOISY SPHERE PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 0.2 0.4 0.6 0.8 1 1.2

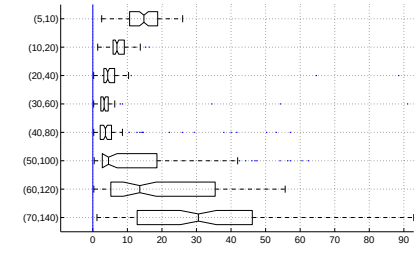

NOISY ELLIPSOID PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 0.5 1 1.5 2 2.5 3

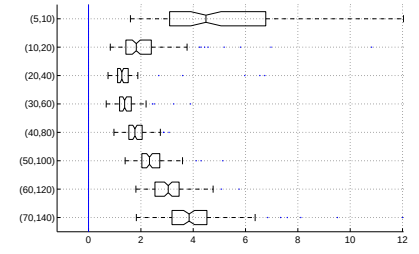

NOISY STEP ELLIPSOID PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
−0.5 0 0.5 1 1.5 2 2.5 3 3.5 4

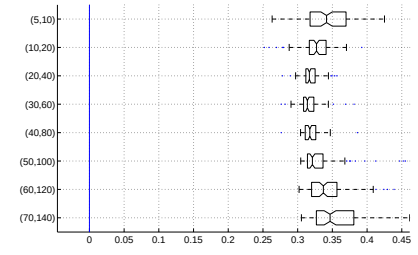

NOISY ROSENBROCK PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 2 4 6 8 10 12

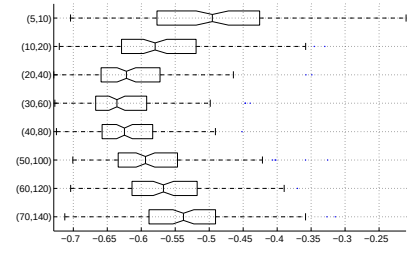

NOISY ACKLEY PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 0.5 1 1.5 2 2.5 3 3.5


NOISY GRIEWANK PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6


NOISY RASTRIGIN PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 10 20 30 40 50 60 70 80 90


NOISY SCHAFFER'S F7 PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 2 4 6 8 10 12


NOISY BRANKE'S MULTIPEAK PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45


NOISY KEANE'S BUMP PROBLEM
(5,10)
(10,20)
(20,40)
(30,60)
(40,80)
(50,100)
(60,120)
(70,140)
−0.7 −0.65 −0.6 −0.55 −0.5 −0.45 −0.4 −0.35 −0.3 −0.25

6.3.3 Comparison Adaptive versus Non-Adaptive

Lastly we run an empirical comparative study of the best instances of the five noise handling schemes incorporated in the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, compared against the canonical instance of both schemes on the complete set of test problems (see Table 6.3). Each instance uses the optimal settings as found in the previous sections (see Table 6.7). For the MEM and the MPM schemes, this varies for each test problem. The question that we aim to answer with this empirical comparison is: how do the adaptive averaging techniques compare to optimally tuned MEM and MPM schemes?

	$(5/2_{DI}, 35)$-σ**SA-ES**	**CMA-ES**
Canonical	default	default
MEM	see Table 6.5	see Table 6.5
MPM	see Table 6.6	see Table 6.6
PUH	$\delta = 0.9$, $\alpha = 1.3$	$\delta = 0.5$, $\alpha = 1.7$
UH	$\theta = 0.9$, $\alpha = 1.1$	$\theta = 0.9$, $\alpha = 1.5$
IUH	$\theta = 0.1$, $\alpha = 1.7$	$\theta = 0.3$, $\alpha = 1.5$

Table 6.7: Algorithm settings comparison noise handling techniques.

Section 6.3.3.1 and Section 6.3.3.2 show the results for the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES respectively. Table 6.8 shows the combined rank scores of the compared schemes.

	$(5/2_{DI}, 35)$-σ**SA-ES**	**CMA-ES**
Canonical	354,792	384,091
MEM	296,002	291,950
MPM	289,257	**135,653**
PUH	335,450	344,457
UH	265,871	329,444
IUH	**261,628**	317,405

Table 6.8: Combined rank sums of the full comparison of noise handling techniques on the full set of test problems.

For the $(5/2_{DI}, 35)$-σSA-ES, the statistics tables and the boxplots show that overall, all schemes yield comparable results except for a few outliers. On the sphere problem, the adaptive averaging schemes clearly outperform the static schemes, which also holds for the Griewank problem and Branke's multipeak problem. On the Rastriging problem, the implicit averaging scheme is clearly better. A remarkable negative result is observed for the PUH scheme on the Ackley problem, Schaffer's f7 problem, and the Keane bump problem, where it is clearly worse than all other schemes. This can be attributed to a very slow convergence rate that is due to a

fast increase of the sample size. The canonical instantiation of the $(5/2_{DI}, 35)$-σSA-ES is in most, but not all cases outperformed by the other schemes.

For the CMA-ES, the implicit averaging scheme shows to be a clear winner, ranking first on all test problems. Although implicit averaging already showed to be more suitable for (weighted) intermediate recombination in Section 5.3.3, the gain is remarkable. Hence, for the CMA-ES, using larger population sizes for noisy objective functions is clearly beneficial. No clear winner can be identified when comparing the adaptive averaging schemes against the explicit resampling scheme. However, Table 6.8 shows a slightly better combined rank sum for the MEM approach. Of the adaptive averaging techniques, the PUH scheme is the worst choice and the IUH seems to be marginally better than the UH scheme. Also here, the canonical instantiation is generally outperformed by the other schemes.

To summarize, a well-tuned static noise handling scheme can yield competitive or better results than adaptive averaging scheme. Increasing the population size is, especially for the CMA-ES, a promising strategy for noise handling. Among the adaptive averaging schemes, the UH and the IUH scheme yield better results than the PUH scheme, and the difference between the UH and the IUH scheme is marginal. Finally, the results confirm that the noise handling schemes generally yield better results than the canonical instantiations of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES.

6.3.3.1 Results $(5/2_{DI}, 35)$-σSA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Canonical	2.39	9.45	0.48	45077	6
MEMopt	2.09	7.86	0.28	33477	4
MPMopt	1.03	4.08	0.39	41957	5
PUH	**1.97**	**8.02**	**0.17**	**18400**	**1**
UH	1.27	6.04	0.20	21950	3
IUH	1.23	6.01	0.18	19439	2

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.60	2.62	1.07	37491	6
MEMopt	2.39	4.94	0.83	31699	4
MPMopt	1.14	1.64	0.89	31829	5
PUH	2.19	5.27	0.86	31632	3
UH	1.51	4.00	0.72	25089	2
IUH	**1.62**	**4.26**	**0.64**	**22560**	**1**

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
Canonical	**1.46**	**4.49**	**0.50**	**20917**	**1**
MEMopt	1.32	3.23	1.00	26712	3
MPMopt	0.48	0.63	0.00	24501	2
PUH	1.60	3.69	1.00	36378	5
UH	1.58	4.64	0.00	33421	4
IUH	1.85	5.28	0.00	38371	6

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	∑#	#
Canonical	8.93	1.16	8.90	34190	6
MEMopt	8.72	1.13	8.71	30623	5
MPMopt	8.58	0.73	8.51	27934	2
PUH	9.48	6.97	8.65	29283	3
UH	8.64	1.01	8.66	30395	4
IUH	**9.16**	**6.68**	**8.52**	**27875**	**1**

NOISY ACKLEY PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.52	2.34	0.77	33316	5
MEMopt	0.95	1.50	0.58	25200	3
MPMopt	1.14	1.80	0.70	29322	4
PUH	2.17	1.68	1.79	48382	6
UH	1.59	2.71	0.49	22154	2
IUH	**1.06**	**1.93**	**0.53**	**21926**	**1**

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.32	0.35	1.23	44692	6
MEMopt	1.16	0.21	1.12	29809	4
MPMopt	1.22	0.12	1.21	42350	5
PUH	**1.10**	**0.24**	**1.08**	**19199**	**1**
UH	1.14	0.35	1.09	21755	2
IUH	1.18	0.40	1.08	22495	3

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	∑#	#
Canonical	23.63	35.54	10.56	32775	2
MEMopt	26.91	39.33	10.52	33178	4
MPMopt	**9.54**	**21.66**	**4.26**	**11411**	**1**
PUH	25.59	35.93	11.53	35138	6
UH	24.80	36.01	10.59	33095	3
IUH	33.85	44.41	11.50	34703	5

NOISY SCHAFFERS F7 PROBLEM

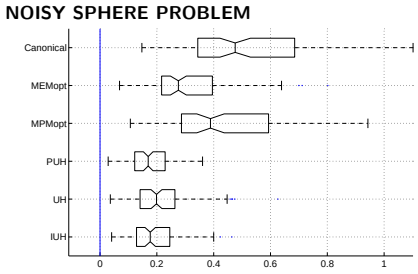
	Mean	Std	Med	∑#	#
Canonical	7.88	17.14	2.34	30599	5
MEMopt	7.49	16.26	2.24	28243	3
MPMopt	**2.02**	**0.42**	**1.97**	**19458**	**1**
PUH	10.41	17.84	3.61	45587	6
UH	10.05	20.42	2.14	26159	2
IUH	10.15	20.34	2.30	30254	4

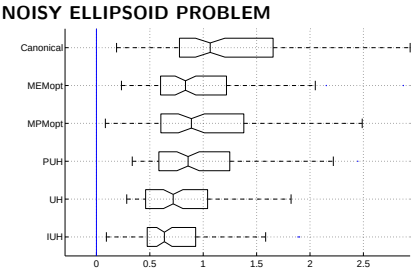
NOISY BRANKE MULTIPEAK PROBLEM

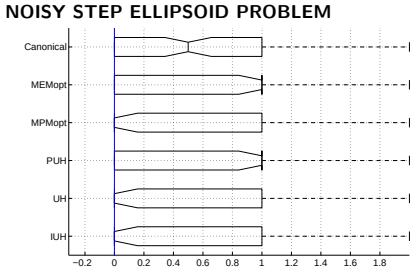
	Mean	Std	Med	∑#	#
Canonical	0.39	0.12	0.35	42964	6
MEMopt	0.34	0.06	0.33	30442	4
MPMopt	0.35	0.02	0.34	40114	5
PUH	0.35	0.07	0.33	28137	3
UH	0.34	0.08	0.32	22750	2
IUH	**0.32**	**0.06**	**0.31**	**15893**	**1**

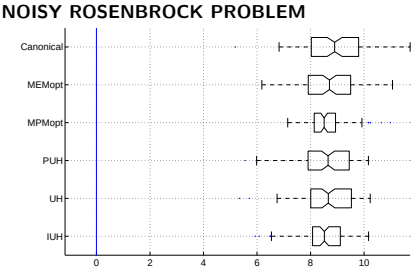
NOISY KEANE BUMP PROBLEM

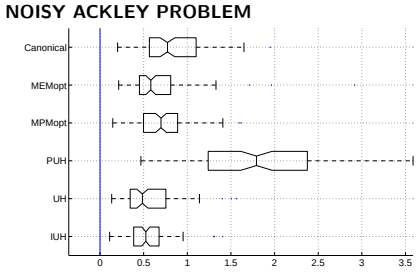
	Mean	Std	Med	∑#	#
Canonical	-0.53	0.20	-0.60	32771	5
MEMopt	-0.61	0.09	-0.64	26619	2
MPMopt	**-0.63**	**0.12**	**-0.66**	**20381**	**1**
PUH	-0.49	0.13	-0.50	43314	6
UH	-0.57	0.15	-0.62	29103	4
IUH	-0.59	0.12	-0.62	28112	3

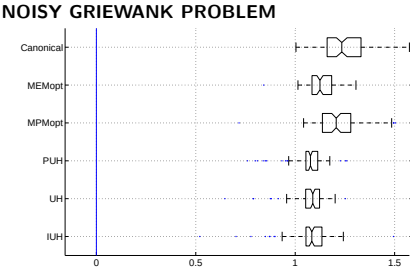

NOISY SPHERE PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
0.2
0.4
0.6
0.8
1

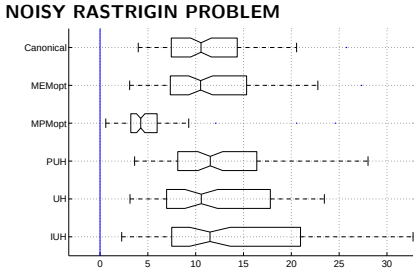

NOISY ELLIPSOID PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
0.5
1
1.5
2
2.5

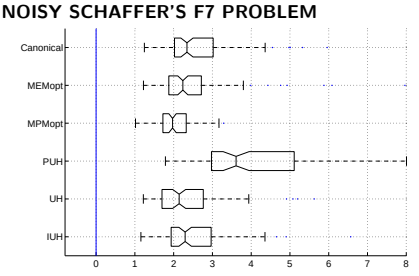

NOISY STEP ELLIPSOID PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
−0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8

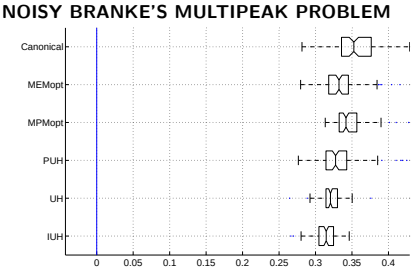

NOISY ROSENBROCK PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
2
4
6
8
10

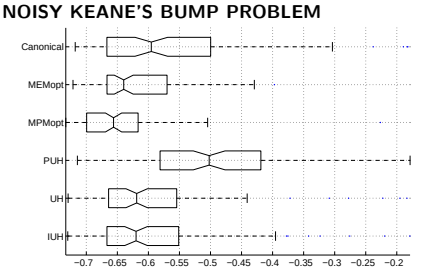

NOISY ACKLEY PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
0.5
1
1.5
2
2.5
3
3.5


NOISY GRIEWANK PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
0.5
1
1.5


NOISY RASTRIGIN PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
5
10
15
20
25
30


NOISY SCHAFFER'S F7 PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
1
2
3
4
5
6
7
8


NOISY BRANKE'S MULTIPEAK PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4


NOISY KEANE'S BUMP PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH
−0.7
−0.65
−0.6
−0.55
−0.5
−0.45
−0.4
−0.35
−0.3
−0.25
−0.2

6.3.3.2 Results CMA-ES

NOISY SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Canonical	0.58	0.24	0.54	52492	6
MEMopt	2.03	8.76	0.20	32436	5
MPMopt	**0.14**	**0.09**	**0.12**	**19068**	**1**
PUH	0.16	0.08	0.15	22805	2
UH	0.18	0.07	0.16	26880	4
IUH	0.79	6.19	0.17	26619	3

NOISY ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.96	3.72	1.30	46425	6
MEMopt	1.86	4.45	0.83	33230	5
MPMopt	**0.35**	**0.25**	**0.28**	**10562**	**1**
PUH	1.37	4.36	0.69	28530	2
UH	1.24	2.78	0.77	31731	4
IUH	0.98	1.42	0.70	29822	3

NOISY STEP ELLIPSOID PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.17	1.03	1.00	27668	3
MEMopt	1.11	2.02	1.00	25590	2
MPMopt	**0.30**	**0.73**	**0.00**	**16698**	**1**
PUH	1.88	7.00	1.00	32718	4
UH	1.90	5.55	1.00	38420	5
IUH	1.20	2.37	1.00	39206	6

NOISY ROSENBROCK PROBLEM

	Mean	Std	Med	∑#	#
Canonical	10.08	6.11	9.83	38624	6
MEMopt	56.56	327.76	9.25	31418	3
MPMopt	**8.64**	**0.63**	**8.67**	**21359**	**1**
PUH	56.58	402.82	9.34	31974	5
UH	66.85	352.85	9.22	31785	4
IUH	8.73	0.99	8.75	25140	2

NOISY ACKLEY PROBLEM

	Mean	Std	Med	∑#	#
Canonical	2.14	1.21	2.11	43224	5
MEMopt	0.80	1.57	0.43	17300	2
MPMopt	**0.35**	**0.41**	**0.28**	**8159**	**1**
PUH	2.19	1.27	2.03	44090	6
UH	1.55	1.33	1.28	34218	4
IUH	1.45	1.35	1.15	33309	3

NOISY GRIEWANK PROBLEM

	Mean	Std	Med	∑#	#
Canonical	1.25	0.26	1.23	47773	6
MEMopt	1.11	0.18	1.10	31302	5
MPMopt	**1.05**	**0.07**	**1.05**	**19048**	**1**
PUH	1.12	0.28	1.08	27034	2
UH	1.08	0.06	1.08	27569	3
IUH	1.08	0.07	1.08	27574	4

NOISY RASTRIGIN PROBLEM

	Mean	Std	Med	∑#	#
Canonical	15.69	12.86	13.94	35374	5
MEMopt	15.10	11.95	13.36	34359	4
MPMopt	**4.61**	**7.33**	**3.25**	**7810**	**1**
PUH	16.25	14.13	14.20	36903	6
UH	14.62	8.53	13.37	33823	3
IUH	14.70	14.38	12.32	32031	2

NOISY SCHAFFERS F7 PROBLEM

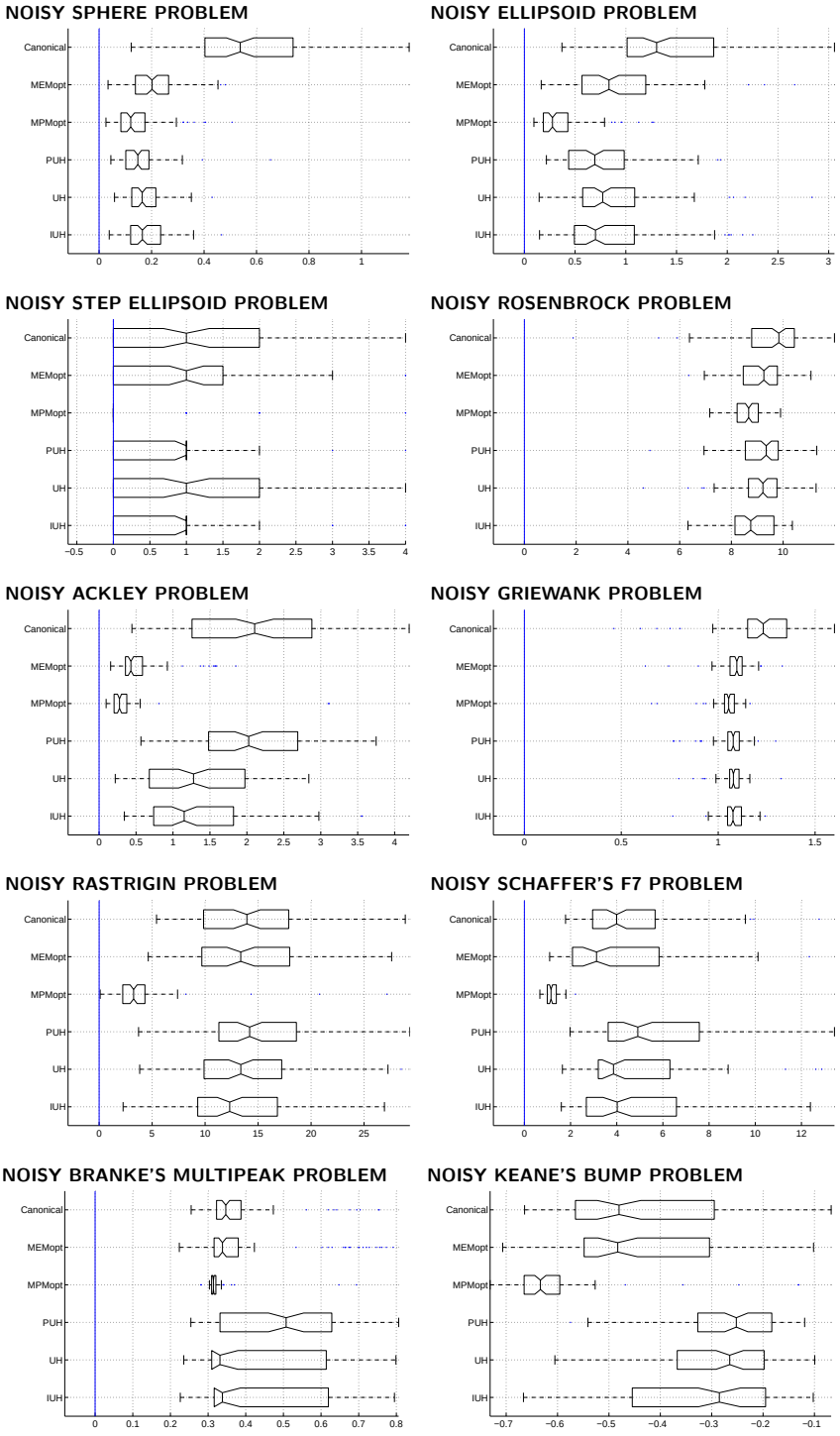
	Mean	Std	Med	∑#	#
Canonical	5.32	6.62	4.00	34220	3
MEMopt	4.78	6.55	3.12	29294	2
MPMopt	**2.96**	**10.21**	**1.15**	**6829**	**1**
PUH	6.25	6.34	4.91	39970	6
UH	5.04	2.91	3.86	35461	5
IUH	5.64	7.15	4.02	34526	4

NOISY BRANKE MULTIPEAK PROBLEM

	Mean	Std	Med	∑#	#
Canonical	0.39	0.12	0.35	32235	5
MEMopt	0.40	0.15	0.34	30623	3
MPMopt	**0.32**	**0.05**	**0.31**	**17366**	**1**
PUH	0.49	0.15	0.51	38577	6
UH	0.42	0.16	0.33	29348	2
IUH	0.44	0.16	0.34	32151	4

NOISY KEANE BUMP PROBLEM

	Mean	Std	Med	∑#	#
Canonical	-0.44	0.17	-0.48	26056	2
MEMopt	-0.43	0.16	-0.48	26398	3
MPMopt	**-0.62**	**0.10**	**-0.63**	**8754**	**1**
PUH	-0.27	0.10	-0.25	41856	6
UH	-0.28	0.11	-0.26	40209	5
IUH	-0.32	0.15	-0.29	37027	4


NOISY SPHERE PROBLEM
NOISY ELLIPSOID PROBLEM
NOISY STEP ELLIPSOID PROBLEM
NOISY ROSENBROCK PROBLEM
NOISY ACKLEY PROBLEM
NOISY GRIEWANK PROBLEM
NOISY RASTRIGIN PROBLEM
NOISY SCHAFFER'S F7 PROBLEM
NOISY BRANKE'S MULTIPEAK PROBLEM
NOISY KEANE'S BUMP PROBLEM
Canonical
MEMopt
MPMopt
PUH
UH
IUH

6.4 Summary and Discussion

In this chapter, the technique of adaptive averaging was studied in more detail, particularly focusing on poset based adaptive averaging (PUH), rank-change based adaptive averaging (UH), and rank-inversions based adaptive averaging or (IUH).

A theoretical study is presented on the growth rate of the sample size in case of an optimally adapted sample size for the noisy sphere problem. For the $(\mu/\mu_I, \lambda)$-ES on the noisy sphere, it is shown that the sample size must grow quartically with respect to the distance to the optimum to keep positive or optimal progress. Consequently, to achieve linear convergence over the number of generations, the sample size must grow exponentially. Hence, a multiplicative update rule should be used for the sample size within inter-generation adaptive resampling methods.

Furthermore, optimal settings have been derived for the adaptive averaging techniques for the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, based on an empirical study on the noisy sphere problem (see Table 6.1). The adaptive averaging techniques seem to be quite robust for different settings of the uncertainty threshold δ/θ and the growth rate of the sample size α.

Finally, from the empirical study in the last part of this chapter we conclude that the adaptive averaging techniques yield results that are comparable to optimally tuned static noise handling techniques. That is, except for the implicit averaging scheme in case of the CMA-ES, which clearly outperforms the other schemes. Hence, in terms of sampling efficiency (see Section 5.6) the adaptive averaging techniques do not yield a gain by starting out with a few samples for each individual and gradually increase the sample budget. The exponential growth required to achieve positive progress possibly negates the gain in earlier generations. On the other hand, advantages of using adaptive averaging over non-adaptive techniques are that they do not require the a priori setting of a sample size or population size and that they allow for arbitrary convergence accuracy. The newly introduced algorithmic parameters (the uncertainty threshold δ/θ and the growth rate of the sample size α) are more robust than the sample size (i.e., weak parameter reduction is achieved). Of the adaptive averaging schemes, the rank-change based adaptive averaging scheme (UH) and the inversions-based adaptive averaging scheme (IUH) provide the most promising results.

In conclusion, for solving noisy optimization problems, well-tuned implicit or explicit averaging techniques are simple but effective ways to counter the effects of noise. The rank-change based and rank-inversions based adaptive averaging technique can yield results comparable to well-tuned static noise handling schemes and are therefore well-suited alternatives when it is not possible to tune the static noise handling approaches.

As a future direction, in the context of global intermediate recombination, instead of increasing the sample size, it might be interesting to consider adaptive averaging techniques based on increasing the population size.

Chapter 7

Finding Robust Optima

This chapter focuses on the scenario depicted in Figure 7.1. In this scenario we have to deal with the fact that in the real-world system, the design variables cannot be set arbitrarily precise, yet there is a (simulation) model in which the solutions can be evaluated arbitrarily precise and that replaces the real-world system. Hence, the aim is to find optima that are actually useful in practice even when the real-world realizations of these solutions differ from their theoretically optimal or desired settings due to uncertainties and noise.

The intent of this chapter is to answer the following questions: 1) How can the aim of finding robust optima be modeled in the optimization problem statement? 2) How does the aim to find robust optima affect the difficulty of an optimization problem? 3) How should Evolution Strategies be adapted in order to find robust optima?

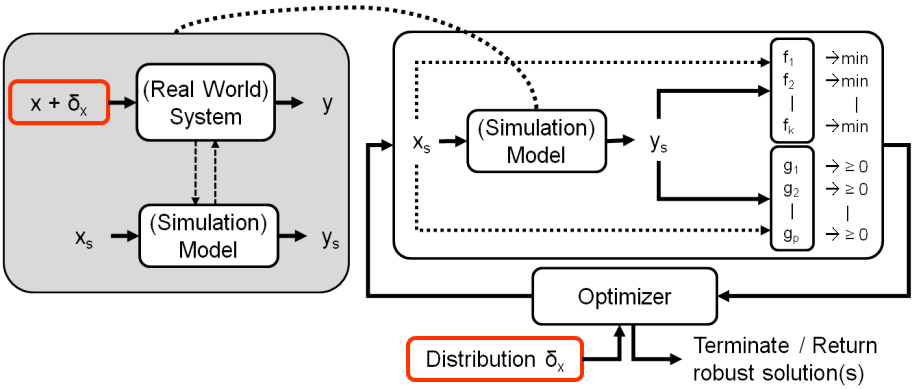


Figure 7.1: A typical robust optimization scenario: there is a real-world system in which the solutions cannot be realized arbitrarily precise. For the optimization, a (simulation) model (for which the solutions can be evaluated arbitrarily precise) replaces the real-world system; the aim is to find optima that are robust with respect to the anticipated input disturbances.

This chapter starts in Section 7.1 with providing an overview of the main concepts of the problem of finding robust optima for general real-parameter optimization problems. Section 7.2 focuses on the application of Evolution Strategies to a more specific class of optimization problems, namely on unconstrained single objective optimization problems. Here, we will restrict ourselves to a particular type of noise and a particular type of robustness measure. For this, we summarize and compare several techniques that have been proposed in the literature. Section 7.3 closes with a summary and discussion.

7.1 Problem Definitions for Finding Robust Optima

For the scenario depicted in Figure 7.1, the objective and constraint functions are of the same form as a normal real-parameter optimization problem, however, instead of aiming to find optima for the normal objective and constraint functions, the aim is to find solutions that are optimal for the problem formulation

$$f_i(\mathbf{x} + \boldsymbol{\delta}_\mathbf{x}) \rightarrow \min, i = 1, \ldots, k, \tag{7.1}$$

$$g_j(\mathbf{x} + \boldsymbol{\delta}_\mathbf{x}) \geq 0, j = 1, \ldots, p. \tag{7.2}$$

Here, $\boldsymbol{\delta}_\mathbf{x}$ denotes the uncertainty or possible variability of the input variables, of which the distribution $\mathrm{pdf}_{\boldsymbol{\delta}_\mathbf{x}}$ is assumed to be known. Hence, for each candidate solution, the objective and constraint functions become uncertain variables of which the possible variation depends on the quality of neighboring solutions that can be possible perturbations of that candidate solution. Or, in other words, the quality of a candidate solutions $\mathbf{x}$ is based on the η-*neighborhood*, defined as:

Definition 7.1.1 (η-neighborhood)**:** For a real-parameter optimization problem $(\mathcal{X}, \mathcal{F}, \mathcal{G})$ for which disturbances of the design variables and environmental parameters are anticipated, the neighborhood $\eta_\mathbf{x}$ is the set of possible disturbances of $\mathbf{x}$, i.e.,

$$\eta_\mathbf{x} = \{\mathbf{x}' \mid \mathbf{z} = \mathbf{x} - \mathbf{x}' \text{ and } \mathrm{pdf}_{\boldsymbol{\delta}_\mathbf{x}}(\mathbf{z}) > 0\}. \tag{7.3}$$

Similar to when having noisy objective functions, also for uncertainty and/or noise in the design variables we can distinguish between stationary and non-stationary noise/uncertainty. That is, in a stationary noise case, $\boldsymbol{\delta}_\mathbf{x}$ is independent of $\mathbf{x}$ and hence can simply be written as $\boldsymbol{\delta}$. Otherwise, $\boldsymbol{\delta}_\mathbf{x}$ is said to be non-stationary. In this work we will restrict ourselves to stationary uncertainty/noise and henceforth we will use the notation $\boldsymbol{\delta}$ for denoting the input perturbations.

Although the idea behind the desire to find robust optima is clear, there are different (possibly conflicting) ways to define a quality measure or *robustness measure*. That is, to define a measure that incorporates both quality and robustness. Choosing a robustness measure is a design choice that should be made when modeling a given problem as an optimization problem. For every objective and every constraint, one should consciously ask in which respect

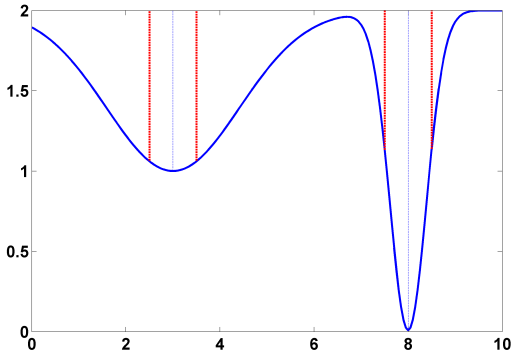


Figure 7.2: Illustration of the trade-off between quality and stability: with variation $\delta \sim \mathcal{U}(-\sigma_\epsilon, \sigma_\epsilon)$ in design variable x, optimizer $x_1 = 3$ is the more stable, but optimizer $x_2 = 8$ is still qualitatively better, given the variation bandwidth $\sigma_\epsilon = 0.5$.

solutions are desired to be robust. Moreover, the aim for robustness may be different for each objective and each constraint. Especially between objectives and constraints there may be different aims for robustness. In the context of robust design, Park [Par07] notes that there is a distinction between *reliability design* and *robust design*. In robust design the emphasis lies on low sensitivity with respect to the objective function(s), while in reliability design the emphasis lies on assuring that the constraints are not violated.

7.1.1 Robustness Measures for Objective Functions

For objective functions, the notion of robustness refers to the quality of a solution with respect to variations caused by uncertainties and noise. As noted by Jin and Sendhoff [JS03], the aim for robustness can be approached from two points of view that are often conflicting: robustness and performance (or: quality and stability):

- **Performance / quality**: Robustness of a solution is measured from the perspective of **overall performance** under the variation of the uncertain parameters of the solution.
- **Robustness / stability**: The robustness of a solution is measured from the perspective of **minimal performance variation** under the variation of the uncertain parameters of the solution.

Figure 7.2 illustrates the trade-off between quality and stability. It shows the fitness landscape of a one-dimensional, single objective, real-parameter optimization problem with design variable x that has a uniform uncertainty variation $\delta \sim \mathcal{U}(-\sigma_\epsilon, \sigma_\epsilon)$, with $\sigma_\epsilon = 0.5$. One could choose here for the most stable optimum at $x_1 = 3$ which has almost constant performance with respect to variations of x, or one could choose for the qualitatively best solution $x_2 = 8$, which has a better overall objective function value than x_1, but is less stable.

Although the choice for quality or stability generally provides an indication of what is meant by robust quality, there are still varying ways in which these aims can be translated into a robust quality measure. That is, a formulation of *effective* or *robust* objective functions is required to capture the aim for robustness in a concrete measure. Commonly used measures for robust quality are:

- **Expected solution quality**: The expected performance under variation of the uncertain design variables. This measure is considered in, amongst others, [TG97, WHB98, Tsu99, Bra98, Bra01, TG03, BS06a, ONL06, PBJ06]. For an objective function $f(\mathbf{x})$, this measure is described as an *expected objective function* as:

$$f_{\text{eff}}(\mathbf{x}) = f_{\text{exp}}(\mathbf{x}) = \mathbf{E}\left[f(\mathbf{x}+\boldsymbol{\delta}) \,|\, \mathbf{x}\right] = \int_{\mathbf{z}\in\mathbb{R}^n} f(\mathbf{x}+\mathbf{z})\ \text{pdf}_{\boldsymbol{\delta}_\mathbf{x}}(\mathbf{z})\ d\mathbf{z}. \tag{7.4}$$

  Obviously, this measure requires that the uncertainties in the design variables $\boldsymbol{\delta}$ are of stochastic nature, or at least can be modeled as such, with a probability density function $\text{pdf}_{\boldsymbol{\delta}_\mathbf{x}}(\mathbf{z})$.

- **Worst-case solution quality**: The worst-case quality with respect to the alternatives under variation of the uncertain variables (considered in, e.g., [LOL05, ONL06]). For an objective function $f(\mathbf{x})$ that is to be minimized, the robust objective function is determined as the maximum function value in the η-neighborhood of each candidate solution. That is,

$$f_{\text{eff}}(\mathbf{x}) = f_{\text{wc}}(\mathbf{x}) = \sup_{\mathbf{x}'\in\eta_\mathbf{x}} f(\mathbf{x}'). \tag{7.5}$$

  Note that for this measure, the η-neighborhood should be bounded and candidate solutions should have distinct η-neighborhoods in order to obtain distinct objective function values.

- **Threshold acceptance probability**: The maximal probability that a perturbation of a solution satisfies a certain threshold (considered in, e.g., [BS06b]). Or, in other words, maximizing the probability that a perturbation of a solution is part of the L_q level set. Given a threshold value q that indicates the acceptable solution quality, the optimization goal is to maximize the conditional probability of the objective functions satisfying this threshold:

$$f_{\text{eff}}(\mathbf{x}) = f_{\text{sat}}(\mathbf{x}) = \text{P}[f(\mathbf{x}+\boldsymbol{\delta}) \leq q \,|\, \mathbf{x}] \rightarrow \max. \tag{7.6}$$

Likewise, commonly used measures for stability are:

- **Performance variance:** Minimizing the performance variance under variation of the uncertain design variables. In this case, the original objective function $f(\mathbf{x})$ can be

remodeled into a robust objective function $f_{\text{eff}}(\mathbf{x})$ as to minimize the conditional variance:

$$f_{\text{eff}}(\mathbf{x}) = f_{\text{var}}(\mathbf{x}) = \text{Var}[f(\mathbf{x} + \boldsymbol{\delta}) \,|\, \mathbf{x}] \rightarrow \min . \tag{7.7}$$

Similar to the expected objective function Eq. 7.4, this measure requires that the uncertainties in the design variables $\boldsymbol{\delta}$ are of stochastic nature or at least can be modeled as such. Moreover, this measure should be accompanied by an additional objective that also takes solution quality into account. In [JL02] and [JS03] methods are proposed that implement this measure in a multi-objective setting by combining it with the optimization of the expected objective function.

- **Sensitivity region / trust region:** Maximization with respect to the region around the candidate solutions in which the deviation in performance is still acceptable. This measure can be used when the solutions need to be as stable as possible and is also frequently accompanied by an additional objective that aims for the optimization of the initial objective function. Examples of approaches that adopt this aim can be found in [BA06] and [LAA05].

  Mathematically, in a more general sense, a robust objective function $f_{\text{eff}}(\mathbf{x})$ implementing this aim can be constructed as:

$$f_{\text{eff}}(\mathbf{x}) = \sup\{r \in \mathbb{R}_+ \,|\, B_r(\mathbf{x}) \subseteq L(\mathbf{x})\} \rightarrow \max, \tag{7.8}$$

  with

$$L(\mathbf{x}) = \{\mathbf{x}' \,|\, f(\mathbf{x}') \in [f(\mathbf{x}) - q, f(\mathbf{x}) + q]\}, \tag{7.9}$$

$$B_r(x) = \{\mathbf{x}' \in \mathbb{R}^n \,|\, d(\mathbf{x}, \mathbf{x}') < r\}. \tag{7.10}$$

  Here, $L(\mathbf{x})$ denotes the set of all points which are within the region of tolerance, and $B_r(\mathbf{x})$ denotes the set of points within radius r from the point $\mathbf{x}$ (hence, it aims to maximize a sphere-like region). The distance measure $d(\mathbf{x}, \mathbf{x}')$ should be proportional or related to the probability of obtaining $\mathbf{x}'$ from $\mathbf{x}$. For example, in Euclidean spaces, the Euclidean distance is a suitable distance measure for uncorrelated Gaussian noise, and for uncorrelated uniform noise, the Tchebychev distance is a suitable distance measure.

7.1.2 Robustness of Constraint Satisfaction

For constraints, the aim for robustness is different. Ideally constraints are always satisfied, however, this is not always possible. As noted by Samsatli et al. [SPS98] a distinction can be made a between *hard constraints* and *soft constraints*.

Hard constraints are the constraints that may not be violated under any circumstances, and should therefore hold under for all possible variations in $\eta_{\mathbf{x}}$, i.e.,

$$g_{\text{eff}}(\mathbf{x}) = \inf_{\mathbf{x}' \in \eta_{\mathbf{x}}} g(\mathbf{x}') \geq 0. \tag{7.11}$$

Also here, the η-neighborhood should be bounded in order to be of practical use.

Soft constraints are less strict and may occasionally be violated (as long as the probability of violation is small). These types of constraints can be accounted for in different ways:

- **Probabilistic constraints**: One possibility is to redefine a constraint function $g(\mathbf{x})$ in a probabilistic form, where the probability of satisfying the constraint should be above a certain threshold c, i.e.,

  $$g_{\text{eff}}(\mathbf{x}) = P\left[g(\mathbf{x} + \boldsymbol{\delta}) \geq 0 \,|\, \mathbf{x}\right] - c \geq 0. \tag{7.12}$$

  With $0 < c \leq 1$. Note that equation (7.11) can be defined in terms of equation (7.12) by setting $c = 1$.

- **Virtual bounds for uncertainty variations**: Soft constraints also arise in cases where the variation due to uncertainty is theoretically unbounded (e.g., Gaussian variation). In these cases it is theoretically not possible to require the satisfaction of each strict constraint, unless the constraint is independent of the particular uncertain variable. A solution to deal with theoretically unbounded variation in the design variables (besides using probabilistic constraints) is to assign virtual bounds to the uncertainty variations between which the uncertainty variations are practically always expected to be. For example: common practice for Gaussian uncertain variables is to assume $-6\sigma < \boldsymbol{\delta} < 6\sigma$ for every uncertain variable $\boldsymbol{\delta}$. This is also the general principle behind design for "six sigma" [KYG04]. The way in which this is modeled for a given constraint $g(\mathbf{x})$ is the following:

  $$g_{\text{eff}}(\mathbf{x}) = \inf_{\mathbf{x}' \in \eta'_{\mathbf{x}} \subset \eta_{\mathbf{x}}} g(\mathbf{x}') \geq 0, \tag{7.13}$$

  with $\eta'_{\mathbf{x}}$ being a proper bounded subset of the η-neighborhood of $\mathbf{x}$.

- **Transformation to objective functions**: Remodeling or transforming soft constraints as additional objective functions. Instead of requiring that a constraint is satisfied, one could also grade the degree by which a constraint is satisfied and use that as an additional objective function. For example, maximization of the satisfaction probability:

  $$f_{\text{eff}}(\mathbf{x}) = P\left[g(\mathbf{x} + \boldsymbol{\delta}) \leq 0 \,|\, \mathbf{x}\right] \rightarrow \max. \tag{7.14}$$

7.1.3 Multi-Objective Robustness Measures

When dealing with the trade-off between stability and quality, the robust optimization of one objective can also seen as a multi-objective task (see, e.g., [JS03]). Hence, individual objective

functions can be split up into multiple robust objective functions. By doing so, particularly the trade-off between stability and quality can be controlled. For constraints a translation of one constraint to multiple robust constraints is not reasonable. However, as shown in the previous section, certain soft constraints can be transformed into additional objective functions.

7.1.4 Robustness Transformations

Given the various different measures for robustness, an obvious next question is: does it really matter which robustness measure is used? Or, is it even worthwhile at all to use a robustness measure instead of performing an optimization on the objective functions in their normal form? Straightforwardly, the answer to these questions is: sometimes it matters and sometimes not; this depends on the optimization problem at hand. However, in order to give an example of how drastic the impact of choosing different robustness measures could be, we consider the following one-dimensional function and suppose that we want to minimize it over x in the interval $[0, 10]$:

$$\begin{aligned} f(x) &= 1 + f_1(x) + f_2(x), \\ f_1(x) &= \begin{cases} \left(\frac{x-4}{5}\right)^2 & , x < 4 \\ 1 & , \text{otherwise} \end{cases}, \\ f_2(x) &= -1.8 \cdot \exp\left(-\frac{(x-5)^2}{0.2}\right) - 2 \cdot \exp\left(-\frac{(x-7)^2}{0.1}\right). \end{aligned} \tag{7.15}$$

For this function, consider an uncertainty of $\delta \sim U(-\sigma_\epsilon, \sigma_\epsilon)$, $\sigma_\epsilon = 0.5$ of the design variable x and consider the following measures for robustness:

- optimize the expected objective function $f_{\text{exp}}(x)$,
- optimize the worst-case objective function $f_{\text{wc}}(x)$,
- optimize on the normal objective function $f(x)$.

Figure 7.3 shows the fitness landscapes of these three different measures. In this example both robust objective functions are very different from the original objective function. Moreover, the optimal solutions for all three measures lie in different parts of the search space. Hence, from this example it can be seen that considering uncertainty can change a problem very much. Although this does not have to be the case for every optimization problem, one should be aware of this, and realize that performing optimization without accounting for the uncertainty/noise in the input variables and/or environmental parameters might yield results of which the realizations are of very poor quality. For constraints, a similar message holds. That is, in some cases optimal solutions might be prone to feasibility failure when considering uncertainty in the design variables in which case it matters, yet in other cases it might not matter at all.

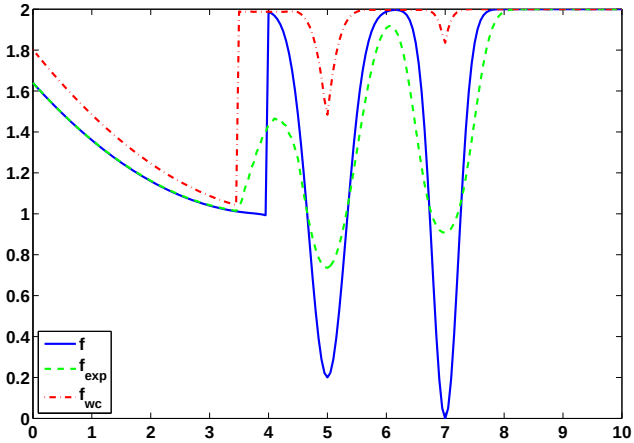


Figure 7.3: A one-dimensional objective function in which different measures of robustness produce different fitness landscapes, given uncertainty/noise in the design variables.

Besides the differences that occur due to the use of different robustness measures, another important factor is the magnitude of the anticipated variation. Small perturbations, for example, yield only small differences between the original objective function landscape and the effective objective function landscape, while large perturbations of the uncertainty/noise level may cause severe differences. Figure 7.4 illustrates how the original objective function landscape of the one-dimensional function Eq. 7.16 with input disturbances $\delta \sim \mathcal{U}(-\sigma_\epsilon, \sigma_\epsilon)$ changes for increasing values of σ_ϵ, considering the expected and worst-case objective function.

Mathematically, one can see the robustness measures described in the previous section as integral transformations (or filters) of the original objective functions and constraints. That is, a robustness measure is a transformation $T_{\text{eff}}(\mathbf{f}(\mathbf{x}), \mathbf{g}(\mathbf{x}))$ that maps the original objective and constraint functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ to new objective and constraint functions $\mathbf{f}_{\text{eff}}(\mathbf{x})$ and $\mathbf{g}_{\text{eff}}(\mathbf{x})$.

To view the problem of finding robust optima in terms of robustness transformations of the objective function landscape allows us to get some insight into the difference of normal optimization and the goal of finding robust optima. For example, consider the expected objective function transformation $f_{\text{exp}}(\mathbf{x})$ with Gaussian noise in the design variables. One can observe that this transformation is an integral transform with the original objective function convoluted with the Gaussian probability density function. More specific, the robustness transformation acts as a Gaussian filter or generalized Weierstrass transform (see, e.g., [Zay96]) from which we know that:

1. the effective objective function landscape should be smoother, hence easier for optimization since the spatial correlation is increased due to this smoothing operation,

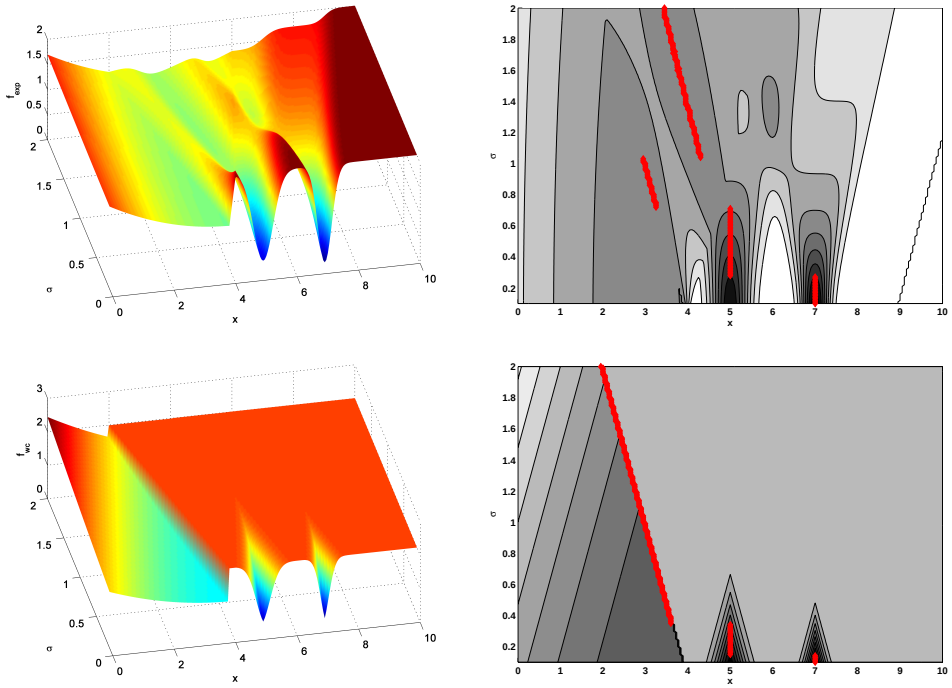


Figure 7.4: Illustration of the change of the effective objective function landscapes when varying the magnitude of the anticipated disturbances ($\sigma_\epsilon \in [0, 2]$) in the design variables. Top: the landscape dynamics for the expected solution quality. Bottom: the landscape dynamics for the worst-case solution quality. Left column: a surface plot of the landscape changing for different magnitudes of the input disturbances. Right column: contour plots showing also the locations of the optima for different uncertainty/noise magnitudes.

2. the extremal values (both minima and maxima) will lie between the extrema of the original objective function,

3. if the original objective function is an integrable function, then so is the effective objective function.

For other standard noise distributions besides Gaussian (e.g., uniform or Cauchy), these properties also hold. However, for different transformations than the expected objective function transformation, other properties might hold (e.g., the worst-case robustness measure).

Paenke et al. [PBJ06] classified the differences that could occur between the original objective function landscape and the effective objective function landscape in four categories:

- **Identical Optimum**: The original optimum and robust optimum are identical.
- **Neighborhood Optimum**: The original optimum and the robust optimum are located on the same hill (with respect to the original objective function landscape).
- **Local-Global-Flip**: An originally local optimum becomes the robust optimum.
- **Max-Min-Flip**: The robust optimum (for minimization) is located at a local maximum of the original objective function.

Figure 7.5 provides an example for each category of this classification.
As an alternative to this classification, we propose to classify robust optimizers by means of their explicit characteristics. This alternative classification is based on whether a robust optimizer is shifted with respect to some original global optimizer or whether it emerges anew:

Definition 7.1.2 (Shifted Robust Optimizer and Emergent Robust Optimizer)**:** Let σ denote a scaling of the magnitude of the input disturbance $\boldsymbol{\delta}$ and let $\mathbf{x}^*_\sigma$ denote a robust optimizer of $f_{\text{eff}\,\sigma}(\mathbf{x})$. That is, $f_{\text{eff}\,\sigma}(\mathbf{x})$ denotes the effective objective function for input perturbations $\boldsymbol{\delta}' = \sigma\boldsymbol{\delta}$. A robust optimizer $\mathbf{x}^*_{\sigma=1} = \mathbf{x}^*_{\text{eff}}$ is called a *shifted robust optimizer* if there exists a path $p : [0,1] \to X$ such that for all $t \in [0,1] : p(t) \in \text{argmin}_{\mathbf{x}\in X} f_{\text{eff}\,\sigma=t}(\mathbf{x})$, $p(0) = \mathbf{x}^*_{\sigma=0}$, and $p(1) = \mathbf{x}^*_{\sigma=1}$, where $\mathbf{x}^*_{\sigma=0} = \mathbf{x}^*$ is some optimizer for the original objective function $f(\mathbf{x})$. Otherwise, it is an *emergent robust optimizer*.

That is, if by gradually increasing the magnitude of the input disturbances the robust optimizer follows a connected path, it shifts. Otherwise, if a small increment of the noise magnitude leads to a completely different robust optimizer, it emerges anew, hence it is called emergent. The advantage of this classification is that it relates more closely to two major issues of finding robust optima: 1) accurately targeting robust optima (in case of shifts), and 2) targeting the right attractor regions in which a/the emergent robust optimizer is located (emergent robust optimizers). The first challenge relates to algorithms which are good in exploitation, the second challenge requires explorative strength.

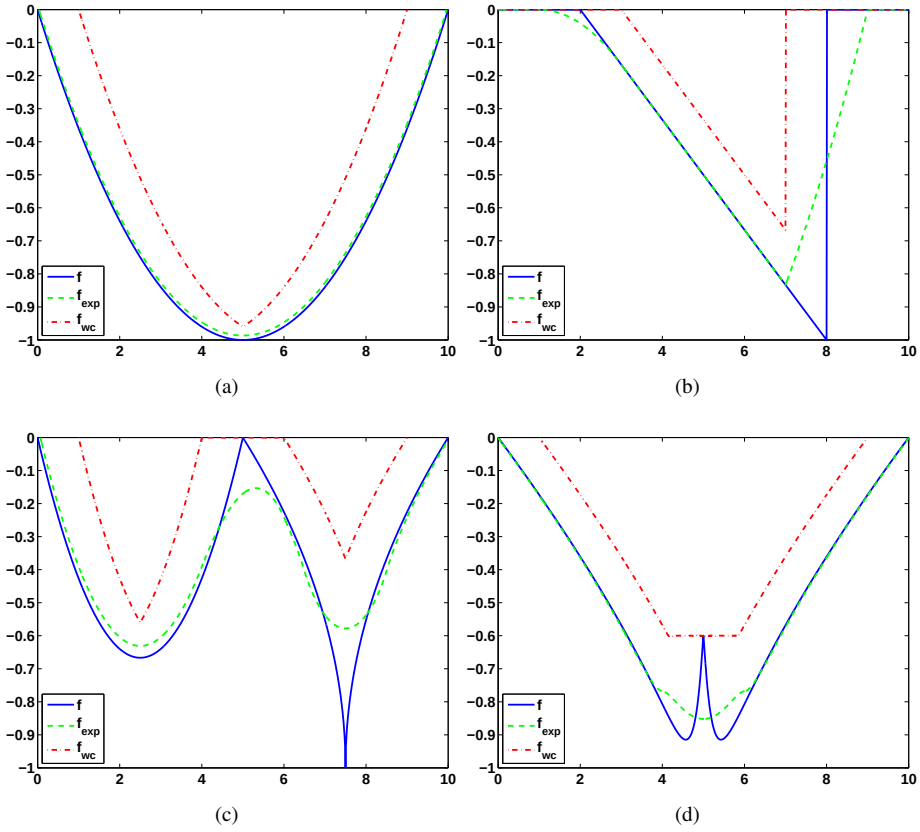


Figure 7.5: Illustrations of the four categories of differences in objective function landscape of the original objective function and the robust objective function according to Paenke et al. [PBJ06]: (a) identical optimum, (b) neighborhood optimum, (c) local-global-flip, (d) max-min-flip.

When we consider once again the function of Eq. 7.16 and look at the plots of Figure 7.4, we see in the right column that there are certain catastrophic events in which the optimum location "jumps" from one location to another. This happens three times for the expected objective function, and two times for the worst-case objective function. Besides these jumps, one can also identify shifts, for instance for the worst-case objective function, when $\sigma_\epsilon \gtrsim 0.4$. Moreover, besides these shifts and jumps, Sendhoff et al. [SBO04] show that input disturbances can even induce multimodality, meaning that an original optimizer might "split up" into multiple robust optimizers when increasing the noise strength. Lastly, it should be noted that two "robust optimizer branches" might also join when increasing the disturbance magnitude.

7.1.5 Computing the Robustness of Optima

The transformation of objective and constraint functions to functions that incorporate the notion of robustness is is the essence of the problem of finding robust optima. From this, we can identify two main issues that govern the problem of finding robust optima:

1. Choosing the appropriate robustness measure for the problem at hand.
2. Computing or approximating the robustness measure for candidate solutions.

The former is a part of the problem modeling, based on the preference of the user/expert and optimization problem context. The latter is a mathematical or algorithmic problem, which is challenging because exact computation of the solution quality for the robustness measures is often not possible.

In many cases it is either difficult or impossible to obtain exact derivations of the effective objective and constraint functions. Hence, alternative evaluation methods are needed that approximate the function values of the robustness measures, for instance, by sampling. Adopting the notation used in this work, we use $\hat{f}_{\text{eff}}(\mathbf{x})$ to denote an approximation function for the effective objective function $f_{\text{eff}}(\mathbf{x})$ and $\hat{g}_{\text{eff}}(\mathbf{x})$ to denote the approximation function for the effective constraint functions $g_{\text{eff}}(\mathbf{x})$.

There are two types of approaches to determine the robust quality of a solution: analytical approaches and statistical approaches. Analytical approaches determine the robustness of a solution by means of analytical techniques, such as using (approximations of) first-order and second-order derivatives in order to obtain an accurate estimate of the robust quality. Statistical methods determine the robustness of a solution by means of statistical approximation. An obvious example is to approximate the expected objective function by means of Monte-Carlo integration. In any case, it is clear that in order to determine the robust objective function value for a given solution $\mathbf{x}$, information is needed about the quality of the solutions in the neighborhood of $\mathbf{x}$. The challenge when searching for robust optima is to obtain good approximations of the robust quality of candidate solutions as efficiently as possible in order to perform optimization on the transformed objective function landscape.

7.2 Strategies for Finding Robust Optima

From here on, we will focus on unconstrained single-objective optimization problems and consider as robustness measure the expected fitness in the light of a known uncorrelated multivariate probability density function of the disturbances, which is either completely uniform or completely Gaussian. In this section, we will summarize approaches that can be used for Evolution Strategies to solve such problems. These approaches are categorized as:

- Myopic approaches (Section 7.2.1)
- Single- and multi-evaluation approaches (Section 7.2.2, Section 7.2.3, and Section 7.2.4)
- Adaptive averaging approaches (Section 7.2.5 and Section 7.2.6)
- Archive based approaches (Section 7.2.7)
- Metamodeling approaches (Section 7.2.8)
- Niching approaches (Section 7.2.9)
- Overlap exploiting approaches (Section 7.2.10)

7.2.1 The Myopic Approach for Finding Robust Optima

The simplest approach for finding robust optima is doing nothing extra, but to rely on the inherent attraction of Evolutionary Algorithms for finding robust optima. Branke [Bra98] states: "The standard EA by nature favors hills over peaks, since usually the probability that some individuals of the initial population are somewhere in the basin of attraction of a hill is higher, also the average fitness of individual in the hill area is higher.". Combining this observation with the fact that for some problems, the original optimizer is also the robust optimizer (or at least is located close to the robust optimizer), it is reasonable to consider normal Evolutionary Algorithms as good alternatives for finding robust optima as well. In this work, we will call this approach the *myopic approach*.

Experiment 7.2.1 (The inherent attraction of robust peaks)**:** For testing the inherent attraction of robust peaks, we perform 1000 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on a 10-dimensional instance of Branke's multipeak problem (see Appendix B.6). This problem consists of $2^n = 1024$ peaks varying in robustness. After each run, the peak score of the final solution is recorded (see Technical Note 7.1). For each run, an evaluation budget of $10,000$ function evaluations is considered.

Figure 7.7 shows histograms of the peak score of the final solution of 1000 optimization runs of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES on a 10-dimensional instance of Branke's multipeak problem. From these plots, the inherent attraction of robust peaks is very apparent. The $(5/2_{DI}, 35)$-σSA-ES converges to the robust peak in more than 65% of the runs and

Technical Note 7.1: The Peak Score of Branke's Multipeak Problem

Branke's multipeak problem is described in detail in Appendix B.6 and visualized in Figure 7.6. In the uncertainty free case, the optima are located at $\{-1, 1\}^n$, where the quality of the local optima is ranked by the number of positive elements of the optimizer, i.e., the global optimum is located at $[1]^n$, the "worst" local optimum is located at $[-1]^n$, and peaks with the same number of positive and negative elements are equivalent (hence, there are n classes of equivalent peaks).
When considering the expected objective function for an input uncertainty $\boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{1/2}, \mathbf{1/2})$, the order in peak quality is reversed. Hence, the most robust peak is located at $[-1]^n$, the least robust peak is located at $[1]^n$. We can construct a peak score ps that identifies the type of peak in which a solution $\mathbf{x}$ is located as

$$\mathrm{ps}(\mathbf{x}) = \sum_{i=1}^{n} \mathrm{signum}(x_i). \tag{7.16}$$

Here, $\mathrm{ps} = 0$ identifies the most robust peak and $\mathrm{ps} = n$ identifies the least robust, but highest peak. Using this peak score, we can analyze the attraction of the different peaks of myopic approaches and therefore the inherent attraction of robust peaks.

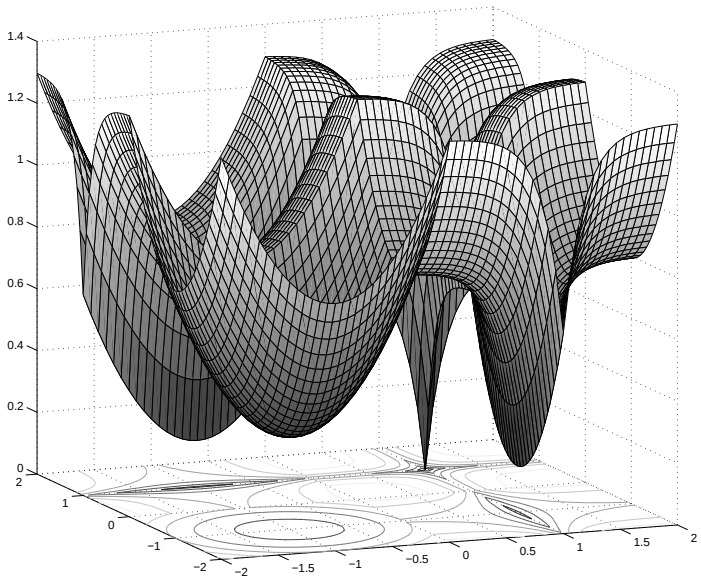


Figure 7.6: A plot of the objective function landscape of a two-dimensional instance of Branke's multipeak problem. This problem consists of 2^n peaks, varying in robustness.

shows a clear preference for the robust peak. The CMA-ES hits the robust peak in more than 30% of the runs and also shows a clear preference for the robust peak, but less clear than the $(5/2_{DI}, 35)$-σSA-ES.

In this small example, it can be seen clearly that the myopic approaches are biased towards

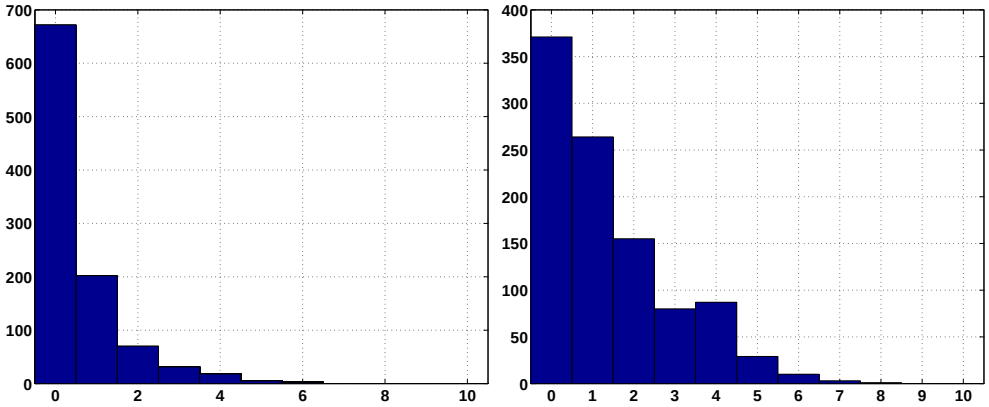


Figure 7.7: Histograms of the peak score of the final solution of 1000 optimization runs of the $(5/2_{DI}, 35)$-σSA-ES (left) and the CMA-ES (right) on a 10-dimensional instance of Branke's multipeak problem.

converging to the more robust peaks, making these approaches for the problem of identifying the more robust peaks already very promising. As an intuitive explanation of the inherent attraction of the more robust peaks as opposed to the sharper peaks, two factors play an important role: 1) the basin of attraction, and 2) the probability that a random point in the robust peak is better than a random point in the sharper peak. That is, the basin of attraction determines the probability of actually generating search points in a certain peak and the probability of a random solution in the robust peak being better than a random solution in the non-robust peak determines the possibility of the former solution to survive. For robust peaks, the latter probability can be assumed to be relatively high (i.e., the solutions in the neighborhood should be of relatively good quality, otherwise the peak is not robust). Hence, for Evolution Strategies, individuals located in these peaks have a high probability to survive and the population will be attracted to these peaks.

The tendency of myopic implementations to converge to the more robust peaks can be seen as undesirable from the perspective of classical optimization, but is advantageous when searching for robust optima. For this, myopic approaches could be serious alternatives for the problem of finding robust optima.

Experiment 7.2.2 (The robust precision of the myopic approach)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on a 10-dimensional instance of the Heaviside sphere problem (see Appendix B.2). The Heaviside sphere problem is a problem in which the robust optimizer is a shifted version of the uncertainty free optimizer. A two-dimensional instance of the Heaviside sphere problem is visualized in Figure 7.8. Each run has a budget of $10,000$ function evaluations.

Figure 7.9 shows the results of Experiment 7.2.2 in terms of the performance, measured in

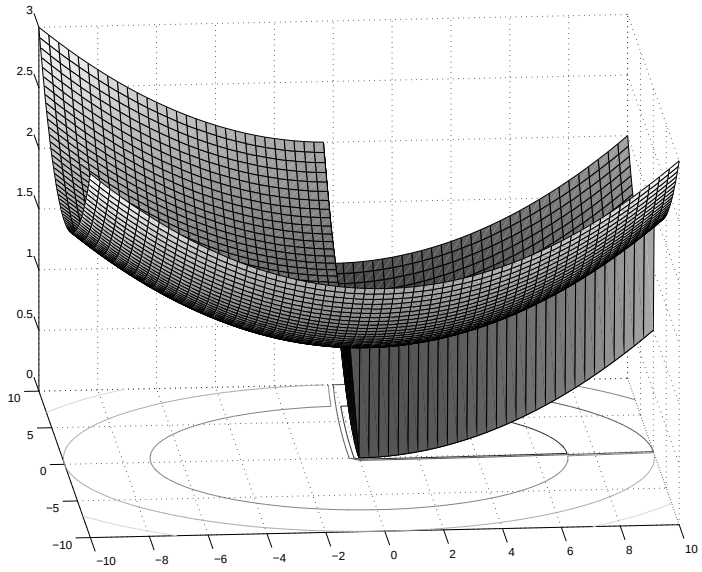


Figure 7.8: A plot of the objective function landscape of a two-dimensional instance of the Heaviside sphere problem. This problem consists one optimizer at the edge of the ridge. The robust optimizer will lie at a distance from the ridge.

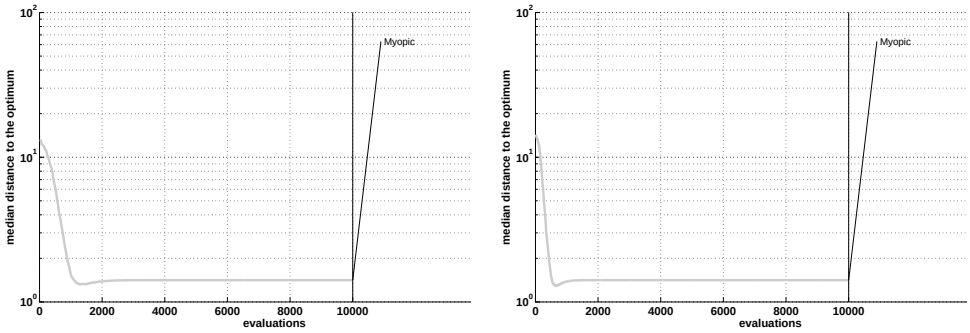


Figure 7.9: Results of Experiment 7.2.2. The performance of running canonical instances of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES on a 10-dimensional instance of the Heaviside sphere problem. The performance is measured in terms of median distance to the optimum. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

terms of median distance to the optimum. For this unimodal test problem with a shifted robust optimizer, it is apparent that myopic approaches will not always target the robust optimum. In this case, both algorithms target the uncertainty free optimum and not the robust optimum and find solutions that are sub-optimal with respect to the effective objective function. Moreover, in addition, even if myopic approaches get close to a robust optimizer, they are not able to detect this and can easily deviate from them again. In Figure 7.9 this "overshooting" behavior can be observed for both the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, where the stagnation distance is higher than the closest distance, reached at approximately 1000 and 500 evaluations respectively.

From the two small experiments performed in this section, we observe that the robust regions of the search space have an inherent attraction on canonical Evolution Strategies. That is, blunt peaks are preferred over sharp peaks of a similar height. For the $(5/2_{DI}, 35)$-σSA-ES it seems that this attraction is more pronounced than for the CMA-ES. However, when the robust optimizer is a shifted version of the uncertainty free optimizer, myopic approaches will naturally fail to zoom in on the robust optimizer with arbitrary precision. The challenge of finding robust optima is therefore to adapt these such that they can accurately target (or zoom in on) robust optima, and improve their already present bias towards the robust peaks.

7.2.2 Single- and Multi-Evaluation Methods

A straightforward way to approximate the expected solution quality is to use Monte-Carlo integration. That is, approximating $f_{\exp}(\mathbf{x})$ for a candidate solution $\mathbf{x}$ as

$$\hat{f}_{\exp}(\mathbf{x}) = \sum_{i=1}^{m} f(\mathbf{x} + \mathbf{z}_i) \ , \ \mathbf{z}_i \sim \boldsymbol{\delta}, \tag{7.17}$$

with $\mathbf{z}_1, \ldots, \mathbf{z}_m$ being m disturbances sampled from the distribution of the anticipated input noise. This approach was proposed originally by Tsutsui et al. [TGF96] for $m = 1$ and shortly after that generalized for arbitrary m in, amongst others, [Bra98, WHB98, Tsu99]. An obvious effect of this evaluation method is that it introduces a measurement error of the expected objective function, i.e., noise. This error can be decreased by increasing the number of samples m. Tsutsui [Tsu99] distinguished between the extreme case of taking only one sample for the evaluation and the generalized version, naming the former the *Single Evaluation Mode* (SEM) and the latter the *Multi Evaluation Mode* (MEM). Here, we also adopt this terminology.

We observe a clear resemblance when comparing this approximation method to the approximation of the expected objective function for noisy objective functions of Eq. 5.12. Essentially there is no difference between the two as one could see the objective function value for each disturbance $f(\mathbf{x} + \mathbf{z}_i)$ as a noisy evaluation of the expected objective function at $\mathbf{x}$. Hence, the problem of finding robust optima can, in principle, be transformed into the problem of optimizing a noisy objective function, bearing in mind that the noise introduced by this

approximation method is non-stationary and, more importantly, biased with respect to the original objective function $f(\mathbf{x})$. Properties of noise introduced by input perturbations have been studied in, e.g., [BOS03, SBO04, BS06b].

For notational convenience, in this work we will include the number of samples in the identification of the particular MEM scheme. For instance, MEM5 will refer to the MEM evaluation scheme with $m = 5$ samples and MEM10 refers to the MEM evaluation scheme with $m = 10$ samples.

In order to obtain an insight into how the SEM and MEM evaluation scheme influence the performance of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES and how they compare to canonical instances of these algorithms, we perform the following experiment:

Experiment 7.2.3 (Performance of the SEM/MEM evaluation scheme for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using different evaluation schemes: the canonical evaluation scheme, the SEM evaluation scheme, the MEM5 evaluation scheme, and the MEM10 evaluation scheme. The experiments are performed on the 10-dimensional test problems: the Heaviside sphere problem (see Appendix B.2) and Branke's multipeak problem (see Appendix B.6). Each run has a budget of $10,000$ function evaluations.

The results of Experiment 7.2.3 are shown in Figure 7.10 by means of plots of the development of the median fitness, a posteriori approximated using Monte-Carlo integration with $m = 100$ samples. For the Heaviside sphere problem, the canonical implementations of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES are clearly outperformed by the SEM/MEM evaluation schemes. That is, these canonical instances converge to the non-robust optimizer which is not the robust optimizer. Moreover, we observe the same effects of using more samples as can be seen with explicit averaging in case of noisy objective functions; using the SEM evaluation scheme yields a fast convergence, yet also stagnates at a certain distance to the optimizer. When using more samples for the MEM scheme, the convergence accuracy increases, but at the cost of slower convergence. For the sphere problem and for Branke's multipeak problem a major difference is the good performance of the myopic instances of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. For Branke's multipeak problem, this is surprising. Apparently, the bias of the myopic instances to converge to more robust peaks is already very high and combining this with the fact that the robust optimizer for this problem is a local optimizer of the original objective function, the canonical instances will converge with high precision and prove to be competitive alternatives.

Experiment 7.2.4 (The attraction of robust peaks)**:** For testing the attraction of robust peaks for the different MEM variants, we perform 500 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) on a 10-dimensional instance of Branke's multipeak problem (see Appendix B.6) using different evaluation schemes: the canonical evaluation scheme,

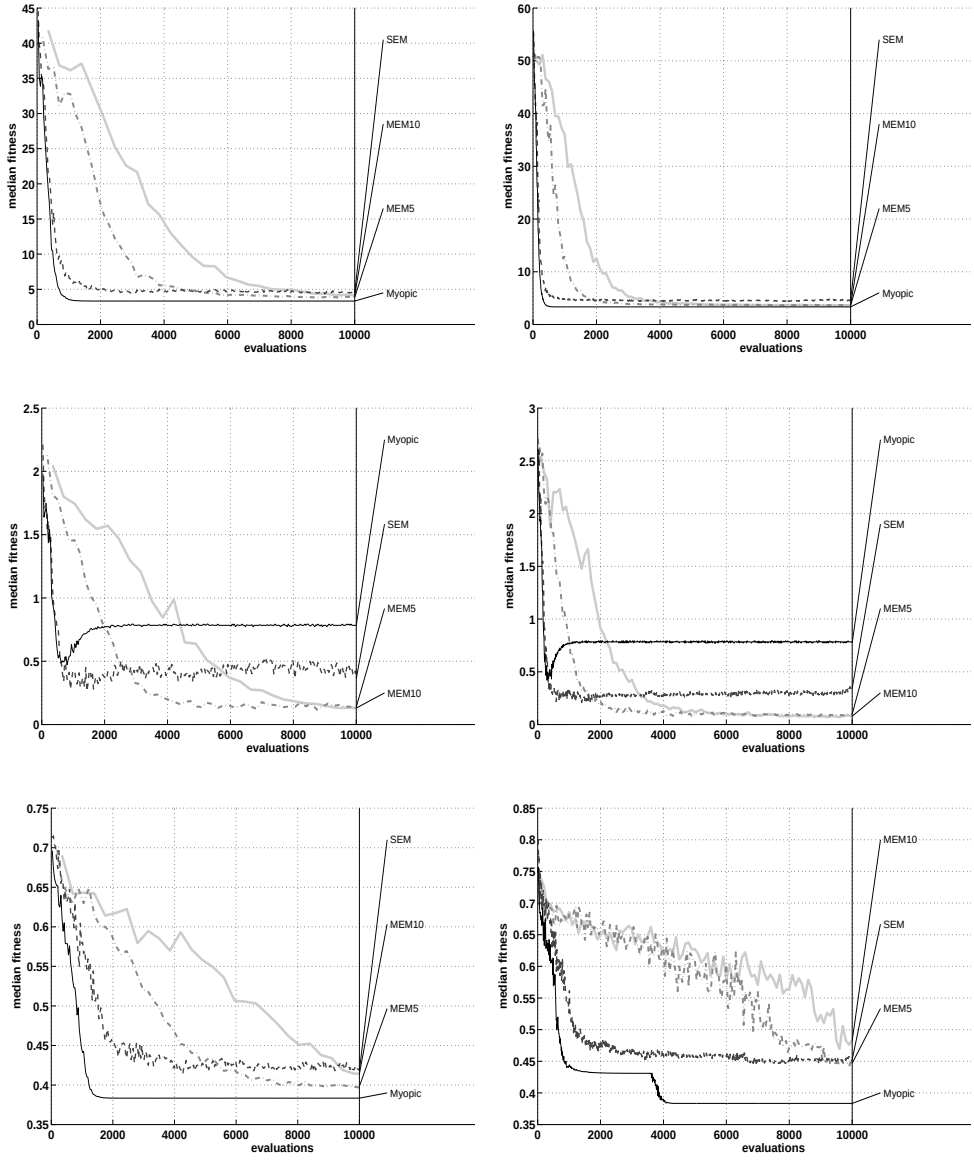


Figure 7.10: Results of Experiment 7.2.3. The performance of the $(5/2_{DI}, 35)$-σSA-ES (left column) and the CMA-ES (right column) using the SEM/MEM evaluation scheme with different sample sizes compared against canonical instances of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. The performance is measured in terms of median fitness approximated a posteriori using Monte-Carlo sampling with 100 samples. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere. Bottom row: the results on Branke's multipeak problem. The results are obtained using 50 runs for each scheme.

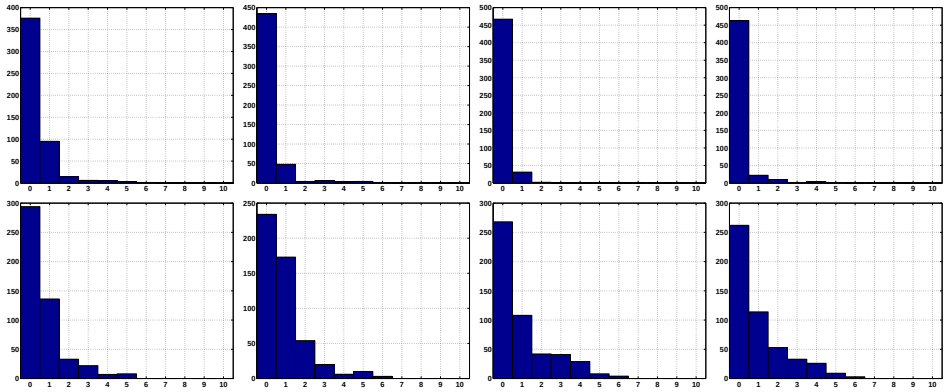


Figure 7.11: Histograms of the peak scores of the final solution of 500 optimization runs of the $(5/2_{DI}, 35)$-σSA-ES (top) and the CMA-ES (bottom) on a 10-dimensional instance of Branke's multipeak problem with varying evaluation schemes. From left to right: myopic, SEM, MEM5, and MEM10.

the SEM evaluation scheme, the MEM5 evaluation scheme, and the MEM10 evaluation scheme. For each run, an evaluation budget of $10,000$ function evaluations is considered.

The results of Experiment 7.2.4 are shown in Figure 7.11 by means of histograms of the peak scores of the final solution of all runs. For the $(5/2_{DI}, 35)$-σSA-ES it can be seen that the frequency of finding the most robust peak indeed slightly increases for MEM evaluation approaches. For the CMA-ES, almost the inverse seems to be happening. That is, comparing the myopic approach with the SEM approach for the CMA-ES, there is a huge drop in the frequency with which the robust optimizer is targeted. For the MEM5 and MEM10 approach, a slight improvement is obtained with respect to the SEM approach, but the bias for the myopic approach is still higher.

7.2.3 Reducing the Sampling Variance

For the MEM evaluation scheme, a modification proposed by Branke [Bra01] is to use Latin Hypercube Sampling (LHS) [MBC00] to obtain a well-distributed sample set of input disturbances in a box around the solution, and using these in a weighted way to determine the robustness, i.e.,

$$\hat{f}_{\exp}(\mathbf{x}) = \frac{\sum_{i=1}^{m} w(\mathbf{z}_i) f(\mathbf{x} + \mathbf{z}_i)}{\sum_{i=1}^{m} w(\mathbf{z}_i)}, \tag{7.18}$$

with $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\} \sim \mathrm{LHS}(-\boldsymbol{\sigma}_\epsilon, \boldsymbol{\sigma}_\epsilon)$ being the set of sample points drawn using Latin Hypercube Sampling from an appropriately set hypercube $[-\boldsymbol{\sigma}_\epsilon, \boldsymbol{\sigma}_\epsilon]$ and $w(\mathbf{z}_i) \propto \mathrm{pdf}_{\boldsymbol{\delta}}(\mathbf{z}_i)$. In [Bra01], this modification was shown to yield a better convergence accuracy compared to the normal MEM evaluation scheme, attributed to a reduced sampling variance among the fitness estimates of the individuals in the population. Additionally, an extra gain in convergence

accuracy was reported when using the same set of disturbances for all individuals in the population.

For notational convenience, we will use the following notation to distinguish between the different variants of the SEM/MEM scheme. A subscript is used for indicating the sampling method when using Eq. 7.18, i.e., $\mathrm{MEM}_{\mathrm{MS}}$ when using Monte-Carlo sampling, and $\mathrm{MEM}_{\mathrm{LHS}}$ when using Latin Hypercube Sampling. Furthermore, the superscripts $^+$ and $^-$ are used to indicate whether or not the same disturbances are used for all the individuals in the current population, i.e., MEM^+ when using the same disturbances for all individuals in the population and MEM^- when resampling the disturbances for every individual in the population. When no subscript or superscript is used (as in Experiment 7.2.3), we assume that the evaluation follows Eq. 7.17 and different disturbance samples for every individual in the population.

In order to obtain an insight into how these different variants of the MEM scheme influence the performance of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, we perform the following experiment:

Experiment 7.2.5 (Performance of basic techniques for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using the different variants of the MEM evaluation scheme, namely $\mathrm{MEM5}^-_{\mathrm{MC}}$, $\mathrm{MEM5}^+_{\mathrm{MC}}$, $\mathrm{MEM5}^-_{\mathrm{LHS}}$, and $\mathrm{MEM5}^+_{\mathrm{LHS}}$. The experiments are performed on the sphere problem (see Appendix B.1), the Heaviside sphere problem (see Appendix B.2), and Branke's multipeak problem (see Appendix B.6). Each run has a budget of $10,000$ function evaluations.

The results of Experiment 7.2.5 are shown in Figure 7.12 by means of plots of the development of the median fitness, a posteriori approximated using Monte-Carlo integration with 100 samples. For the sphere problem and for Branke's multipeak problem, the results confirm the results obtained by Branke [Bra01], i.e., using Latin Hypercube sampling yields better results than Monte-Carlo sampling and also using the same perturbations for all individuals in the current generations yields an additional gain in solution quality. The significance of these results can be read from the boxplots that are shown in Figure 7.13.

However, when looking at the results on the Heaviside sphere problem, remarkably different behavior can be observed. Here, especially for the CMA-ES, the LHS methods seem to be outperformed by the Monte-Carlo based methods and using the same perturbations for all individuals in the population yields a worse final solution quality. Looking at the convergence plot of the CMA-ES (Figure 7.12) we see that the schemes that reuse the same perturbations for all individuals in the current generation show divergent behavior for this particular test problem. The same divergent behavior is exhibited in the other schemes, but less drastically. For the $(5/2_{DI}, 35)$-σSA-ES, these effects are less severe. A possible explanation for this is that the two alterations of the MEM scheme exploit local symmetry, which is beneficial in case of the sphere and Branke's multipeak, but is harmful in case of the Heaviside sphere.

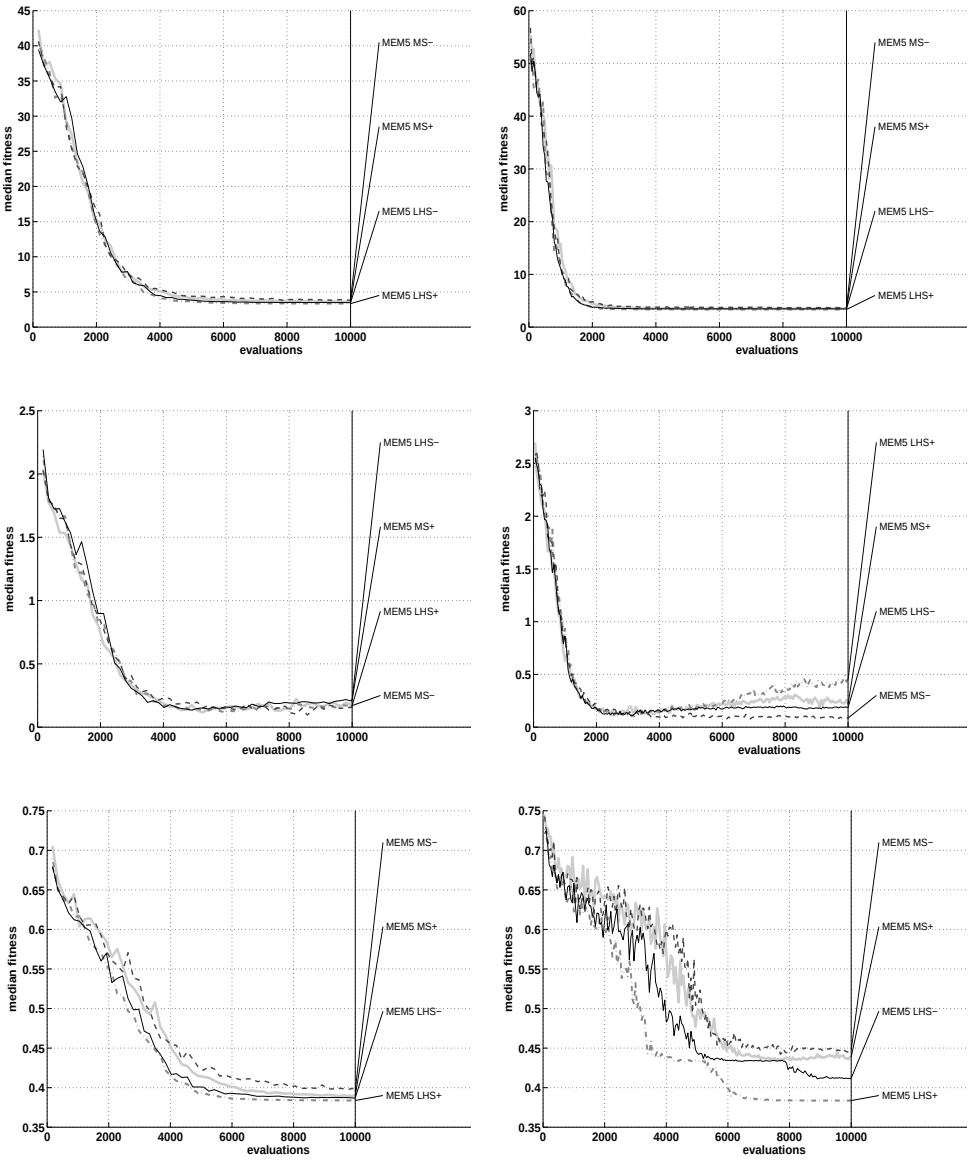


Figure 7.12: Results of Experiment 7.2.5. The performance of the $(5/2_{DI}, 35)$-σSA-ES (left column) and the CMA-ES (right column) using different variants of the MEM evaluation. The performance is measured in terms of median fitness approximated a posteriori using Monte-Carlo sampling with 100 samples. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere problem. Bottom row: the results on Branke's multipeak problem. The results are obtained using 50 runs for each scheme.

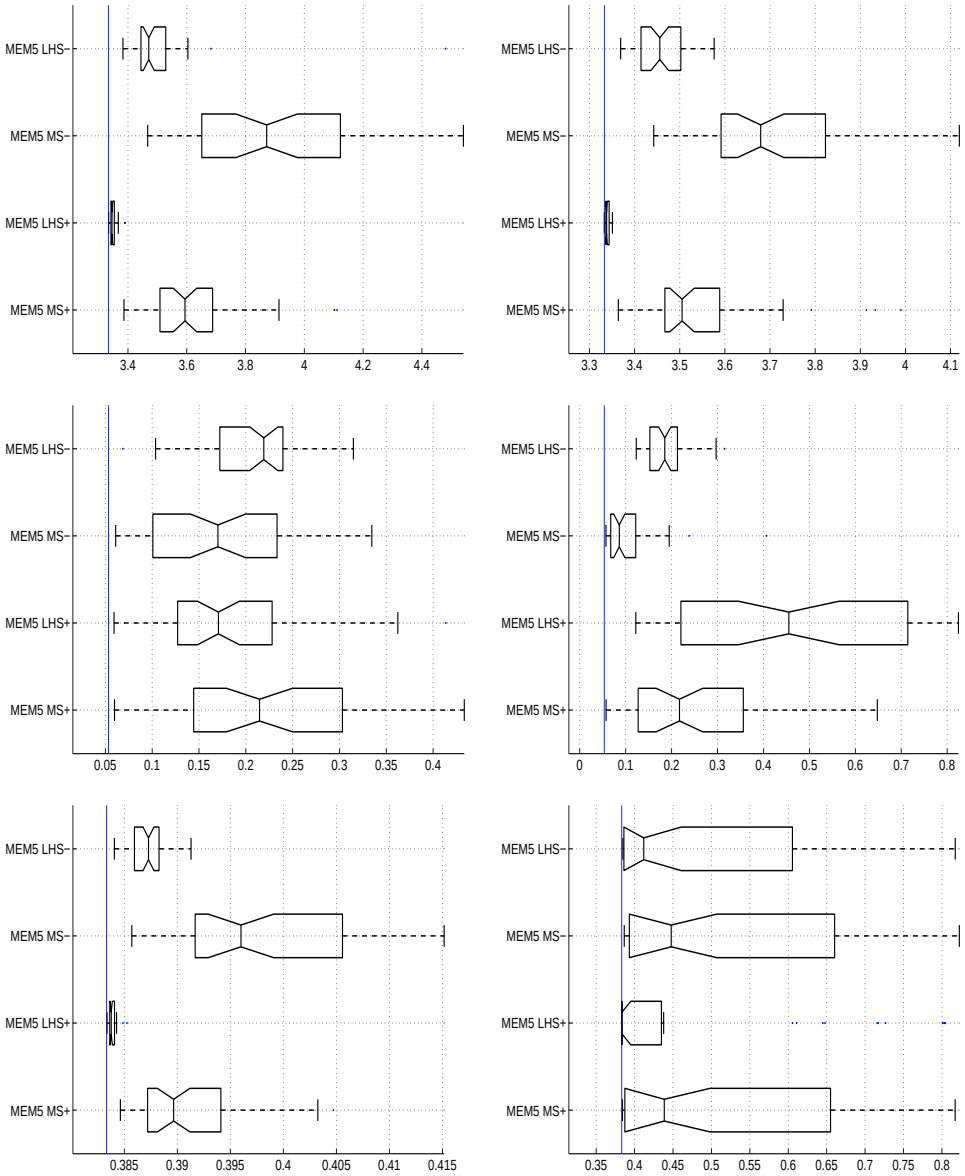


Figure 7.13: Results of Experiment 7.2.5. The final solution quality of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES using different variants of the MEM evaluation. The solution quality is approximated a posteriori using Monte-Carlo sampling with 1000 samples. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere problem. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

From these results we can conclude that the proposed alternatives can be beneficial in cases where the objective function landscape is symmetric around the robust optimizer. However, when considering local discontinuities around the optimizer that shift the robust optimizer, they might yield divergent behavior. In the remainder of this chapter, we will consider the extremal cases of Monte-Carlo sampling without reusing the same disturbances for all individuals in the population (MEM5$^{-}_{\mathrm{MC}}$) and Latin Hypercube sampling in combination with using the same disturbances for all individuals in the population (MEM5$^{+}_{\mathrm{LHS}}$) for further investigation.

7.2.4 Implicit Versus Explicit Averaging for Finding Robust Optima

For noisy objective functions, we have observed that implicit averaging is competitive to or even better than explicit averaging. This is true when considering Gaussian additive noise. The MEM evaluation approach for finding robust optima is essentially generating noisy fitness evaluations, however, for smaller values of m, this noise cannot be assumed to be Gaussian. An interesting question is therefore whether implicit averaging is also applicable in these scenarios. In order to obtain an insight into the differences between implicit and explicit averaging in this scenario, we perform the following experiment:

Experiment 7.2.6 (Implicit versus explicit averaging for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using two variants of the MEM evaluation scheme (using $m = 5$) and compare the results obtained with these two variants against two variants of the SEM scheme in which the population size is increased with a factor 5 (i.e., implicit averaging). The two MEM variants are MEM5$^{-}_{\mathrm{MC}}$ and MEM5$^{+}_{\mathrm{LHS}}$, and the two SEM variants are 5SEM^{-} and 5SEM^{+}. The experiments are performed on the sphere problem (see Appendix B.1), the Heaviside sphere problem (see Appendix B.2), and on Branke's multipeak problem (see Appendix B.6). Each run has a budget of $10,000$ function evaluations.

The results of Experiment 7.2.6 are shown in Figure 7.14 by means of plots of the development of the median fitness, a posteriori approximated using Monte-Carlo integration with 100 samples. From the results we see that, in general, the explicit resampling approaches outperform the implicit averaging schemes. That is, when not taking into account the divergent behavior obtained with the MEM5$^{+}_{\mathrm{LHS}}$ approach on the Heaviside sphere. Comparing this to the conclusion of the previous chapter, in which the implicit averaging scheme showed remarkably good performance, especially for the CMA-ES, this result is surprising. It is reasonable to assume that difference is caused by the fact that the noise that is introduced by using a SEM evaluation scheme is not Gaussian and non-stationary. However, this should be investigated in more detail.

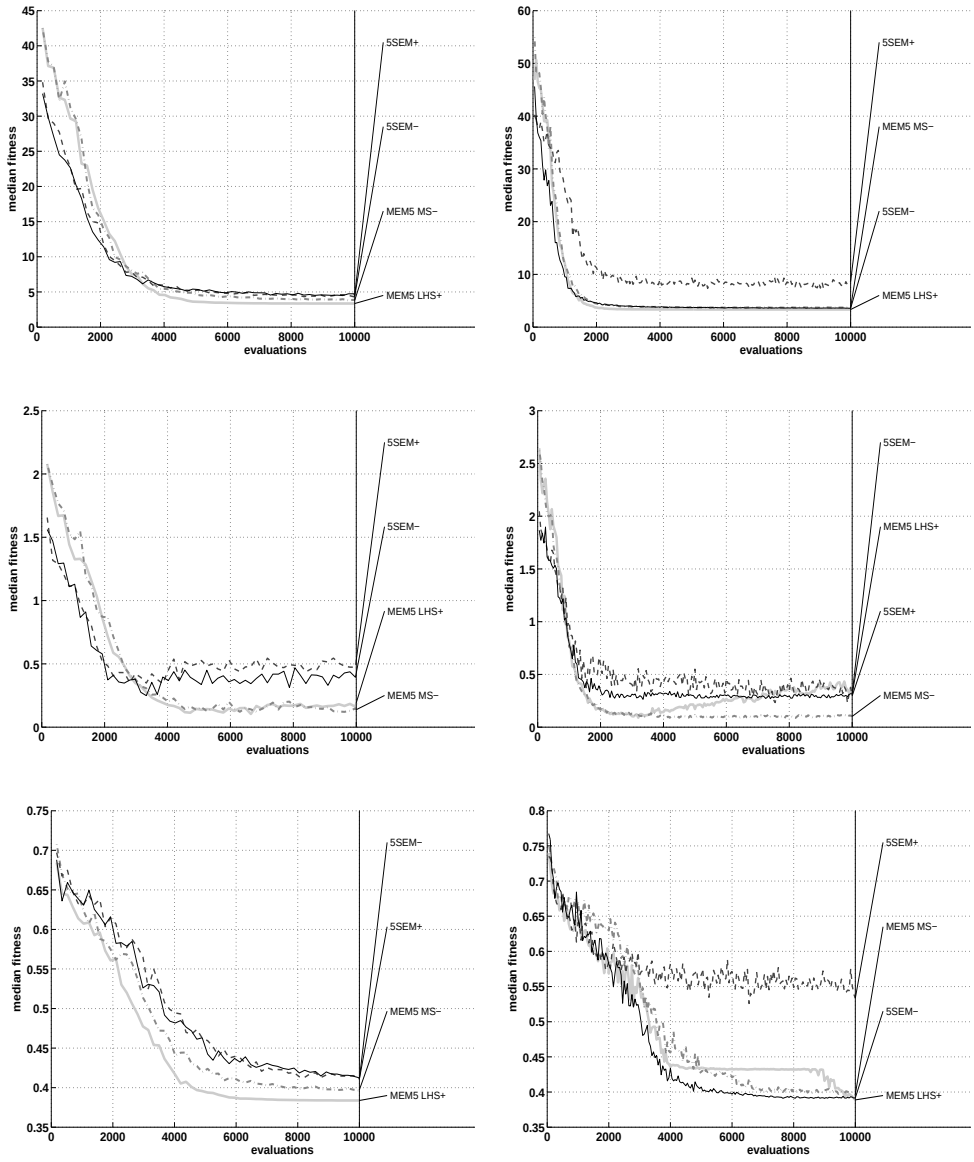


Figure 7.14: Results of Experiment 7.2.6. The performance, in terms of median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples) of different variants of the SEM/MEM evaluation scheme, comparing implicit versus explicit averaging. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

7.2.5 Adaptive Averaging for Finding Robust Optima

As mentioned in Section 7.2.2, using MEM evaluation methods for approximating the expected objective function is essentially the same as using explicit averaging for a noisy objective function. Moreover, also here we encounter the same trade-off between using few samples, yielding a low approximation accuracy or obtaining a high approximation accuracy at the cost of requiring many samples. A straightforward approach to deal with this problem is therefore to use the adaptive averaging techniques proposed in the context of noisy optimization for improving the convergence accuracy of schemes that aim to find robust solutions.

In [KRD+11] such an approach is followed. Here, the UH-CMA-ES proposed in [HNGK09] is adapted for resampling and the objective function is replaced by a $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ evaluation scheme. The incorporation of this idea is straightforward: for any of the adaptive averaging techniques, a MEM evaluation scheme using m samples can be used instead of m times resampling the noisy objective function. The scheme that we will consider in this work as an implementation of this idea is shown in Algorithm 7.1. Here, the rank-change based adaptive averaging technique as described in Section 5.4.5 is modified for incorporation of any MEM evaluation scheme.

In order get an impression of how such a scheme performs in this setting, we consider the following experiment:

Experiment 7.2.7 (Performance of adaptive averaging for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using a MEM evaluation scheme adopting the rank-change based adaptive averaging technique (see Section 5.4.5). The experiments are performed on the sphere problem (see Appendix B.1), the Heaviside sphere problem (see Appendix B.2), and on Branke's multipeak problem (see Appendix B.6). For the adaptive averaging schemes, the settings $\alpha = 1.2$ and $\theta = 0.6$ are used, according to [KRD+11]. Each run uses a budget of $10,000$ function evaluations.

The results of Experiment 7.2.7 are shown in Figure 7.15 by means of convergence plots of the median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples). From the figure it can be seen that for the unimodal problems, the convergence speed in the early stages is much higher, however, for the sphere problem, the normal $\mathrm{MEM5}^{+}_{\mathrm{LHS}}$ yields a higher convergence accuracy than the adaptive averaging schemes. For the Heaviside sphere problem, the adaptive averaging approaches yield a higher convergence accuracy than their non-adaptive counterparts. For Branke's multipeak problem, using adaptive averaging seems not to be beneficial and might even be noted to be worse. The latter complies with the results reported in [KRD+11] that adaptive averaging seems only beneficial for improving the local convergence accuracy, but it does not increase the tendency to converge to the more robust peaks.

Algorithm 7.1: Rank-Change Based Adaptive Resampling for Finding Robust Optima

Procedure parameters: confidence level θ, averaging increment factor α

Procedure variables: sample size indicator m_{eval}, initialized at $m_{\text{eval}} = 2$

1. For all candidate solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ obtain robustness estimates $\hat{f}_{\text{exp}1,\text{old}}, \ldots, \hat{f}_{\text{exp}\lambda,\text{old}}$ based on a MEM evaluation scheme using $m_1 = \lceil m_{\text{eval}}/2 \rceil$ perturbations for each individual

$$L^{\text{old}} = \{\hat{f}_{\text{exp}1,\text{old}}, \ldots, \hat{f}_{\text{exp}\lambda,\text{old}}\}. \tag{7.19}$$

2. Repeat step 1 using $m_2 = \lfloor m_{\text{eval}}/2 \rfloor$ perturbations for each individual and store the mean objective function values in the set $L^{\text{new}} = \{\hat{f}_{\text{exp}1,\text{new}}, \ldots, \hat{f}_{\text{exp}\lambda,\text{new}}\}$.

3. Compute the rank-changes $\Delta_1, \ldots, \Delta_\lambda$ using

$$\Delta_i = \text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}}) - \text{signum}\left(\text{rank}(L_i^{\text{new}}) - \text{rank}(L_i^{\text{old}})\right). \tag{7.20}$$

4. Compute the uncertainty level based on the rank-changes

$$\begin{aligned} s \;=\; & \frac{1}{\lambda_{\text{reev}}} \sum_{i=1}^{\lambda_{\text{reev}}} (2|\Delta_i| \\ & -\Delta_\theta^{\lim}\left(\text{rank}\left(L_i^{\text{new}}\right) - \mathbb{I}\{L_i^{\text{new}} > L_i^{\text{old}}\}\right) \\ & -\Delta_\theta^{\lim}\left(\text{rank}\left(L_i^{\text{old}}\right) - \mathbb{I}\{L_i^{\text{old}} > L_i^{\text{new}}\}\right)), \end{aligned} \tag{7.21}$$

5. Update the sample size m_{eval} using the update rule

$$m_{\text{eval}} = \begin{cases} \alpha \cdot m_{\text{eval}} & , \text{if } s > 0 \\ m_{\text{eval}} & , \text{otherwise} \end{cases}. \tag{7.22}$$

6. Generate a ranking $\mathbf{x}_{1:\lambda}, \ldots, \mathbf{x}_{\lambda:\lambda}$ based on the combined fitness approximations

$$\hat{f}_{\text{exp}i} = \frac{\hat{f}_{\text{exp}1,\text{old}} + \hat{f}_{\text{exp}1,\text{new}}}{2}, \; i = 1, \ldots, \lambda. \tag{7.23}$$

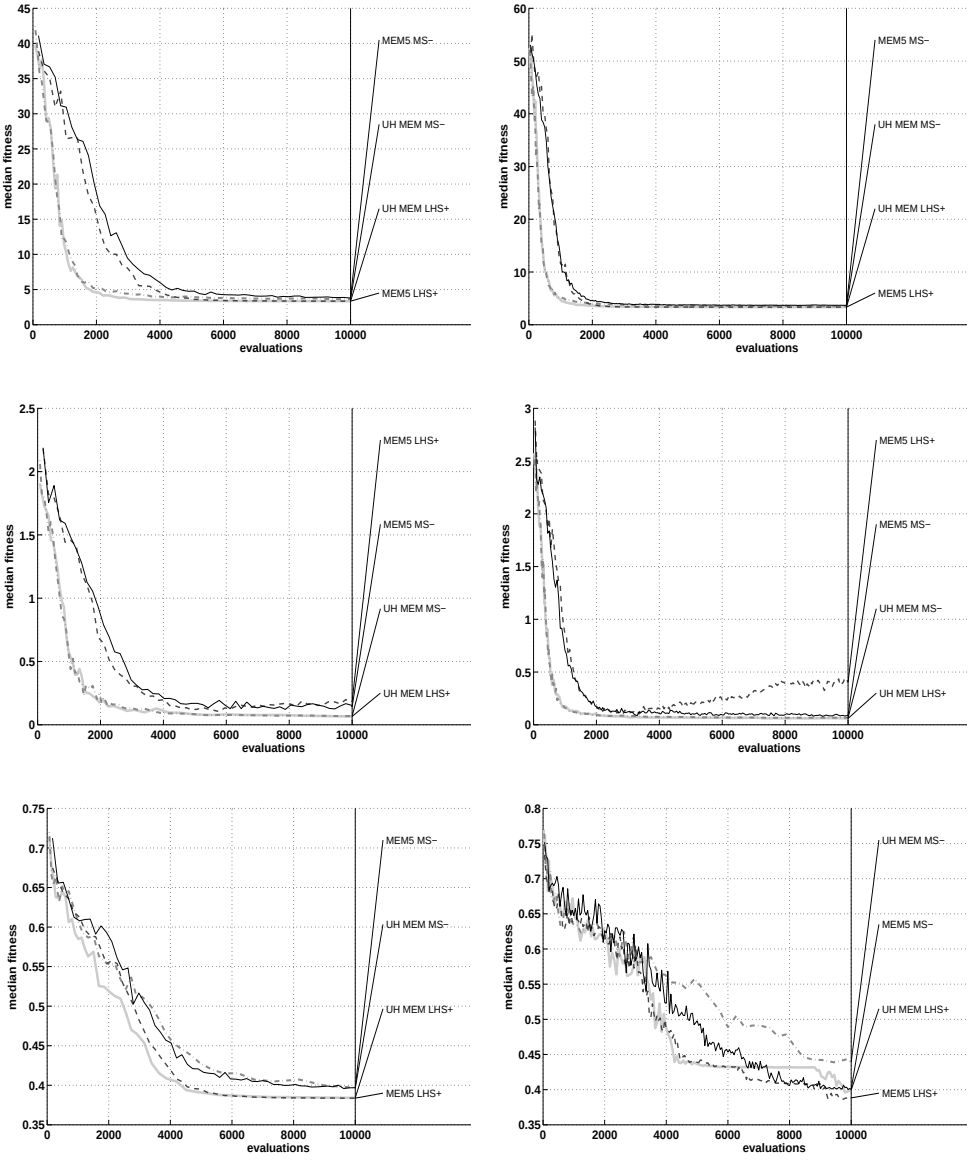


Figure 7.15: Results of Experiment 7.2.7. The performance, in terms of median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples) comparing static explicit averaging versus adaptive averaging for finding robust optima. Top row: results on the Heaviside sphere. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

7.2.6 Mutation as Robustness Tester

Beyer and Sendhoff [BS06a] also propose an adaptation of Evolution Strategies that can be classified as an adaptive averaging approach. However, it also introduces an alternative way of sampling for finding robust optima.

The focus in [BS06a] lies on optimization of the expected objective function with Gaussian input perturbations and they consider the SEM^- evaluation approach in the context of a $(\mu/\mu_I, \lambda)$-ES. For generating offspring, they suggest to include mutation itself as a robustness tester. That is, instead of generating an offspring $\mathbf{x}_\mathrm{o} = \mathbf{x}_\mathrm{p} + \mathbf{z}_\mathrm{o}$ and evaluating on $\mathbf{x}_\mathrm{o} + \mathbf{z}_\epsilon$, $\mathbf{z}_\epsilon \sim \mathrm{pdf}(\boldsymbol{\delta})$, they propose to interpret $\mathbf{z}_\epsilon + \mathbf{z}_\mathrm{o}$ as a mutation in its own right. In addition to this, they observe that selection tends to favor solutions with smaller disturbance vectors lengths $|\mathbf{z}_\epsilon|$. For this, they propose a control mechanism for adding input perturbations such that the variance amongst the μ selected parents is equal to the desired input perturbations. A third modification with respect to canonical Evolution Strategies is the inclusion of an adaptive averaging scheme. In order to prevent stagnation they propose an adaptive averaging technique that increases the population size when necessary (i.e., implicit averaging).

Technical Note 7.2 describes the mutation operator and the control mechanism for assuring that the measured disturbances of the μ selected offspring are distributed according to the targeted noise distribution $\mathrm{pdf}(\boldsymbol{\delta})$. As mentioned, the approach is including the input disturbances in the mutation operator, measuring the realized perturbation lengths of the selected individuals, and scaling the perturbation magnitude such that the realized perturbation length will approximate the desired input pertrubation length. Technical Note 7.3 describes the adaptive averaging approach suggested in [BS06a], which is based on increasing the population size with a certain factor whenever the uncertainty indicator $\overline{\Delta F}$ is below 0, which is checked every $\Delta g = n$ generations. The uncertainty indicator is based on tracking the improvement/decrement of the average population fitness of the select individuals.

The approach suggested in [BS06a] showed promising results on a number of test problems as compared to myopic approaches. The results were obtained using a $(\mu/\mu_I, \lambda)$-ES. For integration of this concept into a CMA-ES, it should take into account the full covariance matrix in the control mechanism for assuring that the measured disturbances of the selected offspring are equal to the targeted disturbances. Furthermore, note that the proposed scheme is specifically designed for SEM^- approaches. When using a SEM^+ or MEM^+, the observation of selection favoring smaller disturbance lengths does not hold anymore, because the disturbances are reused for all individuals in the population. One could think of using these approaches as an alternative fix to prevent selection from having this bias.

Technical Note 7.2: Mutation as Robustness Tester

Consider input perturbations $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathbf{D})$, with $\mathbf{D} = diag((\sigma_\epsilon)_1^2, \ldots, (\sigma_\epsilon)_n^2)$. The mutation operator of Evolution Strategies can be adapted as

$$x_i = \langle x \rangle_i + \sqrt{\sigma_i^2 + \epsilon_i^2} \cdot \mathcal{N}(0,1)\ ,\ i = 1, \ldots, n, \tag{7.24}$$

where $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_n]$ is not equivalent to the desired/targeted input noise $\boldsymbol{\sigma}_\epsilon = [(\sigma_\epsilon)_1, \ldots, (\sigma_\epsilon)_n]$, but is controlled such that the observed standard deviations $\mathbf{d} = [d_1, \ldots, d_n]$ of the μ selected individuals are close to $\boldsymbol{\sigma}_\epsilon$. This is accomplished by using the following update rule after every generation:

$$\epsilon_i = \epsilon_i \exp(\tau_\epsilon \cdot \text{sign}((\sigma_\epsilon)_i - d_i)), \tag{7.25}$$

where $\tau_\epsilon = c_x/3$ is a damping constant and $c_x = n^{-1}$ is a cumulation rate.

The observed standard deviations $\mathbf{d} = [d_1, \ldots, d_n]$ are measured over the main axes of μ selected individuals, which in this case are computed using a cumulated version of $d_i = \sqrt{\text{Var}[\{(\mathbf{x}_{1:\lambda})_i, \ldots, (\mathbf{x}_{\mu:\lambda})_i\}]}$:

$$d_i = \sqrt{\overline{x_i^2} - \overline{x}_i^2}, i = 1, \ldots, n, \tag{7.26}$$

with

$$\overline{x}_i = (1 - c_x)\overline{x}_i + c_x \langle x_i \rangle, \tag{7.27}$$

$$\overline{x_i^2} = (1 - c_x)\overline{x_i^2} + c_x \langle x_i^2 \rangle, \tag{7.28}$$

$$\langle x_i \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} (\mathbf{x}_{m:\lambda})_i, \tag{7.29}$$

$$\langle x_i^2 \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} (\mathbf{x}_{m:\lambda})_i^2. \tag{7.30}$$

Technical Note 7.3: Adaptive Population Size Control

As an uncertainty indicator, the average parental fitness change ΔF is given by

$$\Delta F = \langle F \rangle^{(g)} - \langle F \rangle^{(g-1)}, \tag{7.31}$$

where $\langle F \rangle^{(g)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \hat{f}_{\text{eff}\,i:\lambda}$ for the current generation g. This indicator is smoothened as

$$\overline{\Delta F} = (1 - c_f)\overline{\Delta F} + c_f(\langle F \rangle^{(g)} - \langle F \rangle^{(g-1)}). \tag{7.32}$$

This indicator is checked every $\Delta g = n$ generations and when $\overline{\Delta F} \leq 0$, then the uncertainty treatment mechanism should be applied:

$$\mu = \lceil \alpha\mu \rceil, \tag{7.33}$$
$$\lambda = \lceil \mu/\eta \rceil, \tag{7.34}$$

with $\eta = \mu_0/\lambda_0$.

7.2.7 Archiving for Finding Robust Optima

Using resampling methods for approximating the robustness of candidate solutions can become computationally expensive. In an attempt to reduce the number of objective function evaluations, Branke [Bra98] suggested to evaluate each individual based on its own fitness and the fitness of the other individuals in the neighborhood. More specifically, the expected fitness of each individual $\mathbf{x}$ is approximated as the weighted mean of its own fitness and the fitness of previously evaluated neighboring points $\mathbf{x}'$ using

$$\hat{f}_{\text{exp}}(\mathbf{x}) = \frac{\sum_{\mathbf{x}'} w(\mathbf{x}' - \mathbf{x}) \cdot f(\mathbf{x}')}{\sum_{\mathbf{x}'} w(\mathbf{x}' - \mathbf{x})}, \tag{7.35}$$

where $w(\mathbf{x}'-\mathbf{x})$ is a weight function for which it should hold that $w(\mathbf{x}'-\mathbf{x}) \propto \text{pdf}_{\boldsymbol{\delta}}(\mathbf{x}'-\mathbf{x})$ (cf. Eq. 7.18). Although computing the distances between each pair $(\mathbf{x}, \mathbf{x}')$ introduces an overhead cost, in many cases it is fair to assume that the cost of evaluating one candidate solution is larger than this extra overhead cost. In [KEB10], this idea was extended by means of a method to obtain a more representative archive, which is also used in [SRS11].

A way in which such a scheme can be incorporated into an Evolutionary Algorithm is depicted in Technical Note 7.4. It is similar to a canonical evolution cycle, only a few extra steps are included involving the evaluation of candidate solutions. Note that this framework differs slightly from the description provided in [KEB10].

The evaluation procedure knows two stages. The first stage concerns updating the archive. An optional step is to evaluate each individual precisely. Also, it is checked whether the archive is representative enough to be used for obtaining approximations of the expected fitness. If not, then additional samples are chosen and added to the archive. The approach of Branke [Bra98] omits the latter step, but evaluates each individual precisely. The approach presented

in [KEB10] includes a method for checking the archive and selecting appropriate additional sample points which will be described below. In the second stage, appropriate archive points are selected for each individual and based on that approximations of the expected fitness are obtained following Eq. 7.35. An optional operation after each generation is to include a method to clean up the archive and prevent it to grow out of bounds.

The separation between the archive maintenance and evaluation (which is a difference with respect to the approach presented in [KEB10]) is to be recommended because during the first stage of this algorithm, the archive will be filled with samples that could be relevant for all individuals. Hence, in order to allow all individuals to benefit from the additional samples taken in the first stage, the evaluation stage is decoupled from the first stage. This should prevent the selection from being biased due to the order in which the individuals are handled.

A difference between the approach presented in [Bra98] and the alternative approach presented in [KEB10] is the way in which additional archive points are selected. In [Bra98] this is based on precise evaluation of the candidate solutions in the population, however, a major drawback of this is that the points in the archive are by no means guaranteed to be well-spread over the region of possible variation for each candidate solution. That is, the archive points are selected by the optimization algorithm, which yields a distribution of points that is dependent on the way in which subsequent candidate solutions are selected. Especially in focused searching strategies such as Evolution Strategies, this might lead to an archive that holds limited information about the (local) objective function landscape. In [KEDB10b], an approach is suggested to counter this drawback. The general idea of this approach is to generate a reference sample set of disturbances that should represent an ideal sample set. For each individual, additional sampling is determined by the reference set point that is least represented in the archive. Hence, additional sampling is done in the parts of the regions of uncertainty that are not underrepresented in the archive. By using a Latin Hypercube Sampling reference set, it can be assumed that the reference set is space filling and clustering is avoided.

Algorithm 7.2 describes the archive based selection method as proposed in [KEDB10b]. It takes as arguments a reference sample set $\mathbf{X}_{\mathrm{ref}} = \{\mathbf{x}_{\mathrm{r}}^{(1)}, \ldots, \mathbf{x}_{\mathrm{r}}^{(m)}\}$ of m samples, which should be the LHS reference set for an individual for which a robustness estimate is required, and the archive $\mathbf{A} = \{(\mathbf{x}_{\mathrm{a}}^{(1)}, f_{\mathrm{a}}^{(1)}), (\mathbf{x}_{\mathrm{a}}^{(2)}, f_{\mathrm{a}}^{(2)}), \ldots\}$ which contains design point / objective function value tuples.

For each reference sample $\mathbf{x}_{\mathrm{r}}$ in $\mathbf{X}_{\mathrm{ref}}$ the algorithm searches for the closest point in the archive. Then, for each of these selected archive points, it is checked whether the reference points for which they are selected are also the closest reference points. The archive points for which this is the case are then, together with their objective function values, added to the set $\mathbf{A}_{\mathrm{sel}}$ of selected archive points. For the archive points for which this is not the case one can conclude that the region around its reference point is underrepresented in the archive (making this reference point a good candidate for extra sampling). It is therefore added to the set $\mathbf{X}_{\mathrm{cand}}$

Technical Note 7.4: General Framework of an Archive Based Evolutionary Algorithm for Finding Robust Solutions

initialize parent population and archive
while not terminate **do**
  generate offspring from parents
  for each individual **do**
    (**optional: evaluate** individual and **add** to archive)
    if not enough samples **or** no appropriate sample set available **then**
      find appropriate additional sample points
      evaluate additional sample points and **add** to archive
    end if
  end for
  for each individual **do**
    select archive points for effective fitness approximation
    compute effective fitness approximation using the archive
  end for
  select best individual
  (**optional: clean up** archive)
end while

Algorithm 7.2: Reference Set Based Archive Selection for Finding Robust Optima

input: reference sample set $\mathbf{X}_{\mathrm{ref}}$, archive $\mathbf{A}$

output: selected archive points $\mathbf{A}_{\mathrm{sel}}$, sampling candidates $\mathbf{X}_{\mathrm{cand}}$

1: $\mathbf{A}_{\mathrm{sel}} \leftarrow \emptyset$
2: $\mathbf{X}_{\mathrm{cand}} \leftarrow \emptyset$
3: **for each** $\mathbf{x}_{\mathrm{r}} \in \mathbf{X}_{\mathrm{ref}}$ **do**
4: $\quad (\mathbf{x}_{\mathrm{a}}, f_{\mathrm{a}}) \leftarrow \{(\mathbf{x}_{\mathrm{a}}, f_{\mathrm{a}}) \in \mathbf{A} \mid \text{for all } (\mathbf{x}'_{\mathrm{a}}, f'_{\mathrm{a}}) \in \mathbf{A} : d(\mathbf{x}_{\mathrm{r}}, \mathbf{x}_{\mathrm{a}}) \leq d(\mathbf{x}_{\mathrm{r}}, \mathbf{x}'_{\mathrm{a}})\}$
5: $\quad$ **if** $(\text{for all } \mathbf{x}'_{\mathrm{r}} \in \mathbf{X}_{\mathrm{ref}} \ : \ d(\mathbf{x}_{\mathrm{r}}, \mathbf{x}_{\mathrm{a}}) \leq d(\mathbf{x}'_{\mathrm{r}}, \mathbf{x}_{\mathrm{a}}))$ **then**
6: $\quad\quad \mathbf{A}_{\mathrm{sel}} \leftarrow \mathbf{A}_{\mathrm{sel}} \cup \{(\mathbf{x}_{\mathrm{a}}, f_{\mathrm{a}})\}$
7: $\quad$ **else**
8: $\quad\quad \mathbf{X}_{\mathrm{cand}} \leftarrow \mathbf{X}_{\mathrm{cand}} \cup \{\mathbf{x}_{\mathrm{r}}\}$
9: $\quad$ **end if**
10: **end for**
11: **return** $(\mathbf{A}_{\mathrm{sel}}, \mathbf{X}_{\mathrm{cand}})$

of candidate sample points. In case that all reference points have been assigned archive points, then the reference point of the archive point / reference point pair that are located farthest apart from each other is selected as candidate for extra sampling[1]. This will assure that the archive is always updated in the region in which it needs the extra samples the most.

Figure 7.16 illustrates the scheme: Step **a)** displays archive points using the $\bullet$-symbol, the reference samples with the $\otimes$-symbol, and the box represents the η-neighborhood of the solution to be evaluated. In step **b)** for each reference point, the closest archive point is identified. In step **c)** it is checked whether the reference points are also the closest reference points of their selected archive points and in step **d)** the archive points for which this is the case are selected as archive points (solid circles) and the reference points for which this is not the case are selected as candidates for additional sampling (dashed circles).

The framework provided in Technical Note 7.4 combined with the archive selection scheme of Algorithm 7.2 yields an evaluation scheme for finding robust optima that can be integrated within any type of Evolutionary Algorithm. In this work, we adopt a slightly modified version of the scheme as considered in [KEDB10b]. It is presented in Algorithm 7.3. For

[1] This step is not shown in the algorithmic description of Algorithm 7.2.

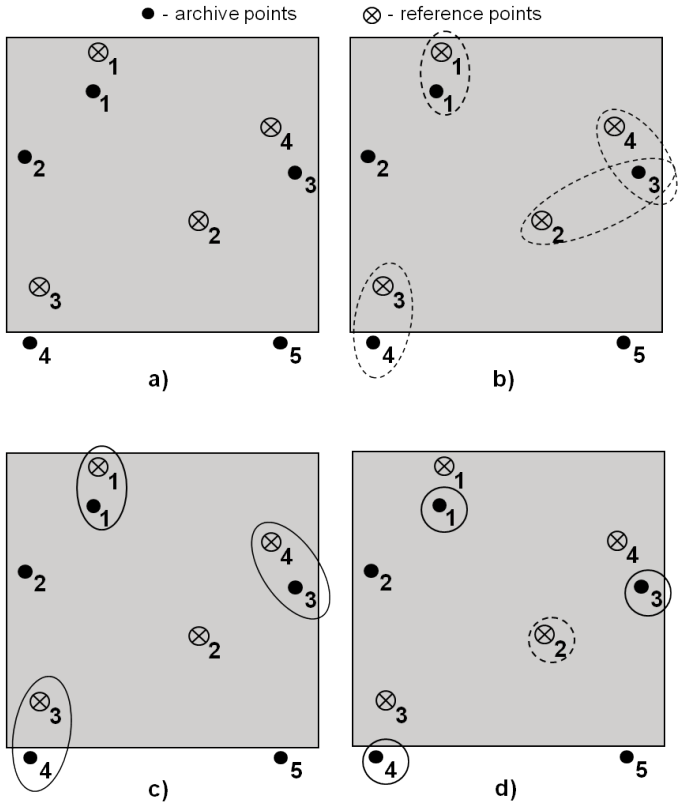


Figure 7.16: An illustration of the reference set based archive selection method proposed in [KEDB10b]. Step **a)**: the archive points are displayed with the ●-symbol, the reference samples with the ⊗-symbol, and the box represents the η-neighborhood of the to be evaluated solution. Step **b)**: for each reference point, the closest archive point is identified. Step **c)**: check whether the reference points are also the closest reference points of their selected archive points. Step **d)**: the archive points for which this is the case are selected as archive points (solid circles) and the reference points for which this is not the case are selected as candidates for additional sampling (dashed circles).

each individual, a reference set $\mathbf{X}_{\mathrm{ref}}$ of m samples is used to obtain a set of selected archive points $\mathbf{A}_{\mathrm{sel}} = \{(\mathbf{x}_{\mathrm{s}}^{(1)}, f_{\mathrm{s}}^{(1)}), (\mathbf{x}_{\mathrm{s}}^{(2)}, f_{\mathrm{s}}^{(2)}), \ldots\}$ and a set $\mathbf{X}_{\mathrm{cand}} = \{\mathbf{x}_{\mathrm{c}}^{(1)}, \mathbf{x}_{\mathrm{c}}^{(2)}, \ldots\}$ of suggested candidates for extra sampling. Then, in this variant, instead of all, only one of the suggested candidate points for extra sampling is evaluated and added to $\mathbf{A}$. An expected fitness approximation is thereafter generated by considering all archive points and using the weighted function as in Eq. 7.35.

Note that it is also possible to take for each individual a sample set using the reference set based archive selection method. However, it should be noted that this set should be selected anew in order to allow all individuals to use the same archive for evaluation. Using the complete archive for evaluation is justifiable when assuming that the archive will be locally well-spread (because additional samples will be taken in the least represented areas). An additional advantage of this is allows to take arbitrarily many samples for the evaluation (in the sense of adaptive averaging) and only requires a setting for m for the maintenance of the archive.

In [KEB10] an example of the behavior of this way of maintaining an archive is given by showing the archive development on a two-dimensional version of the Heaviside sphere (see Appendix B.2). Here, the archive based reference set selection scheme (named *ABRSS*) is compared against the archiving method as considered in [Bra98] (named *PROX*), both incorporated into a CMA-ES (see Section 4.2.3). Figure 7.17 shows, for a run of both the ABRSS scheme and the PROX scheme, the archive after 100, 200, and 300 generations respectively. Here, the small plots inside each plot are magnified versions on the interval $[0, 2]^2$ (i.e., the interval around the optimum). The asset of the archive maintenance scheme can be seen clearly; whereas the PROX method zooms in on a narrow region, the ABRSS scheme takes into account the whole region of uncertainty around the point on which it zooms in, leading to a well-spread set of archive points around the optimum (in effect generating more accurate fitness approximations). Although in higher dimensions it will take more time to build up a well-spread archive, the ABRSS scheme, in contrast to the PROX scheme, has the potential to do so.

Finally, in order to get an impression of the performance of this archive maintenance scheme on 10-dimensional problems, we consider the following experiment:

Experiment 7.2.8 (Performance of archive based evaluation for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using the archive based reference set approach for assessment of the expected fitness as described in Algorithm 7.3. The experiments are performed on the sphere problem (see Appendix B.1), the Heaviside sphere problem (see Appendix B.2), and on Branke's multipeak problem (see Appendix B.6). For the archive based reference set selection, a sample size of $m = 2n = 20$ is used. Each run uses a budget of $10,000$ function evaluations.

Algorithm 7.3: Archive Maintenance in Evolutionary Algorithms for Finding Robust Optima

Procedure parameters: the number of samples used for generating the reference set m

Procedure variables: solution archive A

1: $\mathbf{P} \leftarrow$ *generate_initial_population*()
2: $A \leftarrow \emptyset$
3: **while** not terminate **do**
4: $\quad\mathbf{O} \leftarrow$ *generate_offspring*($\mathbf{P}$)
5: $\quad$**for each** $\mathbf{x} \in \mathbf{O}$ **do**
6: $\quad\quad\mathbf{X}_{\mathrm{ref}} \leftarrow$ *latin_hypercube_sampling*($\mathbf{x}, \boldsymbol{\sigma}_\epsilon, m$)
7: $\quad\quad(\mathbf{A}_{\mathrm{sel}}, \mathbf{X}_{\mathrm{cand}}) \leftarrow$ *reference_set_based_archive_selection*($\mathbf{X}_{\mathrm{ref}}, \mathbf{A}$)
8: $\quad\quad$**for a random** $\mathbf{x}_c \in \mathbf{X}_{\mathrm{cand}}$ **do**
9: $\quad\quad\quad f_{\mathrm{c}} \leftarrow$ *evaluate*($\mathbf{x}_{\mathrm{c}}$)
10: $\quad\quad\quad A \leftarrow A \cup \{(\mathbf{x}_{\mathrm{c}}, f_{\mathrm{c}})\}$
11: $\quad\quad$**end for**
12: $\quad$**end for**
13: $\quad$**for each** $\mathbf{x} \in \mathbf{O}$ **do**
14: $\quad\quad\hat{f}_{\mathrm{exp}}(\mathbf{x}) \leftarrow (\sum_{\{\mathbf{x}_{\mathrm{a}}, f_{\mathrm{a}}\} \in \mathbf{A}} w(\mathbf{x}_{\mathrm{a}} - \mathbf{x}) \cdot f_{\mathrm{a}}) / (\sum_{\{\mathbf{x}_{\mathrm{a}}, f_{\mathrm{a}}\} \in \mathbf{A}} w(\mathbf{x}_{\mathrm{a}} - \mathbf{x}))$
15: $\quad$**end for**
16: $\quad\mathbf{P} \leftarrow$ *select*($\mathbf{O}$)
17: **end while**

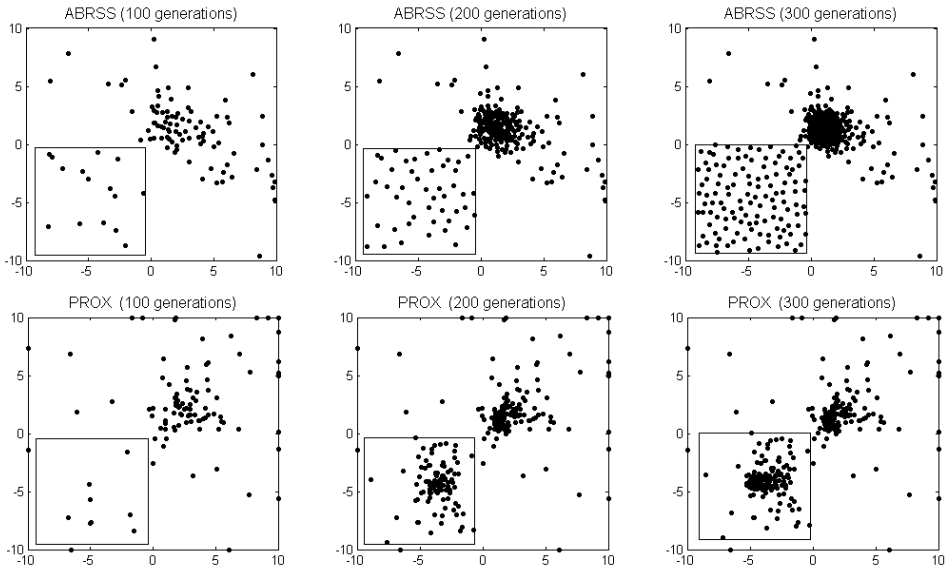


Figure 7.17: Archive after 100, 200, and 300 generations of the ABRSS scheme (top row) and PROX scheme (bottom row) on a two-dimensional instance of the heaveyside sphere. The zoom window magnifies the interval $[0, 2]^2$.

Figure 7.18 and Figure 7.19 show the results of Experiment 7.2.8. In Figure 7.18 the performance is shown in terms of median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples) and in Figure 7.19 the final solution quality (approximated a posteriori using Monte-Carlo sampling with 1000 samples) is displayed. From the figures it can be seen that the methods incorporating the ABRSS scheme outperforms the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ scheme on all runs. Comparing it, however, to the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ scheme, it is outperformed on the sphere problem, and on Branke's multipeak problem, but is clearly better on the Heaviside sphere. The results suggest that the ABRSS is indeed capable on zooming in on the robust optimum, but for 10-dimensional problem spaces, it might take some while for the archive to fill. When the objective function landscape is symmetric around the (robust) optimum, the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$, that exploits such symmetry, will be able to zoom in on the robust optimum more quickly. The convergence speed obtained by using the ABRSS scheme (see Figure 7.19) is much higher than when using the MEM schemes.

In conclusion, using an archive of previously evaluated solutions provides a way for efficiently using objective function evaluations. By using the archive maintenance approach as proposed in [KEB10], it can be assured that the archive will be updated in the places where it is underrepresented. This approach is suitable especially for low-dimensional search spaces (dimension $\lesssim 10$). The higher the dimensionality of the search space, the longer it will take before the archive is filled sufficiently well to be representative. Regarding the computational

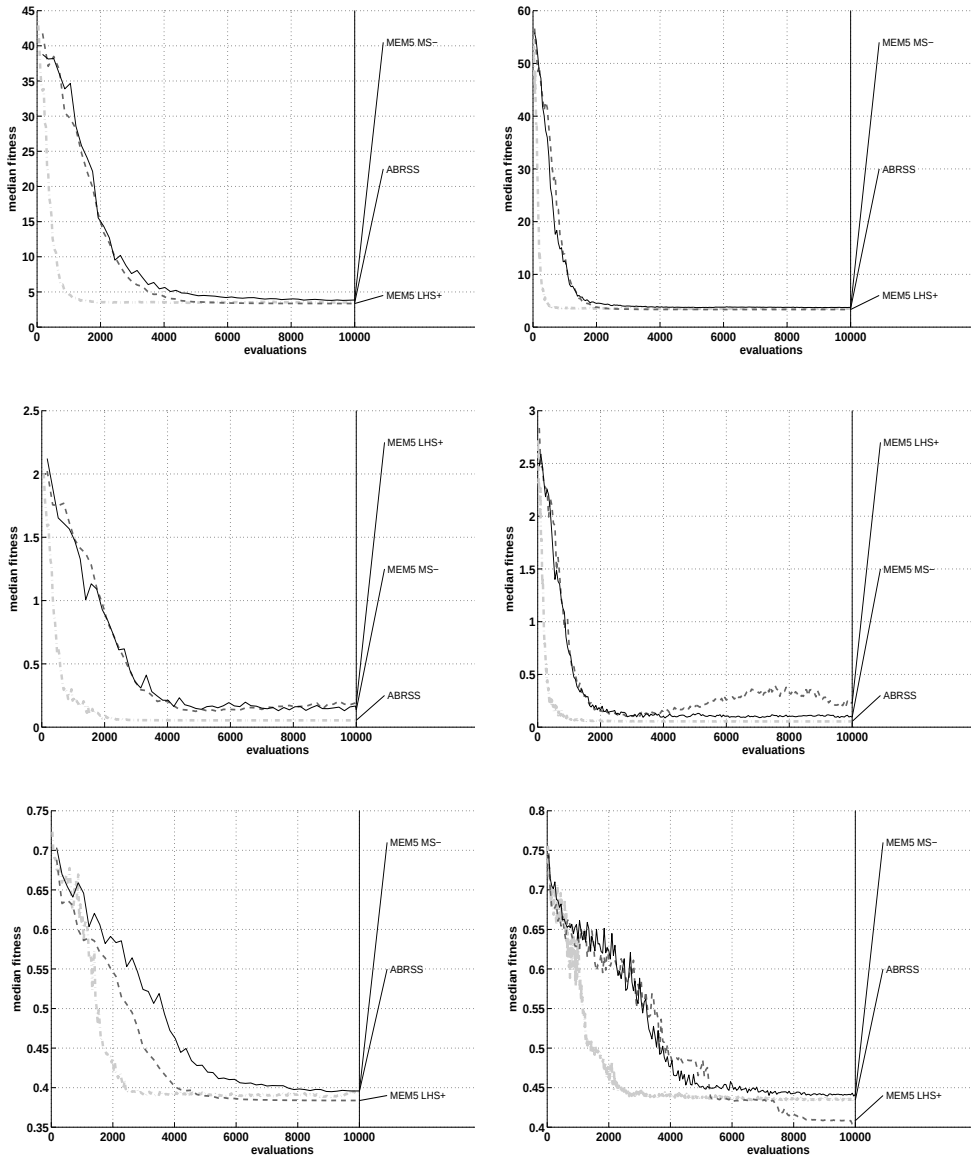


Figure 7.18: Results of Experiment 7.2.8. The performance, in terms of median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples) of the ABRSS evaluation scheme incorporated into the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, compared against two variants of the MEM evaluation schemes. Top row: results on the sphere. Middle row: results on the Heaviside sphere. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

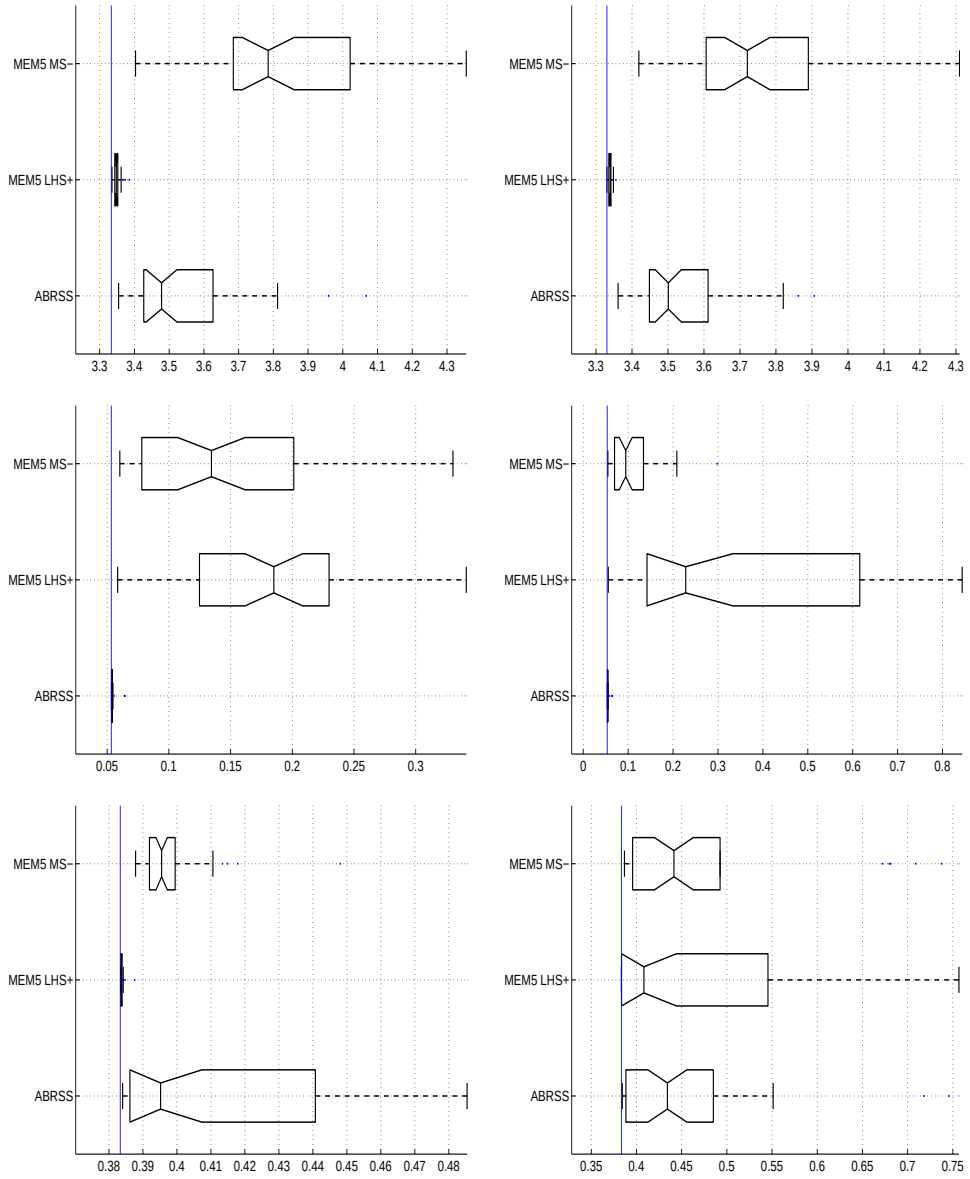


Figure 7.19: Results of Experiment 7.2.8. The final solution quality of the ABRSS evaluation scheme incorporated into the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, compared against two variants of the MEM evaluation schemes. The solution quality is approximated a posteriori using Monte-Carlo sampling with 1000 samples. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere problem. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

cost of using the archive maintenance approach as considered in this section, it should be noted that a serious additional overhead is introduced by following this method. It is therefore specifically useful when the (computational) cost of an objective function evaluation is high.

7.2.8 Metamodeling for Finding Robust Optima

Metamodeling approaches are closely related to archive based approaches and are also particularly useful when objective function evaluations are expensive. The general idea is to construct (local) approximation models (or metamodels) of the objective function for which evaluation is cheap and obtain accurate robustness approximations based on these metamodels. In the context of classical optimization using Evolutionary Algorithms, metamodeling techniques know many applications (for an overview, see, e.g., [Jin05]) and it is also applied in the context of noisy optimization (see Section 5.5). In the context of finding robust optima, for which the demand for objective function evaluations is high, the application possibilities are apparent. Approaches that are proposed in this context are, e.g., [ONL06, PBJ06, PL09, KEDB10a].

The way in which metamodeling techniques can be integrated into a scheme for finding robust optima depends on a number of matters, i.e.,

- the modeling assumptions and modeling approach used for constructing the metamodels,
- the way in which archive points are selected for constructing metamodels,
- the frequency of the model updates / regenerations and the usage of the metamodels,
- the robustness measure and approximation method used on the metamodels.

Studying and testing all possible configurations for these different issues is obviously infeasible. Regarding the modeling assumptions and the modeling approach, this depends largely on the problem at hand and the same holds for the robustness measure and the approximation method. As an indication of the variety of possibilities, Paenke et al. [PBJ06] consider linear interpolation, quadratic interpolation, linear regression, and quadratic regression, Ong et al. [ONL06] consider radial basis functions, and Poles and Locison [PL09] consider polynomial models. The way in which the archive is maintained and the frequency of the model updates are more general choices. For the former, choosing additional sample points is much related to the discussion of the Section 7.2.7. The quality of a metamodel depends largely on the available sample set, which should contain "sufficient" information for generating an accurate model (see Section 5.5.3 for a discussion). As an indication for the possibilities regarding the frequency of the model updates and the usage of the metamodels, four possibilities studied by Paenke et al. [PBJ06] are:

- **Singe model**: generate a metamodel for each individual separately.

- **Nearest model**: construct a metamodel for each individual separately, but use for each sample point the closest model to estimate its objective function value.
- **Ensemble**: construct a metamodel for each individual in the population and use the ensemble of metamodels of all individuals in a weighted way to obtain objective function value approximations.
- **Multiple models**: generate a separate metamodel for each sample point that is to be evaluated.

Technical Note 7.5 shows a (possible) general framework of how metamodeling techniques can be integrated into an Evolutionary Algorithm for finding robust optima. For each offspring, a suitable set of archive points is selected which can be used to construct a local metamodel. In case such a suitable set of archive points does not exist, additional samples are taken, evaluated on the original objective function and also added to the archive. Thereafter, the fitness is determined by obtaining a robustness approximation based on the local metamodel or (optional) on the ensemble of metamodels of all offspring.

Note that the framework of Technical Note 7.5 much resembles the framework shown in Technical Note 7.4 on page 165. In fact, it can be argued that the archiving method of Section 7.2.7 is a simple form of metamodeling.

In this section, we will restrict ourselves to one configuration of the framework of Technical Note 7.5. This configuration is similar to the one considered in [KRD$^+$11] and serves as an example case to represent the metamodeling based approaches. This approach uses ordinary Kriging as metamodeling technique, which is briefly summarized in Appendix C. For maintaining an archive, the archive maintenance method as presented in Section 7.2.7 is used. A separate model is constructed for each individual in the population (i.e., the single model is followed). The robustness measure that will be adopted is the expected fitness, approximated using a $\mathrm{MEM}^+_{\mathrm{LHS}}$ evaluation scheme. The implementation of this specific configuration is described in Algorithm 7.4.

Experiment 7.2.9 (Performance of Kriging based evaluation for finding robust optima)**:** We perform 50 runs of a $(5/2_{DI}, 35)$-σSA-ES (see Section 4.2.2) and a CMA-ES (see Section 4.2.3) using the Kriging based approach for assessment of the expected fitness as described in Algorithm 7.4. The experiments are performed on the sphere problem (see Appendix B.1), the Heaviside sphere problem (see Appendix B.2), and on Branke's multipeak problem (see Appendix B.6). For building the Kriging metamodels, a sample size of $n_{\mathrm{krig}} = 2n = 20$ is used, and for the approximation for the expected fitness, a $\mathrm{MEM}50^+_{\mathrm{LHS}}$ approach is followed. Each run uses a budget of $10,000$ function evaluations.

Figure 7.20 and Figure 7.21 show the results of Experiment 7.2.9. In Figure 7.20 the performance is shown in terms of median fitness (a posteriori approximated using Monte-Carlo

Technical Note 7.5: General Framework of a Metamodel Assisted Evolutionary Algorithm for Finding Robust Optima

initialize parent population
initialize archive
while not terminate **do**
  generate offspring
  for each offspring **do**
    (**optional: evaluate** offspring and **add** to archive)
    select archive points for metamodel construction
    if no representative set of samples available **then**
      get extra sample points
      evaluate the extra sample points
      add the extra sample points to the archive
      construct a local metamodel for the current individual
    end if
  end for
  for each offspring **do**
    evaluate robust fitness using the (ensemble of) local metamodel(s)
  end for
  select best offspring as new parent population
  (**optional: clean up** archive)
end while

Algorithm 7.4: Kriging Based Evolutionary Algorithm for Finding Robust Optima

Procedure parameters: number of samples used for metamodel construction n_{krig}, number of samples used for estimating the effective fitness m

Procedure variables: solution archive A

1: $\mathbf{P} \leftarrow$ *generate_initial_population*()
2: $A \leftarrow \emptyset$
3: **while** not terminate **do**
4: $\quad\mathbf{O} \leftarrow$ *generate_offspring*($\mathbf{P}$)
5: $\quad$**for each** $\mathbf{x}_{\mathrm{o}} \in \mathbf{O}$ **do**
6: $\quad\quad\mathbf{X}_{\mathrm{ref}} \leftarrow$ *latin_hypercube_sampling*($\mathbf{x}_{\mathrm{o}}, \boldsymbol{\sigma}_{\epsilon}, n_{\mathrm{krig}}$)
7: $\quad\quad(\mathbf{A}_{\mathrm{sel}}, \mathbf{X}_{\mathrm{cand}}) \leftarrow$ *reference_set_based_archive_selection*($\mathbf{X}_{\mathrm{ref}}, \mathbf{A}$)
8: $\quad\quad$**for a random** $\mathbf{x}_{\mathrm{c}} \in \mathbf{X}_{\mathrm{cand}}$ **do**
9: $\quad\quad\quad f_{\mathrm{c}} \leftarrow$ *evaluate*($\mathbf{x}_{\mathrm{c}}$)
10: $\quad\quad\quad A \leftarrow A \cup \{(\mathbf{x}_{\mathrm{c}}, f_{\mathrm{c}})\}$
11: $\quad\quad\quad A_{\mathrm{sel}} \leftarrow A_{\mathrm{sel}} \cup \{(\mathbf{x}_{\mathrm{c}}, f_{\mathrm{c}})\}$
12: $\quad\quad$**end for**
13: $\quad\quad\hat{f}_{\mathbf{x}_{\mathrm{o}}}(\mathbf{x}) \leftarrow$ *calibrate_kriging*($\mathbf{A}_{\mathrm{sel}}$)
14: $\quad\quad\tilde{f}_{\mathbf{x}_{\mathrm{o}}} \leftarrow \hat{f}_{\mathrm{eff}}(\hat{f}(\mathbf{x})_{\mathbf{x}_{\mathrm{o}}}, \mathbf{x}_{\mathrm{o}})$
15: $\quad$**end for**
16: $\quad\mathbf{P} \leftarrow$ *select*($\mathbf{O}$)
17: **end while**

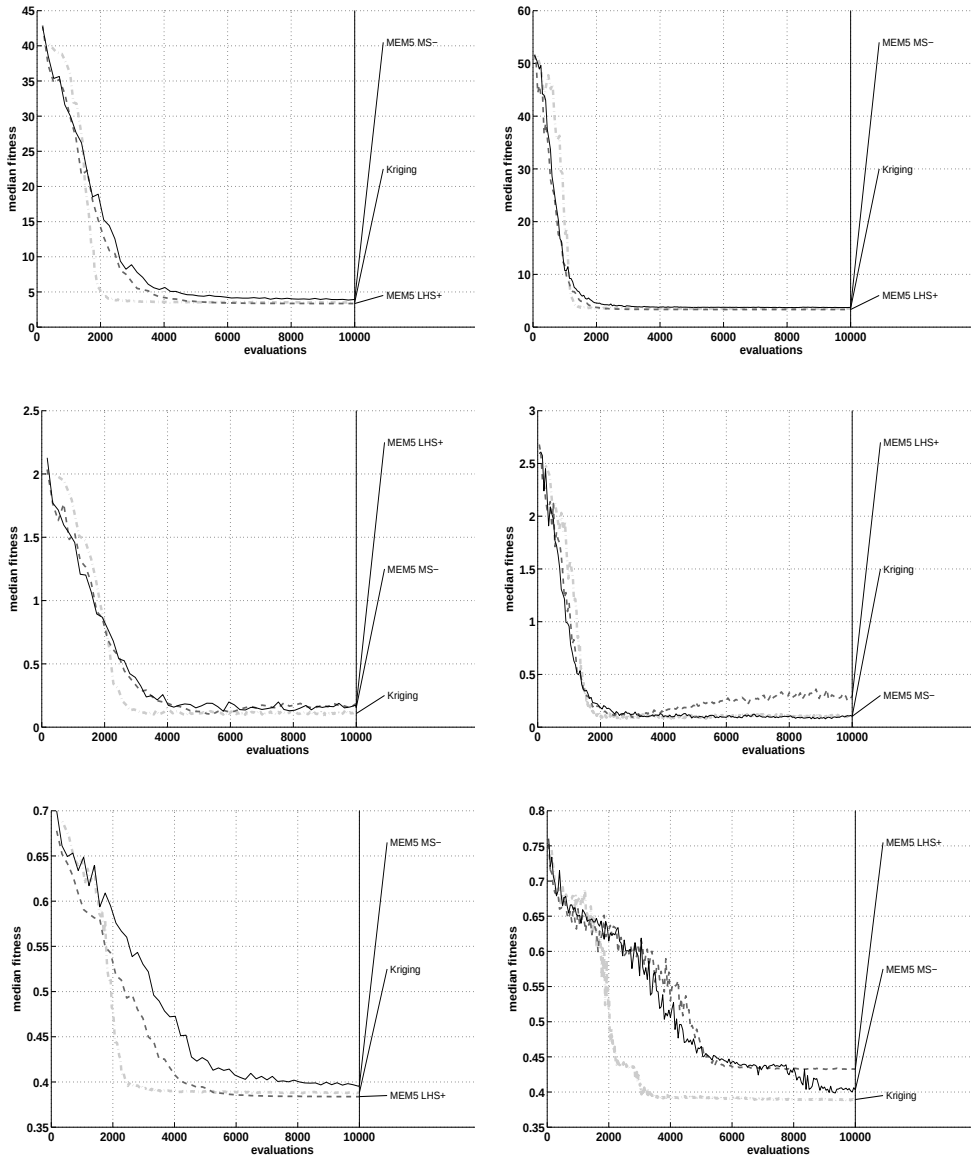


Figure 7.20: Results of Experiment 7.2.9. The performance, in terms of median fitness (a posteriori approximated using Monte-Carlo integration with 100 samples) of the Kriging based evaluation scheme incorporated into the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, compared against two variants of the MEM evaluation schemes. Top row: results on the sphere. Middle row: results on the Heaviside sphere. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

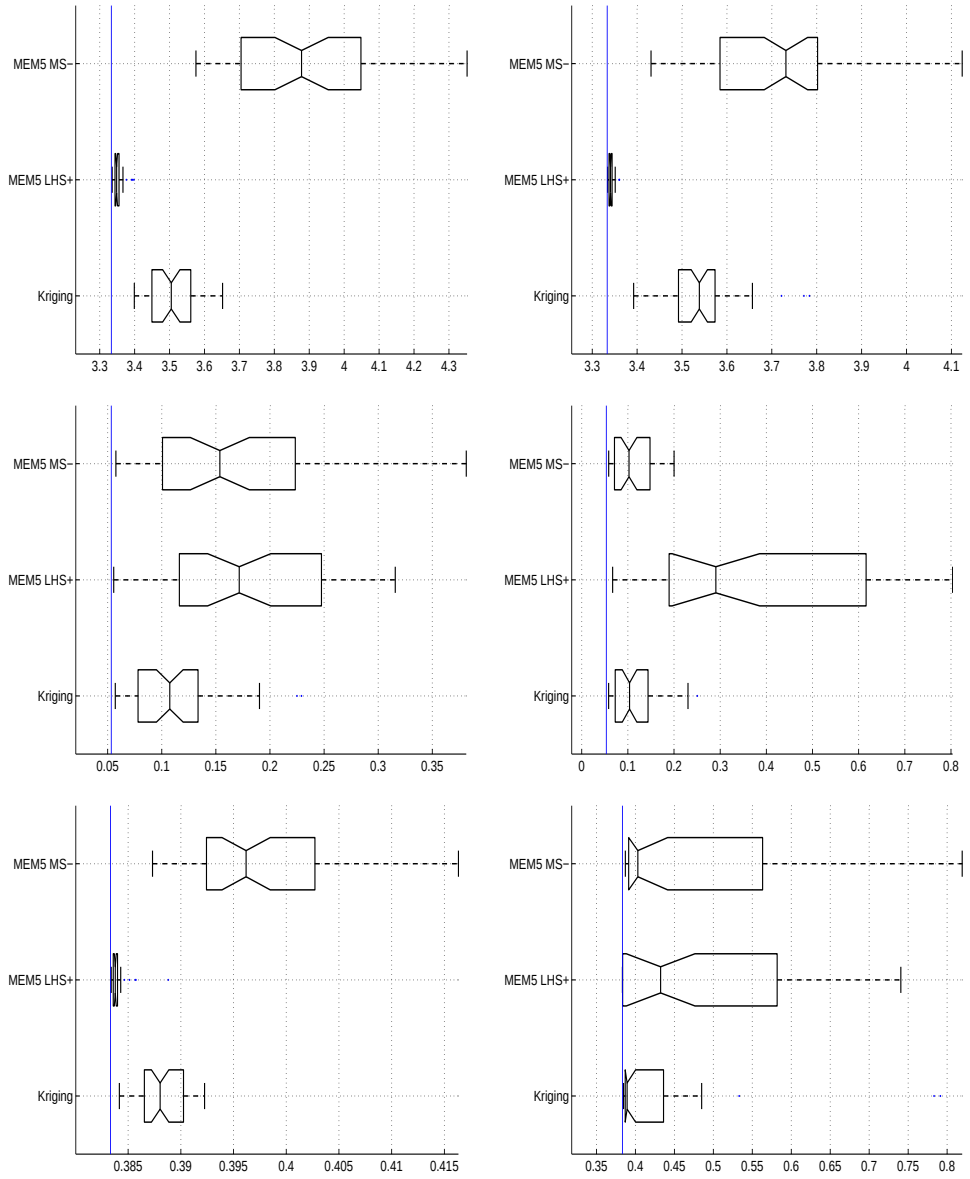


Figure 7.21: Results of Experiment 7.2.8. The final solution quality of the Kriging based evaluation scheme incorporated into the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, compared against two variants of the MEM evaluation schemes. The solution quality is approximated a posteriori using Monte-Carlo sampling with 1000 samples. Top row: the results on the sphere problem. Middle row: the results on the Heaviside sphere problem. Bottom row: the results on Branke's multipeak problem. Left column: the $(5/2_{DI}, 35)$-σSA-ES. Right column: the CMA-ES. The results are obtained using 50 runs for each scheme.

integration with 100 samples) and in Figure 7.21 the final solution quality (approximated a posteriori using Monte-Carlo sampling with 1000 samples) is displayed. From the figures it can be seen that we can draw similar conclusions as for the ABRSS approach of the previous section. The methods incorporating the Kriging scheme outperforms the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ scheme on all runs except for one where it is comparable. However, comparing it to the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ scheme, it is outperformed on the sphere problem and on Branke's multipeak problem, but is clearly better on the Heaviside sphere. The convergence speed obtained by using the Kriging based scheme (see Figure 7.21) is in the early stages a bit slower which is due to the fact that the archive needs to fill. Once the archive is filled, the Kriging based approach quickly zooms in and catches up with the MEM schemes.

In conclusion, we note that metamodeling can be successfully employed within Evolutionary Algorithms for finding robust optima. As the cost of constructing metamodels is high, such methods are only applicable if the evaluation cost of the original objective function is higher than the cost of constructing the metamodel. When this is the case, performing sampling on the metamodels is a promising way to get estimate for the effective fitness. A downside of metamodeling based approaches is that when it is too expensive to use all archive points for constructing the metamodels (e.g., as in Kriging), a subset selection should be made. This introduces an algorithm parameter that highly affects the accuracy of the metamodels. That is, when using a limited number of samples for metamodel construction, the metamodel itself is limited with respect to the prediction accuracy. In this sense, modeling approaches that use iterative updates are favorable, as those should be able to use the complete archive.

Additionally, an issue worthwhile mentioning is the possibility of performing exact robustness analysis on the metamodels instead of using sampling methods. That is, the metamodels themselves are not black-box functions, but have an explicit mathematical description. For some of these techniques, explicit derivation of the effective objective function measures might be possible. Poles and Lovison [PL09] already present such an approach in the context of polynomial models.

7.2.9 Niching for Finding Robust Optima

All techniques that have been discussed up to now have been noted to work well with respect to zooming in on the robust optimizer in case it is shifted, but do not noticeably improve the capabilities of Evolution Strategies to target the more robust peaks. This observation has been made, for instance, in [KEB10] in the context of archiving and in [KRD+11] in the context of adaptive averaging. Niching techniques are designed to improve the explorative behavior of Evolutionary Algorithms by actively separating sub-populations in different parts of the search space. For finding robust optima, such techniques could also be incorporated, with the specific goal in mind to find the more robust parts of the search space. Preliminary results, providing a proof of concept for this idea, are presented by Tsutsui and Ghosh [TG97]. In this study,

the sharing scheme of Goldberg and Richardson [GR87] is used in combination with a SEM evaluation approach. However, despite promising results, no further studies exist that include niching techniques for finding robust optima.

A straightforward incorporation of niching into a scheme for finding robust optima is, for instance, to use a SEM/MEM evaluation scheme and include a basic niching approach, e.g., as proposed by Shir et al. [SB09]. Algorithm 7.5 describes the general framework of this niching approach, in which the evaluation procedure should simply be a SEM/MEM approach when aiming to find robust optima.

In this framework, q niches are maintained, which are in the simplest case q parent individuals which are separated in the search space (i.e., the niche leaders or alpha individuals). Every generation λ offspring are generated for each niche separately. After that, the full population of offspring is evaluated, and based on the fitness, the dynamic peak set, which is the set of new niche leaders for the next generation, is selected by means of the dynamic peak identification procedure (see Algorithm 7.6). Each individual in the DPS forms a separate niche and inherits the strategy parameters from its parent, which are updated according to the normal update rules. Finally, if there are fewer than q niches, new niches are created randomly.

The dynamic peak identification algorithm, as described in Algorithm 7.6 works by sorting the individuals by fitness, and then considering them in that order (i.e., fitter individuals are treated earlier). The best individual is always a niche leader and will always form a niche. The other individuals will form a niche only if they are not within a radius ρ of an already formed niche and if there are not yet q niches. This way, a set of at most q individuals is selected, which are the niche leaders for the next generation.

Regarding the two important parameters ρ and q: the number of desired niches is a design choice and the niche radius is recommended to be set relative to the number of niches,

$$\rho = \frac{\sqrt{\sum_{i=1}^{n}((\mathbf{x}_\mathrm{u})_i - (\mathbf{x}_\mathrm{l})_i)^2}}{2\sqrt[n]{q}}. \tag{7.36}$$

Although the approach sounds reasonable, a simple experiment on a two-dimensional instance of Branke's multipeak problem shows that it is more complicated than this:

Experiment 7.2.10 (Dynamics of niching for finding robust optima)**:** We consider four two-dimensional instance of Branke's multipeak problem (see Appendix B.6) in which the anticipated input uncertainties vary. The noise levels are: $\boldsymbol{\delta} = \mathbf{0}$ (no noise), $\boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.05}, \mathbf{0.05})$ (low noise), $\boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.1}, \mathbf{0.1})$ (medium noise), $\boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.5}, \mathbf{0.5})$ (default noise). On these problems we run a normal CMA-ES (see Section 4.2.3) incorporating a SEM^- evaluation scheme and a niching based CMA-ES, which uses $q = 4$ niches, one parent per niche (i.e., $\mu = 1$), $\lambda = 10$ offspring per niche, and also the SEM^- evaluation scheme. For both schemes one run is performed on each problem instance.

Figure 7.22 shows the results of Experiment 7.2.10 in terms of distance to the robust optimizer

Algorithm 7.5: Niching Based Evolutionary Algorithm for Finding Robust Optima

Procedure parameters: Number of niches q, niching radius ρ

1: **initialize** parent population
2: **while** not terminate **do**
3: **for** each niche **do**
4: **generate** λ offspring from the current niche
5: **end for**
6: **evaluate** offspring
7: **compute** the Dynamic Peak Set (DPS) of the population
8: **for each** individual in the DPS **do**
9: **set** the current individual as niche
10: **inherit** the strategy parameters from the parent
11: **update** the strategy parameters of the current niche
12: **end for**
13: **if** $|\mathrm{DPS}| < q$ **then**
14: **generate** new niche from scratch
15: **end if**
16: **end while**

Algorithm 7.6: Dynamic Peak Identification (DPI)

Procedure parameters: Number of niches q, niching radius ρ

Sort the population according to fitness: $P = \{\mathbf{x}_{1:q\lambda}, \ldots, \mathbf{x}_{1:q\lambda}\}$
$i \leftarrow 1$
$numpeaks \leftarrow 0$
$DPS \leftarrow \emptyset$
while $numpeaks < q$ **and** $i \leq q \cdot \lambda$ **do**
  $is\_niche \leftarrow \text{true}$
  for each $\mathbf{x}_{\text{niche}} \in DPS$ **do**
    if $||\mathbf{x}_{\text{niche}} - \mathbf{x}_{i:q\lambda}|| < \rho$ **then**
      $is\_niche \leftarrow \text{false}$
      break
    end if
  end for
  if is_niche **then**
    $DPS \leftarrow DPS \cup \{\mathbf{x}_{i:q\lambda}\}$
    $numpeaks \leftarrow numpeaks + 1$
  end if
  $i \leftarrow i + 1$
end while
return DPS

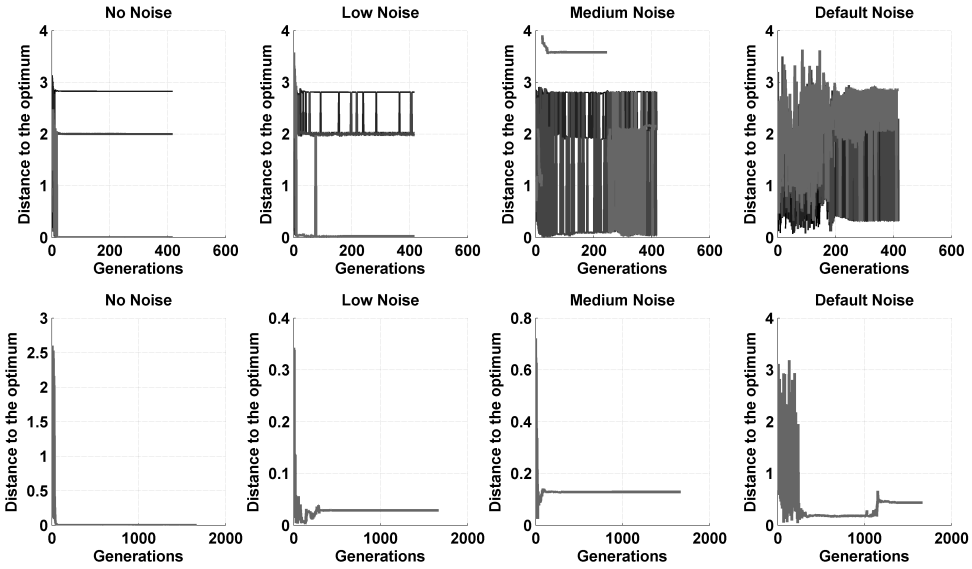


Figure 7.22: A single run of the niching approach incorporating a SEM evaluation scheme (top) and a normal MEM approach. The plot shows the results in terms of distance to the robust optimizer at $[-1]^n$.

at $[-1]^n$. From this plot it can be seen that when there is no noise, the niching approach easily settles in the four optima that exist in two dimensions. However, when increasing the anticipated input disturbances, the niching approach will encounter more and more difficulties for forming the niches. Hence, the niche formation process seems to become unstable in this particular scenario. An explanation for why this instability can arise is that the four peaks lie very close to each other. At times, neighboring niches might be destroyed when an individual in one niche with a randomly high fitness lies to close to the other niche. This way, niches can be destroyed and prevent each other to settle in a peak.

From the small preliminary study, we can conclude that although including niching techniques might be a good idea, using straightforward niching implementations in combination with SEM/MEM evaluation may cause undesirable instabilities. For the niching approach considered in this section it would be interesting to study its behavior first on noisy optimization problems and find out how it can be stabilized before including it in the context of finding robust optima. When this is achieved, an interesting idea with respect to the niche radius is to link the niche radius to the magnitude of the input uncertainty. That is, the input uncertainties provide an indicator that fits naturally for specifying the niche radius. The latter remains a parameter that depends on the problem at hands.

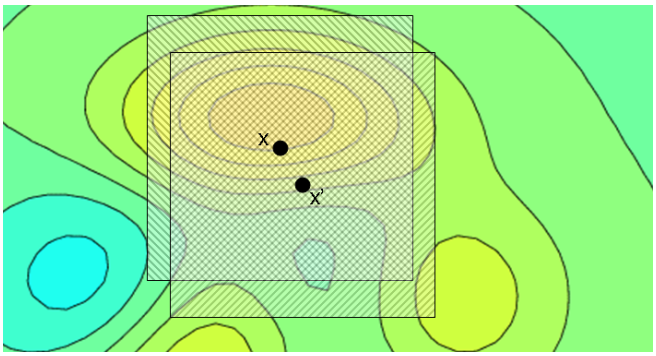


Figure 7.23: Overlap sketch: two solutions $\mathbf{x}$ and $\mathbf{x}'$ around which square regions are drawn indicating the regions of uncertainty $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{x}'}$.

7.2.10 Exploiting Overlap

The approaches that have been discussed so far aim to obtain as accurate as possible fitness approximations using as few as possible objective function evaluations. In [KEDB10b], a different approach is followed.

An important observation when aiming to find robust optima is that sampling regions (i.e., η-neighborhoods) of different candidate solutions in (successive) populations are often overlapping, particularly in later stages of the optimization where the population focuses on a single region of the search space. Hence, the evaluations made in overlapping regions can be used at the same time for evaluating the robustness of different candidate solutions. Or, in some cases, it is even possible to discard a large part of the sampling space when comparing solutions. Consider, for example, the scenario displayed in Figure 7.23, where two solutions $\mathbf{x}$ and $\mathbf{x}'$ are compared given a uniform distribution of the input noise. Here, it suffices to sample the non-intersecting regions, because the contribution of the intersecting region to the expected fitness is the same for both solutions. Instead of trying to acquire an estimate of the effective fitness for each candidate solution separately, alternatively one could compare solutions based on how they relate in their effective fitness; therewith exploiting the overlap of their η-neighborhoods.

Consider the special case of comparing two candidate solutions $\mathbf{x}$ and $\mathbf{x}'$ on their expected fitness based on a uniform perturbation $\boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{l}, \mathbf{l})$. By normalizing the search space we can transform the sampling intervals of the independent input variables such that they have the same width l. For this scenario (illustrated in Figure 7.24), the intersection region A of $\eta_{\mathbf{x}}$ and

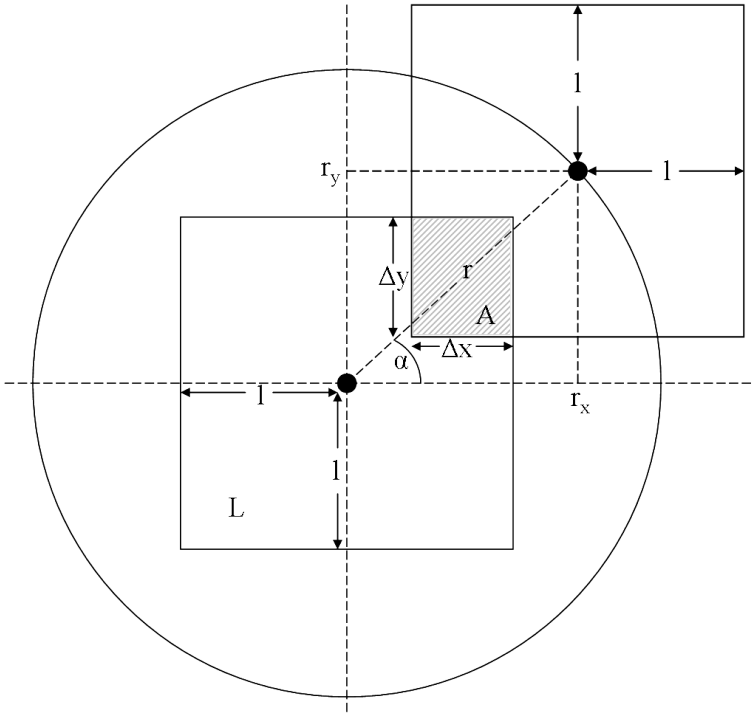


Figure 7.24: A two-dimensional sketch of the overlap of the regions of uncertainty $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{x}'}$ of the two solutions $\mathbf{x}$ and $\mathbf{x}'$.

$\eta_{\mathbf{x}'}$ contributes the same to the expected fitness of both solutions, i.e.,

$$f_{\exp_{\mathbf{x}}} = \int_{\tilde{\mathbf{x}}\in\eta_{\mathbf{x}}\setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} + \int_{\tilde{\mathbf{x}}\in A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{7.37}$$

$$f_{\exp_{\mathbf{x}'}} = \int_{\tilde{\mathbf{x}}\in\eta_{\mathbf{x}'}\setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} + \int_{\tilde{\mathbf{x}}\in A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{7.38}$$

$$f_{\exp_{\mathbf{x}}} - f_{\exp_{\mathbf{x}'}} = \int_{\tilde{\mathbf{x}}\in\eta_{\mathbf{x}}\setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} - \int_{\tilde{\mathbf{x}}\in\eta_{\mathbf{x}'}\setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{7.39}$$

where $\mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}}) \sim \mathrm{pdf}(\mathbf{x}+\boldsymbol{\delta})$ and $\mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}}) \sim \mathrm{pdf}(\mathbf{x}'+\boldsymbol{\delta})$. Hence, for comparisons, A offers no relevant information. Moreover, when $\mathbf{x}$ and $\mathbf{x}'$ are located close to each other, A will be large compared to $\eta_{\mathbf{x}}\setminus A$ and $\eta_{\mathbf{x}'}\setminus A$ and for an evaluation method based on sampling in $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{x}'}$, the probability of sampling within $\eta_{\mathbf{x}}\setminus A$ and $\eta_{\mathbf{x}'}\setminus A$ will be small. Given the surface area (or volume) $|A|$ of A, the probability that one uniformly drawn random sample in $\eta_{\mathbf{x}}$ hits $\eta_{\mathbf{x}}\setminus A$ is

$$P(\text{sample not in } A \mid \text{sample in } \eta_{\mathbf{x}}) = 1 - \frac{|A|}{|\eta_{\mathbf{x}}|} = 1 - \frac{|A|}{2^n l^n}. \tag{7.40}$$

Let X denote the discrete random variable for the number of samples n until the first sample in $\eta_{\mathbf{x}}\setminus A$ is obtained. For a pure Monte-Carlo sampling scheme, this leads to the following

expected number of samples needed to obtain one sample in $\eta_{\mathbf{x}}$ that is not in A

$$E(X) = \sum_{n=1}^{\infty} n \cdot P(X = n) = \frac{1}{1 - \frac{|A|}{\eta_{\mathbf{x}}}} = \frac{2^n l^n}{2^n l^n - |A|}. \tag{7.41}$$

Obviously, the same reasoning can be followed for $\eta_{\mathbf{x}'}$. Provided that there is an overlap in the regions of uncertainty, $|A|$ can be computed as

$$|A| = \prod_{i=1}^{n} (2l - |x_i - x_i'|)\,. \tag{7.42}$$

Or, alternatively, in two dimensions the following expression can be derived for $|A|$, given that $\mathbf{x}$ and $\mathbf{x}'$ lie at a distance r from each other and make an angle α with respect to the x-axis

$$|A| \quad = \quad \begin{cases} (2l - r_x) \cdot (2l - r_y) & \text{, if } r_x \leq 2l \text{ and } r_y \leq 2l \\ 0 & \text{, otherwise} \end{cases}. \tag{7.43}$$

Here, $r_x = |r \cos \alpha|$ and $r_y = |r \sin \alpha|$. From this, and assuming a periodicity of $\pi/2$ for α (i.e., $\alpha \in [0, \pi/2]$), an expression can be derived for the expected number of samples in $\eta_{\mathbf{x}}$, m, needed in order to hit the area $\eta_{\mathbf{x}} \backslash A$ at least c times

$$m \quad = \quad \begin{cases} 4cl^2 / \left(2lr\left(\cos\alpha + \sin\alpha\right) - r^2 \cos\alpha \sin\alpha\right) & \text{, if } r_x \leq 2l \text{ and } r_y \leq 2l \\ c & \text{, otherwise} \end{cases} \tag{7.44}$$

By expressing r in terms of l, substituting $r = kl$, this can be simplified to

$$m \quad = \quad \begin{cases} 4c / \left(2k\left(\cos\alpha + \sin\alpha\right) - k^2 \cos\alpha \sin\alpha\right) & \text{, } k\cos\alpha \leq 2 \text{ and } k \sin\alpha \leq 2 \\ c & \text{, otherwise} \end{cases} \tag{7.45}$$

The derivation above is still dependent on the angle α between $\mathbf{x}$ and $\mathbf{x}'$. It is desirable to have an approximation independent of α. This yields a general approximation for the required number of samples m needed for two individuals at a distance r (still using $r = lk$, i.e., $k = r/l$) to have at least c samples in the regions $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{x}'}$ respectively. For this, we look at the upper bound of $|A|$ with respect to α and note that for $|A|$ to be maximized, $\alpha = 0$ or $\alpha = \pi/4$

$$\begin{aligned} |A| \quad &= \quad \begin{cases} \max\{4l^2 - 2kl^2, 4l^2 - 2\sqrt{2}kl^2 + \frac{1}{2}k^2l^2\} & \text{, if } k \leq 2\sqrt{2} \\ 0 & \text{, otherwise} \end{cases} \\ &= \quad \begin{cases} 4l^2 - 2\sqrt{2}kl^2 + \frac{1}{2}k^2l^2 & \text{, if } k \leq 4\left(\sqrt{2} - 1\right) \\ 4l^2 - 2kl^2 & \text{, if } 4\left(\sqrt{2} - 1\right) < k \leq 2\sqrt{2}\,. \\ 0 & \text{, otherwise} \end{cases} \end{aligned} \tag{7.46}$$

This upper bound of A can be used to approximate (the upper bound of) m

$$m = \begin{cases} 8c/\left(4\sqrt{2}k - k^2\right) & \text{, if } k \leq 4\left(\sqrt{2}-1\right) \\ 2c/k & \text{, if } 4\left(\sqrt{2}-1\right) < k \leq 2\sqrt{2} \\ 0 & \text{, otherwise} \end{cases}. \tag{7.47}$$

For general n-dimensional cases, $|A|$ can be assumed to be maximized in the cases equivalent to the two-dimensional cases of $\alpha = 0$ and $\alpha = \pi/4$. Following this, and again using the substitution $r = kl$, we obtain

$$|A| = \begin{cases} 2^n l^n \max\left\{\left(1 - \frac{1}{2}k\right), \left(1 - \frac{1}{2\sqrt{n}}k\right)^n\right\} & \text{, if } k \leq 2\sqrt{n} \\ 0 & \text{, otherwise} \end{cases}, \tag{7.48}$$

which can be used to obtain an expression for the expected number of samples in $\eta_{\mathbf{x}}$ needed to hit $\eta_{\mathbf{x}} \backslash A$ at least c times

$$m = \begin{cases} c \max\left\{\frac{2}{k}, 1/\left(1 - \left(1 - \frac{1}{2\sqrt{n}}k\right)^n\right)\right\} & \text{, if } k \leq 2\sqrt{n} \\ c & \text{, otherwise} \end{cases}. \tag{7.49}$$

It is clear that when using normal sampling approaches, many samples are practically wasted when the distance between two solutions becomes small. Figure 7.25 shows the number of required samples m needed to hit $\eta_{\mathbf{x}} \backslash A$ at least once versus k for $n = 10^0, 10^1, \ldots, 10^6$. Interestingly, the plots are not much different for all values of n. For $k \lesssim 1.7$ the term $2/k$ becomes the determining factor and the other term leads to $m \approx 1$, even for $n = 10^6$. Hence, the following rule-of-thumb can be used to indicate the growth of m relative to k

$$m = \begin{cases} \frac{2c}{k} & \text{, if } k \leq 1.7 \\ c & \text{, otherwise} \end{cases}. \tag{7.50}$$

In [KEDB10b], a rejection based sampling procedure is proposed to obtain m samples in $\eta_{\mathbf{x}} \backslash \eta_{\mathbf{x}'}$. This approach is described in Algorithm 7.7 and simply works by uniformly sampling in $\eta_{\mathbf{x}}$ and rejecting samples in $\eta_{\mathbf{x}} \cap \eta_{\mathbf{x}'}$. Although the practical viability of this scheme is limited, it can be incorporated into a simple (1+1)-Evolution Strategy to test the approach of exploiting overlap. Additionally, besides the fact that we can either avoid the region A in case of uniform noise, or at least reuse the samples for the evaluation of $\mathbf{x}$ and $\mathbf{x}'$ in case of other noise distributions, there is also a symmetry in the regions $\eta_{\mathbf{x}} \backslash A$ and $\eta_{\mathbf{x}'} \backslash A$. Given a perturbation $\mathbf{x}_\mathrm{p} \sim U(-\mathbf{l}, \mathbf{l})$ we note that

$$\mathbf{x} + \mathbf{x}_\mathrm{p} \in \eta_{\mathbf{x}} \backslash A \Leftrightarrow \mathbf{x}' - \mathbf{x}_\mathrm{p} \in \eta_{\mathbf{x}'} \backslash A. \tag{7.51}$$

Hence, when using the rejection based sampling algorithm of Algorithm 7.7, it is only required to obtain one set of samples for every pair $\mathbf{x}$ and $\mathbf{x}'$, because $\mathbf{X}'$ can be deduced from $\mathbf{X}$.

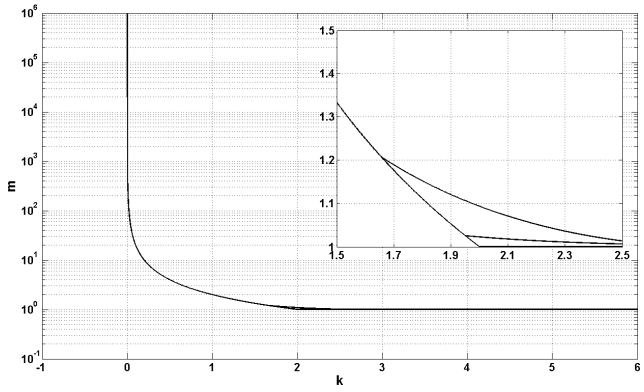


Figure 7.25: The number of samples needed to obtain at least 1 sample in the region $\eta_{\mathbf{x}} \backslash A$, plotted against the normalized distance k between $\mathbf{x}$ and $\mathbf{x}'$ for $n = 10^0, 10^1, \ldots, 10^6$ with a zoom on the interval $k \in [1.5, 2.5]$.

Algorithm 7.7: Rejection Based Uniform Sampling in $\eta_{\mathbf{x}} \backslash \eta_{\mathbf{x}'}$

input: η-neighborhoods $\eta_{\mathbf{x}}$ and $\eta_{\mathbf{x}'}$

output: a set $\mathbf{X}$ of m uniformly drawn samples in $\eta_{\mathbf{x}} \backslash \eta_{\mathbf{x}'}$

1: $\mathbf{X} \leftarrow \emptyset$
2: **while** $|\mathbf{X}| < m$ **do**
3: $\quad \mathbf{x}_{\mathrm{s}} \sim U(\eta_{\mathbf{x}})$
4: $\quad$ **if** $\mathbf{x}_{\mathrm{s}} \notin \eta_{\mathbf{x}'}$ **then**
5: $\quad\quad \mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}_{\mathrm{s}}\}$
6: $\quad$ **end if**
7: **end while**
8: **return** $\mathbf{X}$

As a proof of concept, in [KRD+11], two versions of the (1+1)-ES are compared on the sphere problem (see Appendix B.1): one incorporating a $\mathrm{MEM}^{+}_{\mathrm{MS}}$ scheme (and reevaluation of the parent) and one using the rejection based sampling approach for comparing the parent and the offspring. In both schemes the sample size was set to $m = 2n = 20$ and an evaluation budget of $2 \cdot 10^6$ was used. The results of one run of both schemes are shown in Figure 7.26.

From the convergence plots (the top left plot for the normal sampling approach and the top

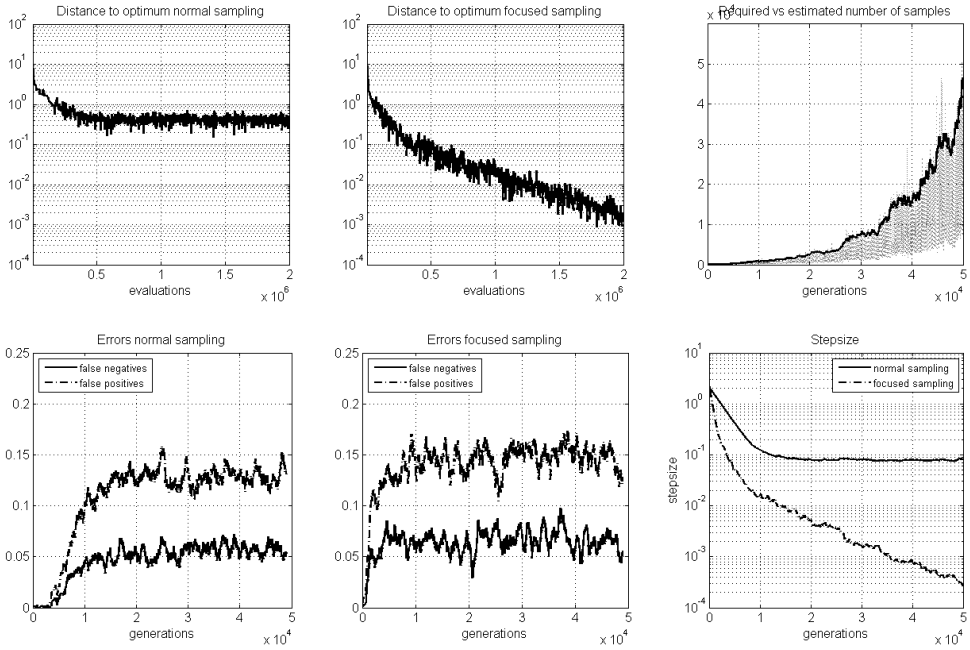


Figure 7.26: Left column: distance to the optimum and ordering error frequency of the normal sampling approach. Middle column: distance to the optimum and ordering error frequency of the focused sampling approach. Top right: the required number of samples and an upper bound estimator for the required number of samples of the focused sampling run. Bottom right: the stepsize development for both approaches.

right plot for the rejection based sampling approach) it can be seen that while the normal sampling stagnates after about $5 \cdot 10^5$ evaluations, there is no sign of stagnation for the approach implementing the focused sampling. Hence, although it still uses $m = 20$ samples for each evaluation, the rejection based sampling approach remains making progress. Supported also by the stepsize plots (bottom right), these empirical results suggest linear generation- and evaluation-wise convergence for this particular problem.

The error frequency plots (bottom left for the normal sampling approach and bottom right for the rejection based sampling approach) show the frequency of false negatives and false positives versus the number of generations (these frequencies are computed over a window of 1000 generations). For both approaches, the error frequencies stay at the same levels at a certain point in time. For the normal sampling approach, this can be related to the stagnation of the stepsize (i.e., the error rate is coupled to the noise ratio, which is directly coupled to the stepsize). However, for the rejection based sampling approach the error rates stay at the same levels even with the stepsize decreasing.

The number of samples that required in order to obtain m samples in $\eta_{\mathbf{x}} \backslash A$ is shown in the top right plot. Note that the implementation used for these experiments uses the simple

rejection method of Algorithm 7.7. The thick solid line shows for every generation the expected upper bound of the number of samples, computed using the simple rule-of-thumb of Eq. 7.50, with $r = \sigma\sqrt{n}$. This plot shows how this rule-of-thumb accurately determines the upper bound for the required number of samples, but also that in many cases fewer samples suffice.

In conclusion, the ideas proposed in [KEDB10b] provide an alternative view on robustness evaluation where the focus is not on trying to obtain accurate robustness estimates for individual candidate solutions, but rather to compare the solutions of (successive) populations with respect to robustness.

The experiments on the $(1{+}1)$-ES show how the proposed idea can successfully be integrated in common optimization algorithms. However, a step is still to be made to also include this concept in population based schemes that work with sets of candidate solutions rather than two. An implementation for tournament selection can easily be derived, but for $(\mu/\rho\overset{+}{,}\lambda)$-selection, more sophisticated schemes are required. For $(\mu/\rho\overset{+}{,}\lambda)$-schemes, samples of overlapping regions can at least be reused when evaluating candidate solutions, but specifically targeting non-overlapping regions will become computationally more expensive. An idea that requires further development is to replace the rejection based sampling by Gibbs sampling [CG92, GW92]. This would decrease the computational complexity of the sampling method tremendously, making this approach from that perspective viable.

Finally, two issues that should still be addressed are: 1) the way in which this sampling strategy can be adopted in cases of other input noise distributions, and 2) the applicability of this method in case there are disturbance-free design variables. In the former case, the overlapping region may still be relevant for comparing two candidate solutions. In the latter case, there will never be an overlap, hence, in principle this method cannot be applied.

7.3 Summary and Discussion

This chapter has discussed the problem of finding robust optima. It has been shown how these types of problems may be modeled within the optimization problem statement and how different robustness measures can be derived from this. The problem of finding robust optima is therefore reformulated into the problem of optimization of the effective objective functions, while satisfying the effective constraints.

The effective objective and constraint functions can be seen as transformations of the original objective and constraint functions. However, as precise evaluation of these functions is often impossible, approximation methods are required. The algorithmic design aspect is therewith reduced to efficient approximation or comparison of candidate solutions with respect to the effective objective and constraint functions.

In the second part of this chapter, techniques have been reviewed that can be used within Evolution Strategies to find robust optima. The particular focus of this chapter was on single objective optimization of the expected objective function.

The myopic approach (Section 7.2.1) is the strategy which simply uses canonical instances of the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. This view relies on the observation that Evolutionary Algorithms in practice already tend to converge to the more robust peaks. It has been shown to be quite effective for identifying the more robust peaks, but it fails in case of shifted robust optimizers.

When actively accounting for robustness, using Monte-Carlo integration methods is the most straightforward way to evaluate the effective fitness of candidate solutions. Compared to myopic approaches, these evaluation approaches (named SEM when using only one sample and MEM when using multiple samples) allow for a closer convergence to shifted robust optimizers. Additionally, for MEM evaluation approaches one could use the same sample perturbations for all individuals in the population and base the Monte-Carlo integration on a sample set obtained with Latin Hypercube sampling. These modifications seem to be beneficial when the region around the robust optimum is symmetric, but can yield divergent behavior when sharp ridges cause the robust optimizer to be shifted. A brief comparison of MEM evaluation approaches with m samples to the alternative of using SEM evaluation and increasing the population size with the same rate m (i.e., implicit averaging) shows that for finding robust optima, explicit resampling seems to work better.

Similar to noisy objective functions, adaptive averaging techniques can also be used in the context of finding robust optima. These techniques provide a way of omitting the problem of determining an appropriate sample size m and in theory allow Evolution Strategies to zoom in on the optimizer with arbitrary precision, although in practice convergence can be slow.

For objective functions for which evaluations are expensive, archive based approaches and metamodeling approaches can be used for finding robust optima. These techniques aim to efficiently use an archive of previous evaluations to serve as samples for robustness evaluation. Although the overhead cost of these methods is considerable, these techniques provide promising results compared to standard MEM approaches. A difficulty is, however, that especially in higher dimensional search spaces it takes a while before the archive is filled and can effectively be used for robustness estimation or metamodeling.

When aiming to actively improve the capabilities of targeting the more robust peaks, it has been suggested to use niching techniques. However, although this idea sounds promising, in this chapter it has been observed that a straightforward implementation of niching with a MEM evaluation approach tends to become unstable. In this context it should be accounted for that objective function evaluations are noisy, hence, the niching technique should be robust against noise. The niching technique considered in Section 7.2.9 is, without modifications, not suitable for this.

Lastly, an alternative evaluation scheme has been considered in which it is suggested to compare pairs of individuals with respect to their robustness rather than trying to obtain accurate robustness estimates. By exploiting the overlap in the regions of uncertainty (or η-neighborhoods) it is in some cases possible to drastically improve the convergence accuracy

of MEM evaluation approaches (Section 7.2.10). However, this approach has been formulated so far only for simple $(1+1)$-schemes and requires further study to be applied in practical scenarios.

The techniques for finding robust optima that are discussed in this chapter represent the main classes of approaches that can be followed. Based on this review, the myopic approach, the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ and $\mathrm{MEM}^{-}_{\mathrm{MS}}$, the adaptive averaging approach, the archive based approach, and the metamodeling approach are feasible approaches for practical scenarios. The question that remains is which of these techniques can best be used in practice. In Chapter 8, we will study this question.

Chapter 8

Empirical Study on Finding Robust Optima

In Chapter 7, the problem of finding robust optima in the context of Evolution Strategies has been discussed and an overview has been given of techniques for solving such problems. In this review, the myopic approach, the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ approach, the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ approach, the adaptive averaging approach, the archive based approach, and the metamodeling approach are shown to be feasible approaches for practical scenarios. An interesting and yet unanswered question is: how do these different techniques compare against each other with respect to performance? This chapter presents the results of an empirical study that is designed to shed some light on this question.

The structure of this chapter is as follows: Section 8.1 describes the general experimental setup adopted in this empirical study. Section 8.2 shows the results of an empirical study for finding the optimal sample sizes for the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ and the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ approach. Section 8.3 shows the results of a full empirical comparison of the different methods for finding robust optima. Section 8.4 closes with a summary and discussion.

8.1 Experimental Setup

The purpose of the experimental study is to find out how the different evaluation techniques for finding robust optima compare when used within the same algorithmic basis, namely the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. The general experimental settings, shown in Table 8.1, restrict to one particular search space dimension size, $n = 10$, and an evaluation budget of $10,000$ function evaluations, which is taken as a standard setup throughout this chapter. For the assessment of the quality of each scheme, we record the final solution quality over multiple runs. Here, the final solution quality refers to a highly accurate Monte-Carlo approximation (using $m = 1000$ samples) of the expected objective function value of the solution returned after each optimization run).

General experimental settings	
Search space dimension size	$n = 10$
Evaluation budget per run	10,000
Runs per algorithmic scheme	50
Performance indicators	Final solution quality, approximated with Monte-Carlo integration using 1000 samples, (mean, std, median) over all runs, and rank sum for ranking of the algorithmic schemes

Table 8.1: The general experimental setup.

The test problems are enlisted in Table 8.2 and full descriptions can be found in Appendix B. The set of test problems is constructed based on test problems from literature. It incorporates different difficulties that can be encountered within these types of optimization problems.

The *RO Sphere Problem* is a modified version of the original *Sphere Problem*. Obviously, a myopic approach will perform much better on this test problem than any scheme that aims to approximate the expected objective function. However, it is still useful as a test problem for comparing the convergence limitations of the schemes designed for finding robust optima. The *RO Heaviside Sphere Problem* and the *RO Sawtooth Problem* are problems with a shifted robust optimizer that emerges due to a sharp ridge at the original optimum. The *RO Volcano Problem* is a problem with a plateau where the robust is located in the center. The other problems have multimodal objective functions in which the robust optimizer is classified as emergent. The *RO Pickelhaube Problem* is a problem with two peaks, and the algorithmic challenge is to target the most robust peak. The *RO Branke Multipeak Problem* has 2^n peaks, varying in robustness. The other two multipeak problems are more complex and provide cases in which the emergent optimizer is also shifted with respect to the original local optimizers.

Test problem	Properties of the underlying signal function	
RO Sphere Problem	unimodal	robust optimizer equals original optimizer
RO Heaviside Sphere Problem	unimodal	shifted robust optimizer
RO Sawtooth Problem	unimodal	shifted robust optimizer
RO Volcano Problem	unimodal	shifted robust optimizer
RO Pickelhaube Problem	multimodal	emergent robust optimizer
RO Branke's Multipeak Problem	multimodal	emergent robust optimizer
RO Multipeak F1 Problem	multimodal	emergent robust optimizer
RO Multipeak F2 Problem	multimodal	emergent robust optimizer

Table 8.2: The test problems used for empirical comparison.

The different evaluation schemes for finding robust optima that are compared in this empirical study are enlisted in Table 8.3. For the multi-evaluation methods, two variants are considered, namely $\text{MEM}^{-}_{\text{MS}}$ and $\text{MEM}^{+}_{\text{LHS}}$. Theses are considered to be tuned optimally for each test problem, therefore, Section 8.2 presents the results of the tuning of these methods. Both resampling methods are also considered in an adaptive averaging form: UH-$\text{MEM}^{-}_{\text{MS}}$ and UH-$\text{MEM}^{+}_{\text{LHS}}$. These two adaptive averaging methods use the rank-based adaptive averaging approach for updating the sample size, as described in Section 7.2.5. The ABRSS and the Kriging metamodeling approach are used as presented in Section 7.2.7 and Section 7.2.8 respectively.

Evaluation schemes for finding robust optima	
Myopic	A canonical $(5/2_{DI}, 35)$-σSA-ES and CMA-ES.
$\mathbf{MEM^{-}_{MS}}$	The multi-evaluation method (MEM) using Monte-Carlo integration and resampling the disturbances for all individuals in a generation.
$\mathbf{MEM^{+}_{LHS}}$	The multi-evaluation method (MEM) using Latin Hypercube sampling and the same disturbances for all individuals in a generation.
UH-$\mathbf{MEM^{-}_{MS}}$	The rank-based adaptive averaging method using the $\text{MEM}^{-}_{\text{MS}}$ evaluation scheme.
UH-$\mathbf{MEM^{+}_{LHS}}$	The rank-based adaptive averaging method using the $\text{MEM}^{+}_{\text{LHS}}$ evaluation scheme.
ABRSS	The archive based evaluation approach.
Kriging	The Kriging (metamodeling) based evaluation approach.

Table 8.3: The methods considered in the empirical study for finding robust optima.

8.2 Tuning the Static Resampling Schemes

For the empirical comparison of the schemes enlisted in Table 8.3, we consider optimally tuned versions of the $\text{MEM}^{-}_{\text{MS}}$ and the $\text{MEM}^{+}_{\text{LHS}}$ schemes for each test problem. Hence, before presenting the results on the full comparison, Section 8.2.1 shows the results of the tuning experiments of the $\text{MEM}^{-}_{\text{MS}}$ evaluation scheme and Section 8.2.2 shows the results of the tuning experiments of the $\text{MEM}^{+}_{\text{LHS}}$ evaluation scheme.

8.2.1 The Optimal Sample Size for $\text{MEM}^{-}_{\text{MS}}$

This experiment is done in order to determine, for each test problem, the optimal sample size for the $\text{MEM}^{-}_{\text{MS}}$ evaluation scheme. Different instances of the $\text{MEM}^{-}_{\text{MS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\text{MEM}^{-}_{\text{MS}}$-CMA-ES are considered with varying sample sizes: $m = 1, 2, \ldots, 10$. These sample sizes are compared on the test problems listed in Table 8.2 using the experimental setup shown in Table 8.1. The results of these experiments are shown in the tables and figures of

Section 8.2.1.1 and Section 8.2.1.2 for the $\mathrm{MEM}^{-}_{\mathrm{MS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\mathrm{MEM}^{-}_{\mathrm{MS}}$-CMA-ES respectively.

Based on the results, we conclude that for the explicit averaging schemes (the $\mathrm{MEM}^{-}_{\mathrm{MS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\mathrm{MEM}^{-}_{\mathrm{MS}}$-CMA-ES) for each of the test problems with respect to the general experimental setup the optimal sample sizes lie at the values shown in Table 8.4. From these results we observe the trade-off in convergence speed versus convergence accuracy. It depends on the test problem which sample size is most suitable, i.e., there is no clear winner. It seems that for the CMA-ES a slightly higher sample size is required than for the $(5/2_{DI}, 35)$-σSA-ES. Also, we see that for these test problems, the SEM evaluation approach is not optimal in any case.

	MEM$^{-}_{\mathrm{MS}}$-$(5/2_{DI}, 35)$-σSA-ES	**MEM$^{-}_{\mathrm{MS}}$-CMA-ES**
RO Sphere Problem	m = 6	m = 10
RO Heaviside Sphere Problem	m = 9	m = 7
RO Sawtooth Problem	m = 4	m = 8
RO Volcano Problem	m = 5	m = 10
RO Pickelhaube Problem	m = 3	m = 4
RO Branke's Multipeak Problem	m = 3	m = 4
RO Multipeak F1 Problem	m = 2	m = 6
RO Multipeak F2 Problem	m = 6	m = 8

Table 8.4: The optimal sample size for the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ approach to achieve best convergence accuracy on a budget of $10,000$ function evaluations.

8.2.1.1 Results $\mathrm{MEM}^{-}_{\mathbf{MS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	8.61	11.88	4.85	20487	10
MEM2 MS-	4.13	0.33	4.06	13628	8
MEM3 MS-	7.64	12.98	3.97	12697	6
MEM4 MS-	4.34	3.00	3.81	9550	2
MEM5 MS-	5.33	7.12	3.85	9790	3
MEM6 MS-	**3.89**	**0.30**	**3.80**	**8695**	**1**
MEM7 MS-	6.27	8.41	3.95	11791	5
MEM8 MS-	6.46	9.32	3.89	10880	4
MEM9 MS-	4.96	5.40	4.03	13067	7
MEM10 MS-	7.42	10.27	4.18	14665	9

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.49	0.39	0.42	20794	10
MEM2 MS-	0.28	0.16	0.26	16241	9
MEM3 MS-	0.25	0.24	0.22	14843	8
MEM4 MS-	0.17	0.08	0.16	11802	7
MEM5 MS-	0.15	0.07	0.13	9861	3
MEM6 MS-	0.21	0.25	0.15	11672	5
MEM7 MS-	0.25	0.40	0.15	11718	6
MEM8 MS-	0.18	0.25	0.12	8944	2
MEM9 MS-	**0.18**	**0.37**	**0.10**	**8183**	**1**
MEM10 MS-	0.22	0.31	0.14	11192	4

RO SAWTOOTH PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.33	0.12	0.29	19929	10
MEM2 MS-	0.26	0.04	0.25	12821	7
MEM3 MS-	0.30	0.13	0.25	12676	6
MEM4 MS-	**0.25**	**0.04**	**0.24**	**8721**	**1**
MEM5 MS-	0.26	0.06	0.25	10063	3
MEM6 MS-	0.25	0.03	0.24	9248	2
MEM7 MS-	0.27	0.09	0.24	10284	4
MEM8 MS-	0.26	0.06	0.25	10917	5
MEM9 MS-	0.29	0.10	0.26	14090	8
MEM10 MS-	0.29	0.08	0.27	16501	9

RO VOLCANO PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.88	0.40	0.77	17980	10
MEM2 MS-	0.82	0.42	0.73	13189	8
MEM3 MS-	0.77	0.26	0.72	11845	5
MEM4 MS-	0.79	0.37	0.71	9808	2
MEM5 MS-	**0.75**	**0.26**	**0.71**	**8412**	**1**
MEM6 MS-	0.76	0.26	0.72	10592	4
MEM7 MS-	0.77	0.30	0.71	10340	3
MEM8 MS-	0.85	0.43	0.73	13006	7
MEM9 MS-	0.86	0.42	0.73	12872	6
MEM10 MS-	0.99	0.56	0.77	17206	9

RO PICKELHAUBE PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.31	0.20	0.29	13072	6
MEM2 MS-	0.26	0.03	0.24	8561	3
MEM3 MS-	**0.26**	**0.03**	**0.24**	**7914**	**1**
MEM4 MS-	0.26	0.03	0.24	7980	2
MEM5 MS-	0.30	0.22	0.28	10740	4
MEM6 MS-	0.27	0.04	0.28	10900	5
MEM7 MS-	0.34	0.27	0.30	14810	7
MEM8 MS-	0.32	0.10	0.29	15474	8
MEM9 MS-	0.34	0.09	0.32	17538	9
MEM10 MS-	0.37	0.11	0.33	18261	10

RO BRANKE MULTIPEAK PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.43	0.09	0.41	17123	10
MEM2 MS-	0.41	0.03	0.40	13275	7
MEM3 MS-	**0.40**	**0.01**	**0.40**	**9237**	**1**
MEM4 MS-	0.42	0.06	0.40	11051	4
MEM5 MS-	0.40	0.01	0.40	11268	5
MEM6 MS-	0.40	0.02	0.40	9877	2
MEM7 MS-	0.40	0.01	0.40	10710	3
MEM8 MS-	0.42	0.06	0.40	12789	6
MEM9 MS-	0.41	0.02	0.40	14230	8
MEM10 MS-	0.43	0.05	0.41	15690	9

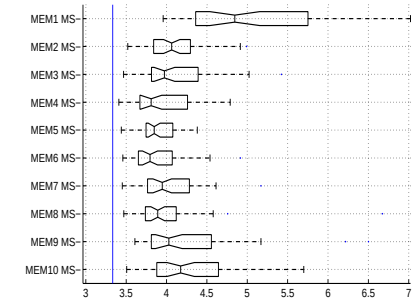
RO MULTIPEAK F1 PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	-0.50	0.09	-0.52	13893	7
MEM2 MS-	**-0.56**	**0.08**	**-0.58**	**8450**	**1**
MEM3 MS-	-0.54	0.08	-0.57	9747	2
MEM4 MS-	-0.53	0.09	-0.55	11273	5
MEM5 MS-	-0.53	0.09	-0.56	10464	3
MEM6 MS-	-0.54	0.08	-0.57	10640	4
MEM7 MS-	-0.50	0.10	-0.53	13312	6
MEM8 MS-	-0.48	0.07	-0.47	15946	9
MEM9 MS-	-0.48	0.07	-0.48	16074	10
MEM10 MS-	-0.48	0.08	-0.46	15451	8

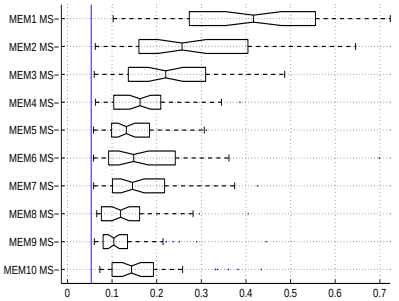
RO MULTIPEAK F2 PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	-0.34	0.20	-0.25	18940	10
MEM2 MS-	-0.50	0.22	-0.62	12415	6
MEM3 MS-	-0.52	0.22	-0.64	11268	4
MEM4 MS-	-0.56	0.20	-0.66	10659	3
MEM5 MS-	-0.58	0.16	-0.65	10153	2
MEM6 MS-	**-0.57**	**0.18**	**-0.68**	**9857**	**1**
MEM7 MS-	-0.53	0.18	-0.63	12242	5
MEM8 MS-	-0.53	0.17	-0.59	12522	7
MEM9 MS-	-0.51	0.17	-0.55	13298	8
MEM10 MS-	-0.51	0.14	-0.52	13896	9

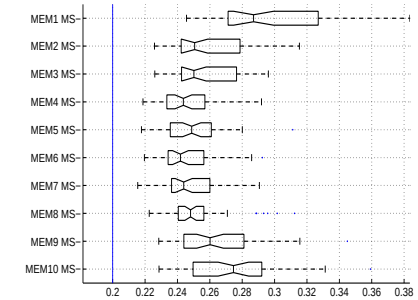
RO SPHERE PROBLEM



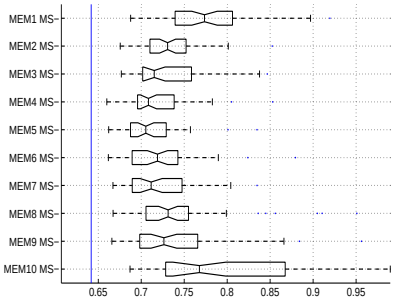
RO HEAVISIDE SPHERE PROBLEM



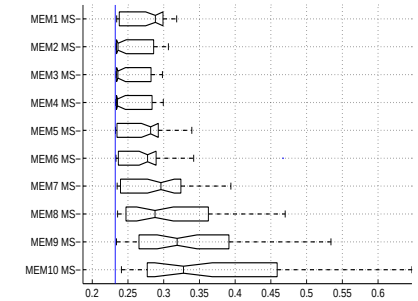
RO SAWTOOTH PROBLEM



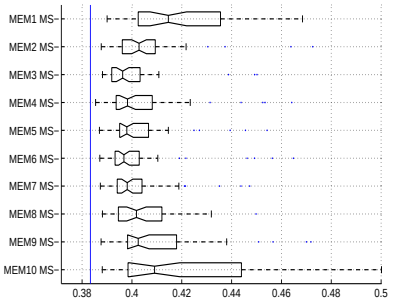
RO VOLCANO PROBLEM



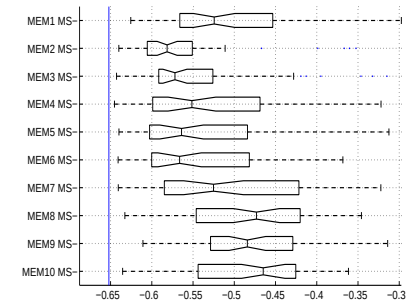
RO PICKELHAUBE PROBLEM



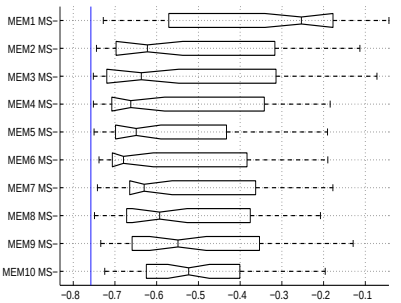
RO BRANKE MULTIPEAK PROBLEM



RO MULTIPEAK F1 PROBLEM



RO MULTIPEAK F2 PROBLEM



8.2.1.2 Results $\mathrm{MEM}^{-}_{\mathrm{MS}}$-CMA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	4.76	0.92	4.60	21779	10
MEM2 MS-	4.99	6.47	3.97	17971	9
MEM3 MS-	3.93	0.31	3.81	15537	8
MEM4 MS-	3.81	0.23	3.75	13268	7
MEM5 MS-	4.02	2.15	3.70	11048	6
MEM6 MS-	5.17	10.44	3.65	10029	4
MEM7 MS-	3.65	0.12	3.63	8952	3
MEM8 MS-	6.52	10.36	3.66	10610	5
MEM9 MS-	**4.21**	**4.10**	**3.57**	**7873**	**1**
MEM10 MS-	5.55	9.55	3.62	8183	2

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.27	0.15	0.26	20171	10
MEM2 MS-	0.23	0.48	0.16	16851	9
MEM3 MS-	0.16	0.09	0.14	15401	8
MEM4 MS-	0.23	0.56	0.10	12720	6
MEM5 MS-	0.11	0.04	0.10	11926	5
MEM6 MS-	0.26	0.63	0.11	12924	7
MEM7 MS-	0.09	0.07	0.08	8429	2
MEM8 MS-	0.17	0.40	0.09	10358	4
MEM9 MS-	0.14	0.36	0.08	9127	3
MEM10 MS-	**0.08**	**0.03**	**0.07**	**7343**	**1**

RO SAWTOOTH PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.29	0.03	0.29	21324	10
MEM2 MS-	0.27	0.03	0.26	18009	9
MEM3 MS-	0.25	0.03	0.25	15046	8
MEM4 MS-	0.25	0.03	0.24	12988	7
MEM5 MS-	0.25	0.05	0.24	11193	5
MEM6 MS-	0.25	0.08	0.23	10102	4
MEM7 MS-	0.25	0.06	0.24	11798	6
MEM8 MS-	**0.23**	**0.01**	**0.23**	**7403**	**1**
MEM9 MS-	0.24	0.05	0.23	9228	3
MEM10 MS-	0.26	0.10	0.23	8159	2

RO VOLCANO PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.76	0.06	0.76	20957	10
MEM2 MS-	0.71	0.04	0.71	16305	9
MEM3 MS-	0.75	0.30	0.70	15816	8
MEM4 MS-	0.70	0.03	0.69	12757	7
MEM5 MS-	0.69	0.02	0.69	11052	5
MEM6 MS-	0.73	0.33	0.69	11309	6
MEM7 MS-	0.72	0.29	0.68	9737	4
MEM8 MS-	0.76	0.38	0.68	9622	3
MEM9 MS-	0.68	0.02	0.68	8966	2
MEM10 MS-	**0.68**	**0.02**	**0.68**	**8729**	**1**

RO BRANKE MULTIPEAK PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.26	0.03	0.24	12457	7
MEM2 MS-	0.26	0.03	0.28	11990	4
MEM3 MS-	0.26	0.02	0.28	11023	3
MEM4 MS-	**0.29**	**0.22**	**0.28**	**10173**	**1**
MEM5 MS-	0.32	0.30	0.28	10280	2
MEM6 MS-	0.36	0.26	0.28	12318	6
MEM7 MS-	0.36	0.23	0.28	12174	5
MEM8 MS-	0.57	0.40	0.30	16495	10
MEM9 MS-	0.37	0.21	0.28	13432	8
MEM10 MS-	0.43	0.30	0.28	14908	9

RO MULTIPEAK F1 PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	0.45	0.07	0.44	13547	7
MEM2 MS-	0.48	0.13	0.41	13000	6
MEM3 MS-	0.46	0.10	0.41	11511	3
MEM4 MS-	**0.46**	**0.12**	**0.40**	**10393**	**1**
MEM5 MS-	0.49	0.14	0.44	12248	5
MEM6 MS-	0.49	0.14	0.40	11151	2
MEM7 MS-	0.53	0.15	0.44	13875	10
MEM8 MS-	0.53	0.14	0.44	13685	8
MEM9 MS-	0.52	0.14	0.45	13790	9
MEM10 MS-	0.50	0.13	0.44	12050	4

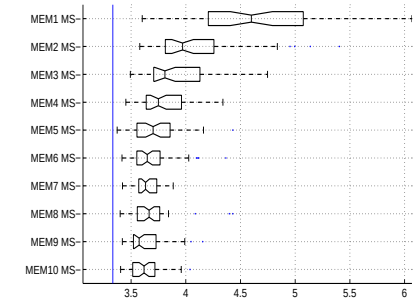
RO MULTIPEAK F2 PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	-0.51	0.07	-0.52	17313	10
MEM2 MS-	-0.53	0.07	-0.54	15347	9
MEM3 MS-	-0.56	0.05	-0.57	11377	3
MEM4 MS-	-0.55	0.08	-0.57	11941	6
MEM5 MS-	-0.56	0.06	-0.57	11243	2
MEM6 MS-	**-0.56**	**0.07**	**-0.58**	**10063**	**1**
MEM7 MS-	-0.54	0.09	-0.58	11649	5
MEM8 MS-	-0.55	0.07	-0.57	11595	4
MEM9 MS-	-0.54	0.09	-0.57	11942	7
MEM10 MS-	-0.54	0.07	-0.56	12780	8

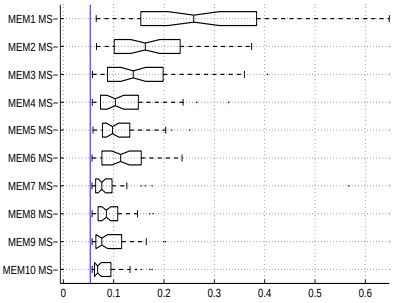
RO FNIM F2 PROBLEM

	Mean	Std	Med	∑#	#
MEM1 MS-	-0.53	0.12	-0.55	19015	10
MEM2 MS-	-0.60	0.11	-0.63	13772	9
MEM3 MS-	-0.64	0.05	-0.66	11116	3
MEM4 MS-	-0.63	0.09	-0.63	12606	7
MEM5 MS-	-0.60	0.12	-0.63	13417	8
MEM6 MS-	-0.63	0.08	-0.63	11119	4
MEM7 MS-	-0.64	0.05	-0.63	11705	6
MEM8 MS-	**-0.64**	**0.07**	**-0.65**	**10395**	**1**
MEM9 MS-	-0.61	0.14	-0.63	11465	5
MEM10 MS-	-0.64	0.07	-0.66	10640	2

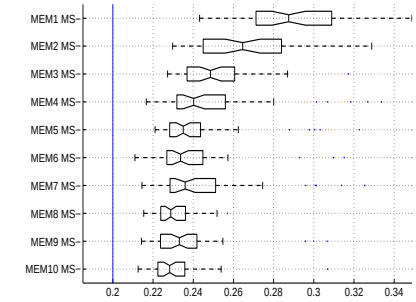
RO SPHERE PROBLEM



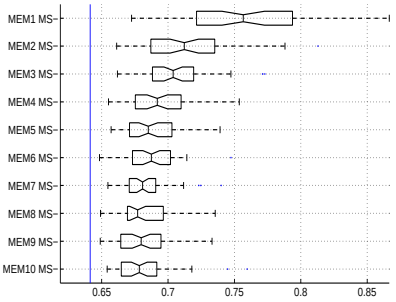
RO HEAVISIDE SPHERE PROBLEM



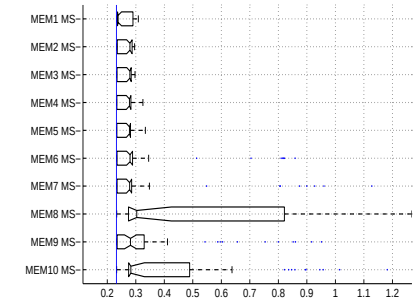
RO SAWTOOTH PROBLEM



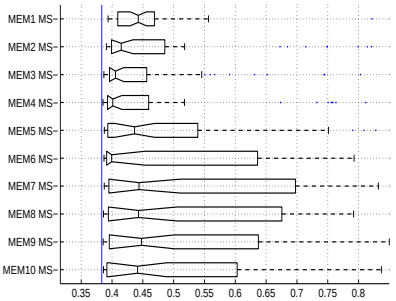
RO VOLCANO PROBLEM



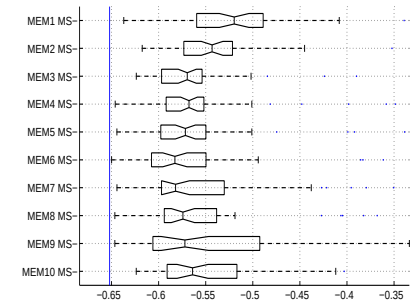
RO PICKELHAUBE PROBLEM



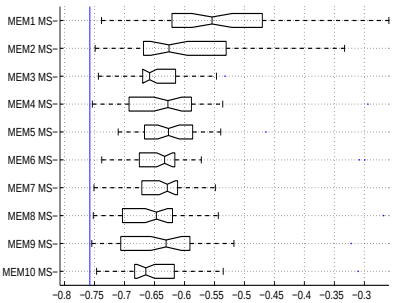
RO BRANKE MULTIPEAK PROBLEM



RO MULTIPEAK F1 PROBLEM



RO MULTIPEAK F2 PROBLEM



8.2.2 The Optimal Sample Size for $\text{MEM}^{+}_{\text{LHS}}$

A first experiment is done in order to determine, for each test problem, the optimal sample size for the $\text{MEM}^{+}_{\text{LHS}}$ evaluation scheme. Different instances of the $\text{MEM}^{+}_{\text{LHS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\text{MEM}^{+}_{\text{LHS}}$-CMA-ES are considered with varying sample sizes: $m = 1, 2, \ldots, 10$. These sample sizes are compared on the test problems listed in Table 8.2 using the experimental setup shown in Table 8.1. The results of these experiments are shown in the tables and figures of Section 8.2.2.1 and Section 8.2.2.2 for the $\text{MEM}^{+}_{\text{LHS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\text{MEM}^{+}_{\text{LHS}}$-CMA-ES respectively.

Based on the results, we conclude that for the explicit averaging schemes, the $\text{MEM}^{+}_{\text{LHS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES and the $\text{MEM}^{+}_{\text{LHS}}$-CMA-ES, for each of the test problems with respect to the general experimental setup the optimal sample sizes lie at the values shown in Table 8.5. The results show a similar picture as observed in the results of Section 8.2.1. Hence, also here the trade-off between convergence accuracy and convergence speed is well visible. Moreover, it seems that the $\text{MEM}^{+}_{\text{LHS}}$ method has a slightly higher optimal sample size, as compared to the $\text{MEM}^{-}_{\text{MS}}$. Also here, the CMA-ES seems to accept a higher sample size than the $(5/2_{DI}, 35)$-σSA-ES.

	$\text{MEM}^{+}_{\text{LHS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES	$\text{MEM}^{+}_{\text{LHS}}$-CMA-ES
RO Sphere Problem	m = 5	m = 8
RO Heaviside Sphere Problem	m = 8	m = 10
RO Sawtooth Problem	m = 7	m = 10
RO Volcano Problem	m = 6	m = 10
RO Pickelhaube Problem	m = 4	m = 3
RO Branke's Multipeak Problem	m = 5	m = 6
RO Multipeak F1 Problem	m = 5	m = 6
RO Multipeak F2 Problem	m = 3	m = 8

Table 8.5: The optimal sample size for the $\text{MEM}^{+}_{\text{LHS}}$ approach to achieve best convergence accuracy on a budget of $10,000$ function evaluations.

8.2.2.1 Results $\text{MEM}^{+}_{\text{LHS}}$-$(5/2_{DI}, 35)$-$\sigma$SA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	7.72	10.14	4.50	22457	10
MEM2 LHS+	4.36	6.10	3.48	17008	8
MEM3 LHS+	4.91	7.63	3.37	10973	5
MEM4 LHS+	3.36	0.01	3.35	6628	2
MEM5 LHS+	**3.97**	**4.35**	**3.35**	**5229**	**1**
MEM6 LHS+	5.65	9.21	3.35	6997	3
MEM7 LHS+	6.12	9.88	3.35	7696	4
MEM8 LHS+	5.18	8.64	3.41	14000	6
MEM9 LHS+	5.00	7.43	3.42	15258	7
MEM10 LHS+	5.48	8.21	3.54	19004	9

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.53	0.32	0.52	19521	9
MEM2 LHS+	0.64	0.26	0.67	21204	10
MEM3 LHS+	0.43	0.24	0.36	17431	8
MEM4 LHS+	0.31	0.31	0.25	14883	7
MEM5 LHS+	0.22	0.28	0.17	11439	6
MEM6 LHS+	0.25	0.47	0.17	10409	5
MEM7 LHS+	0.16	0.27	0.12	7649	3
MEM8 LHS+	**0.17**	**0.28**	**0.11**	**7185**	**1**
MEM9 LHS+	0.24	0.41	0.12	8203	4
MEM10 LHS+	0.13	0.09	0.12	7326	2

RO SAWTOOTH PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.38	0.11	0.35	22446	10
MEM2 LHS+	0.28	0.05	0.27	19425	9
MEM3 LHS+	0.26	0.07	0.24	15067	8
MEM4 LHS+	0.25	0.08	0.23	10824	6
MEM5 LHS+	0.23	0.01	0.23	8006	2
MEM6 LHS+	0.25	0.08	0.23	8719	3
MEM7 LHS+	**0.24**	**0.06**	**0.23**	**7889**	**1**
MEM8 LHS+	0.26	0.10	0.23	9876	4
MEM9 LHS+	0.24	0.05	0.23	10530	5
MEM10 LHS+	0.27	0.09	0.24	12468	7

RO VOLCANO PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.76	0.07	0.76	21241	10
MEM2 LHS+	0.66	0.01	0.66	13464	6
MEM3 LHS+	0.76	0.44	0.65	9709	4
MEM4 LHS+	0.72	0.34	0.65	7378	3
MEM5 LHS+	0.72	0.32	0.65	7193	2
MEM6 LHS+	**0.68**	**0.24**	**0.65**	**5547**	**1**
MEM7 LHS+	0.76	0.43	0.65	10555	5
MEM8 LHS+	0.84	0.54	0.66	13604	7
MEM9 LHS+	0.86	0.51	0.68	17686	8
MEM10 LHS+	0.83	0.42	0.69	18873	9

RO PICKELHAUBE PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.27	0.03	0.28	13646	7
MEM2 LHS+	0.26	0.02	0.27	10019	4
MEM3 LHS+	0.30	0.25	0.27	8408	3
MEM4 LHS+	**0.25**	**0.03**	**0.23**	**6950**	**1**
MEM5 LHS+	0.28	0.19	0.23	7814	2
MEM6 LHS+	0.28	0.13	0.26	11643	5
MEM7 LHS+	0.30	0.21	0.27	13508	6
MEM8 LHS+	0.30	0.08	0.28	15522	8
MEM9 LHS+	0.38	0.23	0.32	17867	9
MEM10 LHS+	0.44	0.22	0.40	19873	10

RO BRANKE MULTIPEAK PROBLEM

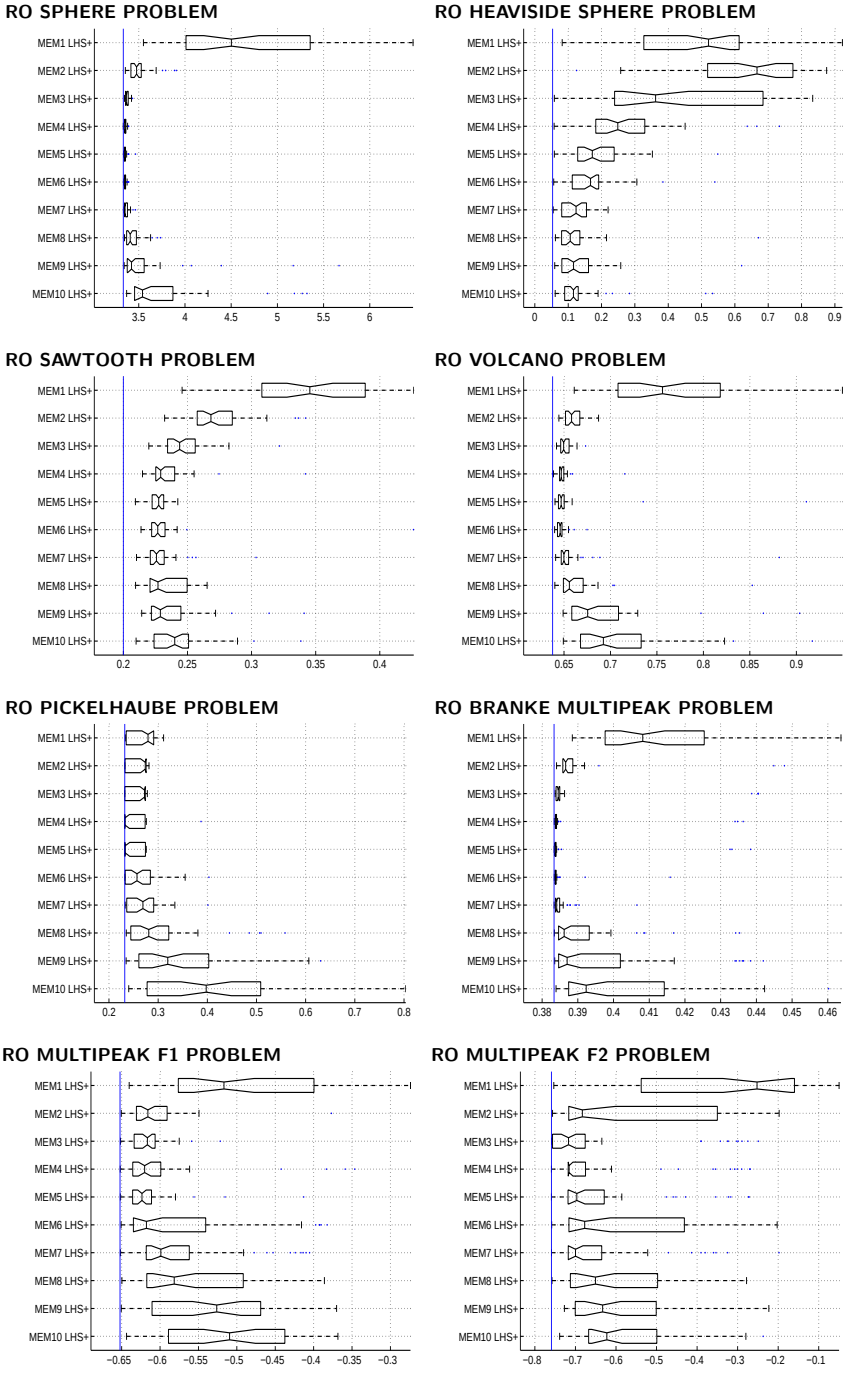
	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.42	0.06	0.41	21356	10
MEM2 LHS+	0.40	0.04	0.39	15944	7
MEM3 LHS+	0.39	0.04	0.38	11200	5
MEM4 LHS+	0.39	0.01	0.38	6913	3
MEM5 LHS+	**0.39**	**0.01**	**0.38**	**5516**	**1**
MEM6 LHS+	0.38	0.00	0.38	5742	2
MEM7 LHS+	0.39	0.00	0.38	9205	4
MEM8 LHS+	0.39	0.02	0.39	14390	6
MEM9 LHS+	0.41	0.04	0.39	16018	8
MEM10 LHS+	0.42	0.05	0.39	18966	9

RO MULTIPEAK F1 PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	-0.49	0.10	-0.52	18600	10
MEM2 LHS+	-0.61	0.04	-0.62	9928	4
MEM3 LHS+	-0.62	0.02	-0.62	8693	3
MEM4 LHS+	-0.60	0.07	-0.62	8388	2
MEM5 LHS+	**-0.62**	**0.04**	**-0.62**	**7648**	**1**
MEM6 LHS+	-0.58	0.09	-0.62	10752	5
MEM7 LHS+	-0.57	0.07	-0.60	12625	6
MEM8 LHS+	-0.55	0.08	-0.58	14524	7
MEM9 LHS+	-0.53	0.08	-0.53	16398	8
MEM10 LHS+	-0.51	0.08	-0.51	17694	9

RO MULTIPEAK F2 PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	-0.35	0.22	-0.25	20208	10
MEM2 LHS+	-0.57	0.20	-0.68	12574	6
MEM3 LHS+	**-0.64**	**0.17**	**-0.72**	**8437**	**1**
MEM4 LHS+	-0.64	0.16	-0.72	9752	2
MEM5 LHS+	-0.64	0.14	-0.70	9753	3
MEM6 LHS+	-0.60	0.17	-0.68	12047	5
MEM7 LHS+	-0.65	0.14	-0.70	10254	4
MEM8 LHS+	-0.61	0.14	-0.65	12637	7
MEM9 LHS+	-0.59	0.14	-0.63	14434	8
MEM10 LHS+	-0.58	0.12	-0.62	15154	9


RO SPHERE PROBLEM
RO HEAVISIDE SPHERE PROBLEM
RO SAWTOOTH PROBLEM
RO VOLCANO PROBLEM
RO PICKELHAUBE PROBLEM
RO BRANKE MULTIPEAK PROBLEM
RO MULTIPEAK F1 PROBLEM
RO MULTIPEAK F2 PROBLEM
MEM1 LHS+
MEM2 LHS+
MEM3 LHS+
MEM4 LHS+
MEM5 LHS+
MEM6 LHS+
MEM7 LHS+
MEM8 LHS+
MEM9 LHS+
MEM10 LHS+

8.2.2.2 Results $\text{MEM}^{+}_{\text{LHS}}$-CMA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	4.18	0.52	4.05	23549	10
MEM2 LHS+	4.06	4.47	3.42	20890	9
MEM3 LHS+	4.41	7.43	3.36	17957	8
MEM4 LHS+	3.35	0.01	3.34	15081	7
MEM5 LHS+	3.34	0.00	3.34	12612	6
MEM6 LHS+	4.45	7.88	3.34	9429	5
MEM7 LHS+	4.64	9.22	3.34	7519	4
MEM8 LHS+	**3.34**	**0.00**	**3.34**	**5681**	**1**
MEM9 LHS+	3.34	0.00	3.33	6016	2
MEM10 LHS+	3.34	0.01	3.34	6516	3

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.58	0.21	0.65	17032	8
MEM2 LHS+	0.70	0.35	0.72	18879	9
MEM3 LHS+	0.68	0.18	0.74	19527	10
MEM4 LHS+	0.59	0.38	0.74	16824	7
MEM5 LHS+	0.50	0.51	0.38	14130	6
MEM6 LHS+	0.28	0.22	0.19	10187	4
MEM7 LHS+	0.29	0.32	0.20	10288	5
MEM8 LHS+	0.16	0.07	0.15	6904	3
MEM9 LHS+	0.22	0.45	0.12	6435	2
MEM10 LHS+	**0.17**	**0.35**	**0.11**	**5044**	**1**

RO SAWTOOTH PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.34	0.06	0.33	22652	10
MEM2 LHS+	0.28	0.05	0.27	19178	9
MEM3 LHS+	0.26	0.06	0.25	16762	8
MEM4 LHS+	0.25	0.05	0.24	14660	7
MEM5 LHS+	0.24	0.06	0.23	13380	6
MEM6 LHS+	0.25	0.10	0.23	11685	5
MEM7 LHS+	0.24	0.09	0.22	8434	4
MEM8 LHS+	0.22	0.02	0.22	6097	2
MEM9 LHS+	0.24	0.09	0.22	6698	3
MEM10 LHS+	**0.23**	**0.08**	**0.21**	**5704**	**1**

RO VOLCANO PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.73	0.06	0.71	23447	10
MEM2 LHS+	0.66	0.01	0.65	19907	9
MEM3 LHS+	0.65	0.00	0.65	15461	8
MEM4 LHS+	0.69	0.30	0.65	13183	7
MEM5 LHS+	0.64	0.00	0.64	10220	6
MEM6 LHS+	0.64	0.00	0.64	8402	2
MEM7 LHS+	0.69	0.36	0.64	8420	3
MEM8 LHS+	0.67	0.18	0.64	9092	4
MEM9 LHS+	0.68	0.30	0.64	9126	5
MEM10 LHS+	**0.68**	**0.26**	**0.64**	**7992**	**1**

RO BRANKE MULTIPEAK PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.30	0.22	0.28	16067	10
MEM2 LHS+	0.25	0.02	0.23	11381	4
MEM3 LHS+	**0.29**	**0.19**	**0.25**	**10555**	**1**
MEM4 LHS+	0.27	0.07	0.27	11202	3
MEM5 LHS+	0.37	0.38	0.27	11768	5
MEM6 LHS+	0.36	0.37	0.27	12038	6
MEM7 LHS+	0.33	0.19	0.27	11198	2
MEM8 LHS+	0.40	0.31	0.27	12270	7
MEM9 LHS+	0.35	0.22	0.27	12784	8
MEM10 LHS+	0.45	0.29	0.28	15987	9

RO MULTIPEAK F1 PROBLEM

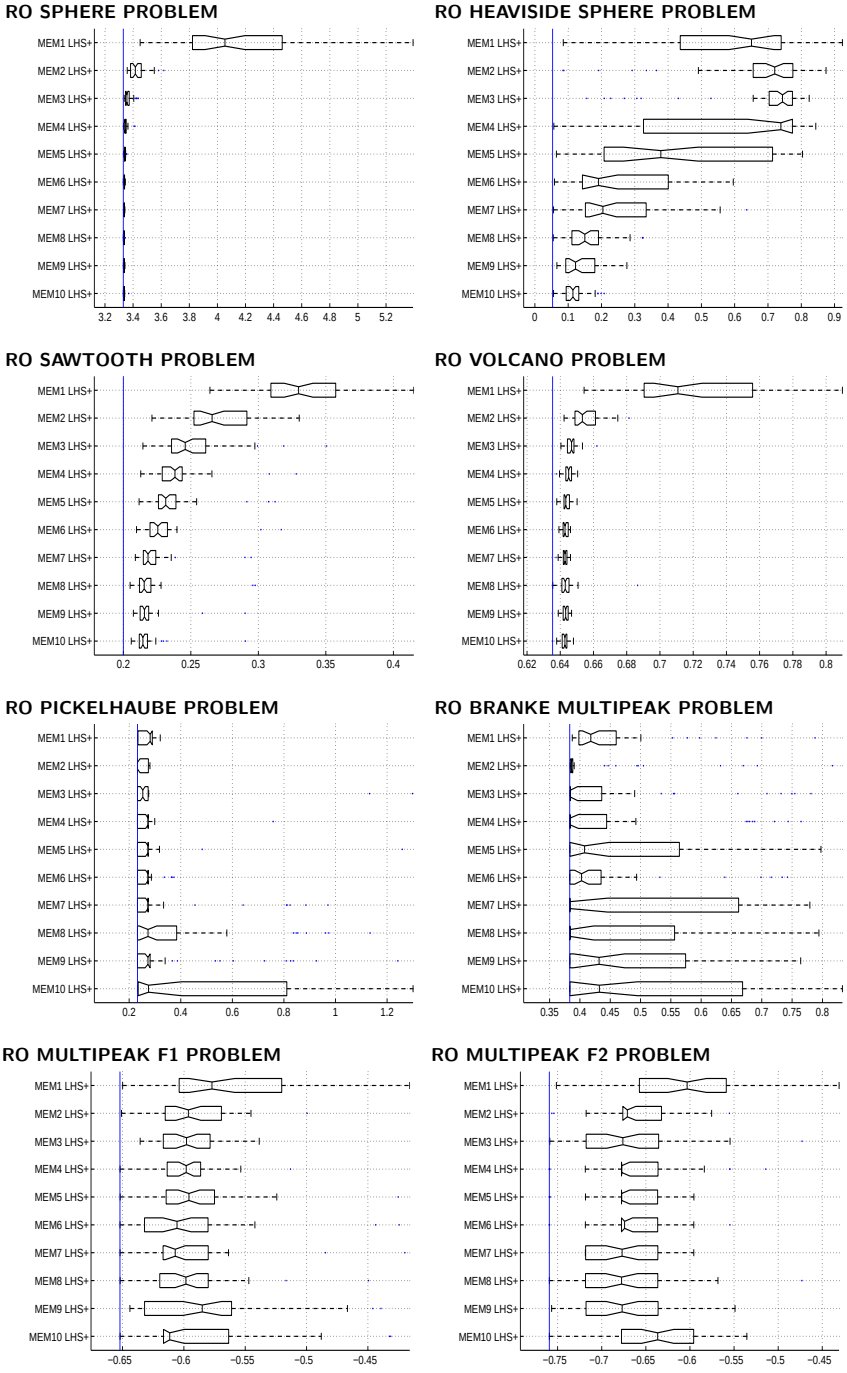
	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	0.45	0.09	0.42	15641	10
MEM2 LHS+	0.42	0.09	0.39	12631	6
MEM3 LHS+	0.45	0.11	0.38	13058	8
MEM4 LHS+	0.45	0.12	0.38	11931	4
MEM5 LHS+	0.48	0.13	0.41	12026	5
MEM6 LHS+	**0.44**	**0.10**	**0.40**	**10595**	**1**
MEM7 LHS+	0.50	0.15	0.38	11928	3
MEM8 LHS+	0.47	0.13	0.38	10956	2
MEM9 LHS+	0.48	0.13	0.43	12888	7
MEM10 LHS+	0.50	0.14	0.43	13596	9

RO MULTIPEAK F2 PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	-0.55	0.08	-0.58	15431	10
MEM2 LHS+	-0.59	0.05	-0.60	12617	7
MEM3 LHS+	-0.59	0.04	-0.60	12144	4
MEM4 LHS+	-0.58	0.07	-0.60	12287	6
MEM5 LHS+	-0.58	0.06	-0.60	13128	8
MEM6 LHS+	**-0.59**	**0.07**	**-0.61**	**10713**	**1**
MEM7 LHS+	-0.58	0.08	-0.61	11500	2
MEM8 LHS+	-0.59	0.06	-0.60	11948	3
MEM9 LHS+	-0.56	0.10	-0.59	13280	9
MEM10 LHS+	-0.57	0.08	-0.61	12202	5

RO FNIM F2 PROBLEM

	Mean	Std	Med	$\sum$#	#
MEM1 LHS+	-0.59	0.11	-0.60	18079	10
MEM2 LHS+	-0.66	0.05	-0.67	13923	9
MEM3 LHS+	-0.65	0.08	-0.68	12630	7
MEM4 LHS+	-0.66	0.05	-0.68	11758	6
MEM5 LHS+	-0.67	0.05	-0.68	10415	2
MEM6 LHS+	-0.64	0.09	-0.67	11753	5
MEM7 LHS+	-0.66	0.05	-0.68	11226	3
MEM8 LHS+	**-0.66**	**0.08**	**-0.68**	**10107**	**1**
MEM9 LHS+	-0.64	0.11	-0.68	11673	4
MEM10 LHS+	-0.64	0.09	-0.64	13686	8


RO SPHERE PROBLEM
RO HEAVISIDE SPHERE PROBLEM
RO SAWTOOTH PROBLEM
RO VOLCANO PROBLEM
RO PICKELHAUBE PROBLEM
RO BRANKE MULTIPEAK PROBLEM
RO MULTIPEAK F1 PROBLEM
RO MULTIPEAK F2 PROBLEM
MEM1 LHS+
MEM2 LHS+
MEM3 LHS+
MEM4 LHS+
MEM5 LHS+
MEM6 LHS+
MEM7 LHS+
MEM8 LHS+
MEM9 LHS+
MEM10 LHS+

8.3 Full Comparison of Schemes Finding Robust Optima

Finally, a full empirical comparison between all schemes for finding robust optima (see Table 8.3) is performed. The $\mathrm{MEM}^{-}_{\mathrm{MS}}$ and the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ schemes are assumed to be near optimally tuned, with the settings as found in Table 8.4 and Table 8.5. The question is: do the advanced evaluation schemes provide yield better results than optimally tuned static schemes?

Approach	Strategy parameters
Myopic	Default
MEM$^{-}_{\mathrm{MS}}$	See Table 8.4
MEM$^{+}_{\mathrm{LHS}}$	See Table 8.5
UH-MEM$^{-}_{\mathrm{MS}}$	$\theta = 0.6$, $\alpha = 1.2$
UH-MEM$^{+}_{\mathrm{LHS}}$	$\theta = 0.6$, $\alpha = 1.2$
ABRSS	Reference set sample size: $m = 2n = 20$
Kriging	Samples for metamodel construction: $n_{\mathrm{krig}} = 2n = 20$. Sampling on the metamodel for approximation of the expected fitness: $\mathrm{MEM50}^{+}_{\mathrm{LHS}}$

Table 8.6: Algorithm settings used for the empirical comparison between the evaluation techniques for finding robust optima.

Section 8.3.1 and Section 8.3.2 show the results for the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES respectively. Table 8.7 shows the combined rank scores of the compared schemes on all benchmark problems.

	$(5/2_{DI}, 35)$-σ**SA-ES**	**CMA-ES**
Myopic	75,157	80,532
MEM$^{-}_{\mathrm{MS}}$	94,463	85,032
MEM$^{+}_{\mathrm{LHS}}$	48,882	49,925
UH-MEM$^{-}_{\mathrm{MS}}$	88,812	83,494
UH-MEM$^{+}_{\mathrm{LHS}}$	**46,923**	**48,684**
ABRSS	66,191	65,102
Kriging	70,972	78,631

Table 8.7: Combined rank sums on the set of test problems.

From the results, we see that the myopic approach outperforms the other approaches on three of the eight test problems. For the sphere problem this is not surprising, being a unimodal test problem where the original optimizer is also the robust optimizer. On the other two problems, Branke's multipeak problem and the pickelhaube problem, which are multimodal problems with emergent robust optimizers, the myopic approach apparently seems to be just as good in targeting the robust peak as the other approaches. Or, the myopic instances seem to be largely

attracted by the robust peaks. For the pickelhaube problem, this is even more surprising given the fact that both peaks have the same basin of attraction areas and cover the same volume.

A possible explanation is the following: The attraction of a peak on an Evolution Strategy is determined by two matters: 1) the probability of generating individuals in that particular peak, and 2) the probability that a random solution of that peak is better than a random solution of the other peaks. For the pickelhaube problem, the probability of generating individuals in both peaks is equal, but the probability that a random solution of the robust peak is better than a random solution of the non-robust peak is larger. This could explain the inherent attraction of the robust peak on the myopic approaches.

On the other problems, however, the myopic approach is clearly not a good alternative. Furthermore, we observe that an optimally tuned $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ evaluation approach yields better results than an optimally tuned $\mathrm{MEM}^{-}_{\mathrm{MS}}$ approach. Also for the uncertainty handling schemes, the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ is to be recommended. When looking at the advanced approaches for finding robust optima, we see that the ABRSS yields remarkably good results on the Heaviside sphere problem and the sawtooth problem, but yields comparably poor results on the other test problems. The Kriging approach, on the other hand, yields good results on the multipeak f1 problem and the multipeak f2 problem, and average results on the other problems. An approach that yields good results across the spectrum is the UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$ approach, which always ranks as one of the three best approaches. The $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ also yields good results across the set of test problems.

To summarize, besides the myopic approach, the $\mathrm{MEM}^{-}_{\mathrm{MS}}$, and the UH-$\mathrm{MEM}^{-}_{\mathrm{MS}}$, all approaches seem to yield comparable results. The ABRSS and the Kriging approach yield particularly good results on specific test problems, the UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$ yields good results across the full set of test problems, and also a well-tuned static $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ seems to work very well, however, slightly worse than the UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$.

8.3.1 Results $(5/2_{DI}, 35)$-σSA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	**5.16**	**9.09**	**3.33**	**1913**	**1**
MEM MS-	5.63	9.02	3.84	14775	7
MEM LHS+	4.59	6.13	3.35	5126	2
UH MS-	3.64	0.15	3.60	12523	6
UH LHS+	4.24	6.21	3.35	5538	3
ABRSS	4.56	7.23	3.49	10593	4
Kriging	5.56	8.43	3.52	10957	5

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.79	0.03	0.78	16075	7
MEM MS-	0.19	0.24	0.12	11193	6
MEM LHS+	0.14	0.26	0.09	9555	4
UH MS-	0.07	0.02	0.07	6324	2
UH LHS+	0.08	0.02	0.07	6520	3
ABRSS	**0.09**	**0.28**	**0.05**	**1721**	**1**
Kriging	0.15	0.24	0.11	10037	5

RO SAWTOOTH PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.55	0.01	0.55	15370	7
MEM MS-	0.27	0.09	0.25	11067	6
MEM LHS+	0.27	0.11	0.23	7484	3
UH MS-	0.26	0.07	0.25	10538	5
UH LHS+	0.24	0.09	0.22	5655	2
ABRSS	**0.21**	**0.06**	**0.20**	**1573**	**1**
Kriging	0.28	0.11	0.24	9738	4

RO VOLCANO PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.72	0.31	0.68	11078	5
MEM MS-	0.82	0.43	0.71	14590	7
MEM LHS+	0.75	0.42	0.65	3989	2
UH MS-	0.86	0.55	0.68	11705	6
UH LHS+	**0.65**	**0.00**	**0.65**	**2709**	**1**
ABRSS	0.77	0.40	0.66	8660	3
Kriging	0.67	0.01	0.67	8694	4

RO PICKELHAUBE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	**0.25**	**0.02**	**0.23**	**4075**	**1**
MEM MS-	0.29	0.18	0.28	12691	7
MEM LHS+	0.28	0.14	0.27	7743	3
UH MS-	0.28	0.13	0.28	10969	6
UH LHS+	0.25	0.02	0.23	6462	2
ABRSS	0.26	0.02	0.27	10026	5
Kriging	0.25	0.02	0.23	9459	4

RO BRANKE MULTIPEAK PROBLEM

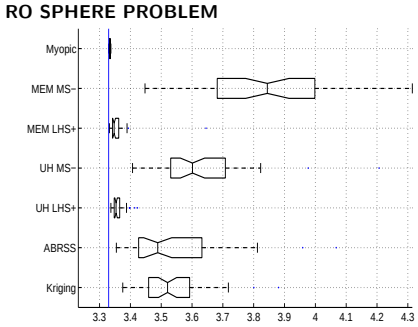
	Mean	Std	Med	∑#	#
Myopic	**0.39**	**0.02**	**0.38**	**3177**	**1**
MEM MS-	0.41	0.02	0.40	13167	7
MEM LHS+	0.38	0.01	0.38	4032	2
UH MS-	0.41	0.03	0.40	13033	6
UH LHS+	0.39	0.04	0.38	6185	3
ABRSS	0.44	0.11	0.40	12109	5
Kriging	0.39	0.02	0.39	9722	4

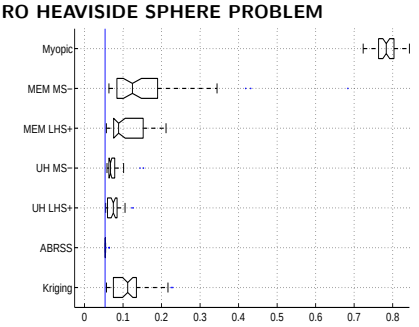
RO MULTIPEAK F1 PROBLEM

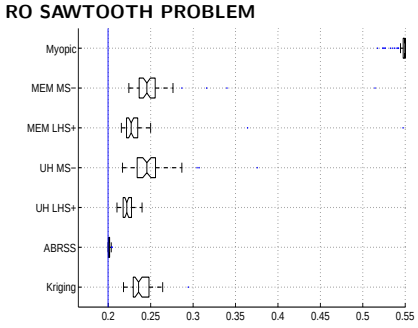
	Mean	Std	Med	∑#	#
Myopic	-0.53	0.09	-0.57	9958	6
MEM MS-	-0.55	0.08	-0.57	9572	4
MEM LHS+	**-0.60**	**0.06**	**-0.62**	**4363**	**1**
UH MS-	-0.46	0.06	-0.47	13553	7
UH LHS+	-0.56	0.09	-0.60	7828	3
ABRSS	-0.51	0.12	-0.57	9926	5
Kriging	-0.59	0.04	-0.60	6225	2

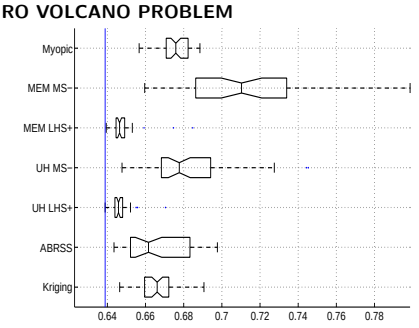
RO MULTIPEAK F2 PROBLEM

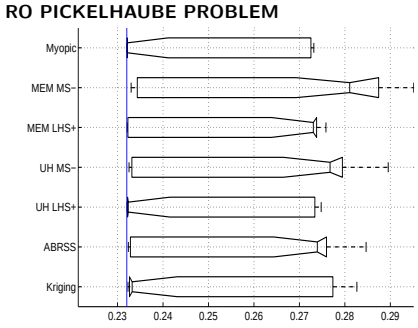
	Mean	Std	Med	∑#	#
Myopic	-0.29	0.17	-0.27	13511	7
MEM MS-	-0.57	0.18	-0.66	7408	4
MEM LHS+	-0.56	0.21	-0.68	6590	3
UH MS-	-0.45	0.14	-0.45	10167	5
UH LHS+	**-0.61**	**0.16**	**-0.67**	**6026**	**1**
ABRSS	-0.38	0.23	-0.30	11583	6
Kriging	-0.63	0.16	-0.67	6140	2

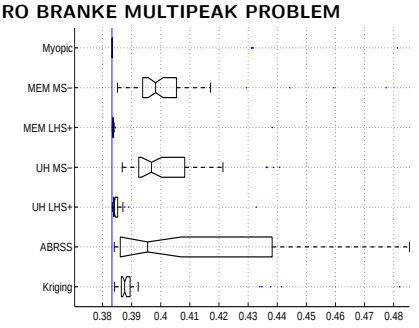

RO SPHERE PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
3.3 3.4 3.5 3.6 3.7 3.8 3.9 4 4.1 4.2 4.3

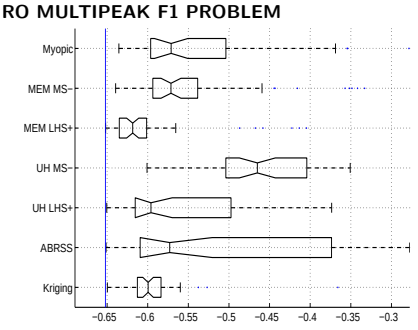

RO HEAVISIDE SPHERE PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8

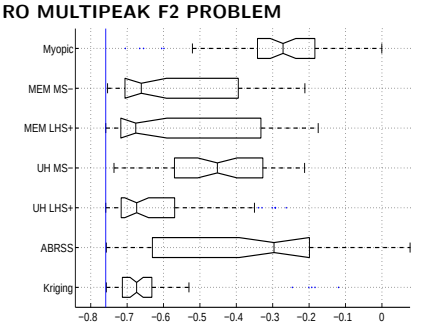

RO SAWTOOTH PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.55


RO VOLCANO PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
0.64 0.66 0.68 0.7 0.72 0.74 0.76 0.78


RO PICKELHAUBE PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
0.23 0.24 0.25 0.26 0.27 0.28 0.29


RO BRANKE MULTIPEAK PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
0.38 0.39 0.4 0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48


RO MULTIPEAK F1 PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
−0.65 −0.6 −0.55 −0.5 −0.45 −0.4 −0.35 −0.3


RO MULTIPEAK F2 PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging
−0.8 −0.7 −0.6 −0.5 −0.4 −0.3 −0.2 −0.1 0

8.3.2 Results CMA-ES

RO SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	**3.33**	**0.00**	**3.33**	**1924**	**1**
MEM MS-	3.61	0.11	3.60	13651	7
MEM LHS+	4.39	7.47	3.34	3690	2
UH MS-	5.80	11.43	3.49	10832	4
UH LHS+	4.45	7.80	3.34	6111	3
ABRSS	3.61	0.16	3.57	12941	6
Kriging	3.55	0.09	3.54	12276	5

RO HEAVISIDE SPHERE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.78	0.03	0.78	16225	7
MEM MS-	0.13	0.28	0.08	9209	4
MEM LHS+	0.14	0.06	0.13	11649	6
UH MS-	0.07	0.01	0.06	6364	3
UH LHS+	0.06	0.01	0.06	5305	2
ABRSS	**0.06**	**0.00**	**0.05**	**1720**	**1**
Kriging	0.11	0.04	0.11	10953	5

RO SAWTOOTH PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.56	0.02	0.55	16182	7
MEM MS-	0.24	0.02	0.23	9923	4
MEM LHS+	0.22	0.02	0.21	5212	2
UH MS-	0.25	0.06	0.24	11216	6
UH LHS+	0.22	0.02	0.21	5514	3
ABRSS	**0.21**	**0.02**	**0.20**	**2267**	**1**
Kriging	0.25	0.05	0.24	11111	5

RO VOLCANO PROBLEM

	Mean	Std	Med	∑#	#
Myopic	0.68	0.01	0.68	14242	7
MEM MS-	0.72	0.31	0.67	12347	6
MEM LHS+	0.70	0.37	0.64	3046	2
UH MS-	0.78	0.47	0.66	9410	3
UH LHS+	**0.64**	**0.00**	**0.64**	**2610**	**1**
ABRSS	0.67	0.02	0.66	9546	4
Kriging	0.67	0.01	0.67	10224	5

RO PICKELHAUBE PROBLEM

	Mean	Std	Med	∑#	#
Myopic	**0.26**	**0.02**	**0.27**	**6018**	**1**
MEM MS-	0.28	0.12	0.23	11225	7
MEM LHS+	0.30	0.22	0.27	7912	3
UH MS-	0.25	0.02	0.23	9392	4
UH LHS+	0.28	0.22	0.23	6287	2
ABRSS	0.29	0.22	0.27	10244	5
Kriging	0.32	0.27	0.27	10347	6

RO BRANKE MULTIPEAK PROBLEM

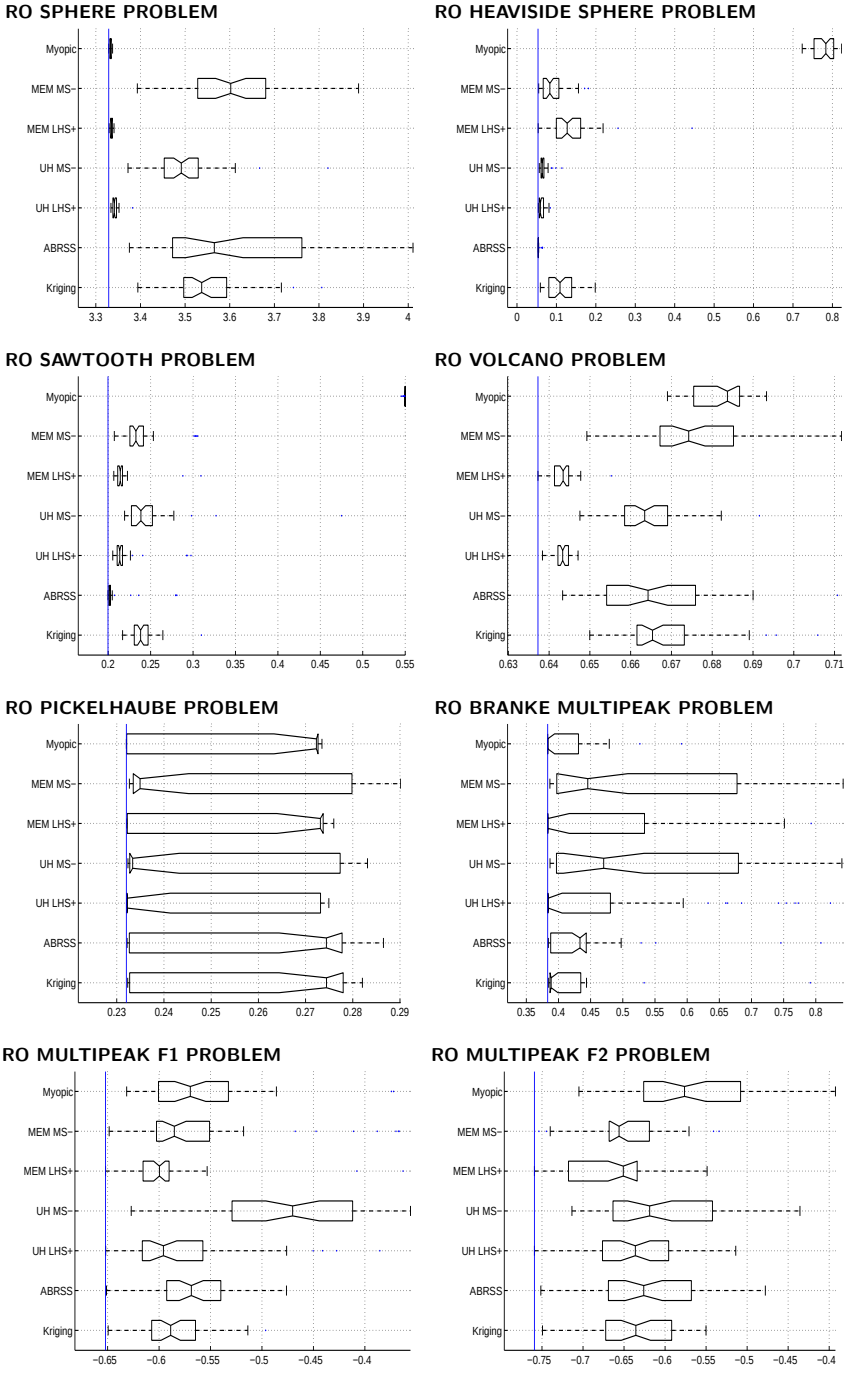
	Mean	Std	Med	∑#	#
Myopic	**0.41**	**0.04**	**0.38**	**4411**	**1**
MEM MS-	0.52	0.15	0.45	12012	6
MEM LHS+	0.47	0.13	0.38	7088	2
UH MS-	0.54	0.15	0.47	12378	7
UH LHS+	0.47	0.13	0.38	7925	4
ABRSS	0.44	0.08	0.43	9804	5
Kriging	0.41	0.06	0.39	7807	3

RO MULTIPEAK F1 PROBLEM

	Mean	Std	Med	∑#	#
Myopic	-0.56	0.05	-0.57	9338	6
MEM MS-	-0.55	0.09	-0.59	8548	4
MEM LHS+	**-0.60**	**0.05**	**-0.60**	**5373**	**1**
UH MS-	-0.48	0.07	-0.47	13621	7
UH LHS+	-0.56	0.10	-0.60	7431	2
ABRSS	-0.56	0.07	-0.57	9186	5
Kriging	-0.57	0.08	-0.59	7928	3

RO MULTIPEAK F2 PROBLEM

	Mean	Std	Med	∑#	#
Myopic	-0.56	0.10	-0.58	12192	7
MEM MS-	-0.62	0.10	-0.66	8117	4
MEM LHS+	**-0.64**	**0.11**	**-0.65**	**5955**	**1**
UH MS-	-0.58	0.12	-0.62	10281	6
UH LHS+	-0.62	0.10	-0.64	7501	2
ABRSS	-0.61	0.08	-0.63	9394	5
Kriging	-0.63	0.09	-0.64	7985	3


RO SPHERE PROBLEM
RO HEAVISIDE SPHERE PROBLEM
RO SAWTOOTH PROBLEM
RO VOLCANO PROBLEM
RO PICKELHAUBE PROBLEM
RO BRANKE MULTIPEAK PROBLEM
RO MULTIPEAK F1 PROBLEM
RO MULTIPEAK F2 PROBLEM
Myopic
MEM MS
MEM LHS+
UH MS
UH LHS+
ABRSS
Kriging

8.4 Summary and Discussion

This chapter has presented the results of an empirical comparison on different techniques that can be used within Evolution Strategies for finding robust optima. From this empirical study we conclude that the myopic approach can be a very risky approach when aiming to find robust optima. It highly depends on the particularities of the objective function landscape, whether this approach will work. However, because the particularities of the objective function landscape are not known beforehand, this approach is not recommendable.

When considering optimally tuned resampling approaches, the results clearly show that the $\mathrm{MEM}^{-}_{\mathrm{MS}}$ is, overall, outperformed by the $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ approach. Hence, it is recommended to use Latin Hypercube Sampling within a resampling approach as well as to use the same sampling disturbances/perturbations for all individuals in the population.

When considering the more advanced methods, we see that the ABRSS and the Kriging approach yield particularly good results on specific test problems, but do not yield exceptionally good performance on other test problems. Of the two, the Kriging approach seems to yield slightly more stable results across all test problems, but ABRSS yields exceptionally good results on the Heaviside sphere and the sawtooth problem.

Two methods that yield a good performance across the full set of test problems are the UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$ approach and a well tuned $\mathrm{MEM}^{+}_{\mathrm{LHS}}$ approach. However, the empirical results in Section 8.2.1 and Section 8.2.2 have shown that the setting of the sample size highly affects the performance of Evolution Strategies for finding robust optima. Therefore, the adaptive averaging approach, the UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$ seems to be a promising method for finding robust optima. Based on these empirical results, we conclude therefore by recommending UH-$\mathrm{MEM}^{+}_{\mathrm{LHS}}$ as the most promising approach when aiming to find robust optima.

Chapter 9

Conclusion

In this work we have presented a definition and a framework of robust optimization that extends the definition of classical optimization and provided practical guidelines for approaching such problems. For two particular scenarios of robust optimization, namely the problem of optimization of noisy objective functions and the problem of finding robust optima, we have studied how Evolution Strategies should be adapted in order to tackle such problems. For these two scenarios we have provided a systematic overview of existing methods and pointed out yet unexplored directions of algorithmic improvement. The algorithmic contributions presented in this work aim to fill these blanks. The empirical comparisons presented in this work provide a refined benchmark set for algorithmic comparison for the two scenarios of robust optimization.

Section 9.1 provides a summary of this work and discusses the conclusions that can be drawn from this research and Section 9.2 closes with an outlook on possible future directions of research.

9.1 Summary

Real-world optimization problems often involve various types of uncertainties and noise emerging in different parts of the optimization problem. When not accounting for these matters, optimization may fail, or may yield solutions that are optimal in the classical strict notion of optimality, but fail in practice. Robust optimization is the practice of optimization that accounts for uncertainties and/or noise in the system or (simulation) model. That is, it considers all types of noise and uncertainties that emerge within the system or (simulation) model, but it does not include uncertainties in the formulation of the goals and constraints.

The goal of robust optimization is twofold: 1) to find optimal solutions despite uncertainties and noise in the optimization model, and 2) to find optimal solutions that are robust with respect to uncertainties and noise, and therewith useful in practice. Dealing with robust optimization problems requires the integration of the notion of robustness in the specification of solution quality. That is, effective objective and constraint functions are needed that incorporate the

notion of robustness or robust quality. This notion changes the original goal of optimization, because robustness and solution quality are often conflicting objectives.

The different sources and types of uncertainty and noise causes a combinatorial explosion of different robust optimization scenarios. However, some scenarios occur more often than others. In this work, two particular robust optimization scenarios are considered: 1) optimization of noisy objective functions, and 2) finding robust optima. These two scenarios frequently emerge in different forms in real-world optimization settings. It is studied how two Evolution Strategy instances, namely the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES, perform in their canonical form on these two scenarios and how they should be adapted to make them more robust.

Optimization of noisy objectives requires a measure for optimization that includes an account for noise, i.e., an effective objective function. When considering the expected quality of candidate solutions as such a measure, Evolution Strategies are fairly insensitive when the noise is relatively small. However, if a higher convergence accuracy is required, additional measures should be taken.

Implicit and explicit averaging provide a straightforward way to increase the convergence accuracy. Implicit averaging refers to the practice of increasing the population size and explicit averaging refers to assessing the quality of candidate solutions by taking the average over multiple evaluations. However, these two techniques require the a priori specification of a sample size or population size and still yield a limited convergence accuracy.

Adaptive averaging techniques are extensions of static noise handling techniques that aim to automatically adapt the evaluation intensity within the process of evolution. These techniques consist of two components: 1) an uncertainty quantification mechanism that measures the effects of noise on the selection operator, and 2) an uncertainty treatment mechanism that involves a static noise handling scheme (such as explicit averaging) and a way to adapt the evaluation intensity (e.g., the sample size) based on the uncertainty quantification.

In this work we consider adaptive averaging schemes that are based on explicit averaging as noise handling method. For a simple quadratic model with Gaussian additive noise it is shown that for an optimally tuned adaptive averaging strategy the resampling effort grows cubically with the inverse distance to the optimum. To achieve a linear convergence rate over the generations, it is thus necessary to at least exponentially increase the resampling effort.

An empirical study shows that an uncertainty measure that is based on rank-differences that emerge when splitting up the evaluation in two rounds is the most promising method for quantifying the uncertainty. Moreover, it is shown that adaptive averaging schemes can yield results comparable to well tuned static noise handling schemes. That is, except for one scenario; a well-tuned implicit averaging scheme for the CMA-ES outperforms all other methods. Being less sensitive to parameter settings, adaptive averaging techniques provide a good alternative to implicit and explicit averaging techniques.

When aiming to find robust optima given anticipated input uncertainty, a number of different effective objective functions can be constructed. Among these, the expected solution quality under consideration of its possible perturbations is a common measure. The expected objective function can be seen as an integral transform of the original objective function. The difficulty of this scenario lies in the fact that precise evaluation of the effective objective function is impossible, hence methods are needed that can approximate the robust quality of candidate solutions. Two types of robust optima can be distinguished: 1) shifted robust optima and 2) emergent robust optima. These types yield two distinct challenges.

The simplest approach for finding robust optima is to do nothing at all, but to rely on the inherent capabilities of Evolution Strategies to target the more robust peaks. This myopic approach is supported by the observation that Evolutionary Algorithms already have an inherent tendency to converge to the more robust parts of the search space. However, it fails when the robust optimizer is a shifted version of the original optimizer.

When actively targeting for robustness, Monte-Carlo integration methods can be used to approximate the expected objective function values for candidate solutions. It is pointed out that doing so yields approximations of the expected objective function value of candidate solutions that makes the problem of finding robust optima very similar to optimization of noisy objective functions. However, in this scenario the noise in the objective function is due to approximation errors and not an inherent part of the system. The limitation of Monte-Carlo integration methods is that they are limited in approximation accuracy and therefore limit the accuracy with which robust optima can be targeted.

Similar to when dealing with noisy objective functions, adaptive noise handling strategies can also be used for finding robust optima. Using this approach has the same advantages as with noisy optimization, namely that it does not suffer from convergence accuracy limitations and it does not require the a priori specification of a sample size.

Another branch of approaches is formed by archiving and metamodeling approaches. These approaches store previously evaluated candidate solutions and use these for estimating the objective function values for newly generated candidate solutions. This makes them especially useful when objective function evaluations are (computationally) expensive. An archive maintenance approach is reviewed that incorporates an advanced scheme to update the archive and to make sure that it is well usable for the obtaining reliable approximations for newly generated candidate solutions. Besides this, a Kriging metamodeling approach is considered and tested that uses the archive not directly, but builds a model from which effective objective function estimates are obtained.

The idea of using niching approaches for the goal of finding robust optima has the alleged advantage that the search focuses on more regions of the search space (which in particular for emergent robust optima looks promising). A straightforward implementation of a standard niching strategy shows, however, that using this idea directly introduces more problems than it solves. For these kinds of purposes, niching strategies are required that can deal with noisy

objective functions.

Last but not least, a method to boost accuracy when aiming to find robust optima is to exploit the overlap of the regions of uncertainty (or η-neighborhoods) of candidate solutions and base the evaluation on how pairs of solutions compare rather than aiming to obtain precise approximations of the effective objective function.

The results of an empirical comparative study show that an adaptive averaging strategy using Latin Hypercube Sampling and using the same disturbances for all individuals in the population for evaluation of the effective objective function is the most promising approach. Compared to the myopic approach and optimally tuned resampling, it yields better performance across the set of test problems. The archive based evaluation approach and the metamodel based approach yield a good performance on specific test problems.

9.2 Outlook

There are many other scenarios that have not been discussed in detail, but which also fall within the scope of robust optimization. Although the observations for the two scenarios considered in this work can be used to a great extent in other scenarios as well, particular dynamics remain to be studied. In particular the scenario of optimization under uncontrollable perturbations of the design variables (see, e.g., [BOS03, SBO04, BS06b]) and the scenario of optimization given uncertainties in the environmental parameters form two interesting classes. Besides this, both scenarios considered in this work can be researched in more depth. For instance, by considering different types of noise distributions or uncertainties and different types of effective objective function measures.

Extending this research to multi-objective optimization is another direction that is useful for many real-world optimization problems. Approaches that have been proposed to find robust optima for multi-objective optimization are presented in, e.g., [JS03, DG06, GA05, LAA05, LOL05, BA06, PL09, Bad10, SRS11]. Linking the observations and findings of this thesis to these studies forms a challenging project. Along the same line of thought, it would interesting to extend this work to robust optimization with constraints.

In this work, the focus was on Evolution Strategies, and in particular the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. Extending this research in the direction of other Evolution Strategy variants or other Evolutionary Algorithms is another logical next step. Besides that, it would be interesting to see to what extent the methods for robust optimization that have been presented in this work can be used within other types of optimization algorithms.

Besides extending this research to different types of robust optimization scenarios, some approaches are well worth to be investigated in more detail, also from the algorithmic perspective.

In Section 5.4.6, an alternative uncertainty quantification approach is proposed. The empirical results of Chapter 6 show that this approach yields comparable results to the

uncertainty quantification measure of Hansen et al. [HNGK09] (see Section 5.4.5). This measure counts the rank-inversions for the uncertainty quantification and compares them with known statistics on purely random orderings. Compared to the uncertainty quantification of Hansen et al. [HNGK09], it has a more stable statistical basis, it is simpler to implement, and yields comparable results. Investigating this uncertainty quantification in more depth is therefore important.

The empirical results of Chapter 6 show that implicit averaging is a very efficient way of countering the effects of noise for the CMA-ES. Studying adaptive averaging techniques that use implicit averaging as noise treatment scheme is therefore a worthwhile object of study.

Maintaining an archive of previously evaluated solutions for the purpose of reusing them for the evaluation of other candidate solutions can enhance the efficiency of optimization when function evaluations are costly. A promising approach for doing so is presented in Section 7.2.7. This approach aims to fill the archive such that it is better spread in the region of interest (i.e., the region around candidate solutions). Also combining this approach in combination with other metamodeling techniques, as shown in Section 7.2.8, is worthwhile being studied.

Also niching for finding robust optima (discussed in Section 7.2.9) is an issue worthwhile of further investigation. Although the straightforward utilization of a standard niching technique does not function well, the idea of focusing the optimization on multiple parts of the search space is interesting. Assuming that the noise in the robustness approximations of the candidate solutions is the main cause of the failure of the niching approach in this context, niching for noisy objective functions is an interesting line of study.

A particularly promising idea is to exploit the overlap in the regions of uncertainty when aiming to find robust optima, presented in Section 7.2.10. For this only a proof of concept is given. Extending this method for efficient comparison among multiple candidate solutions could potentially make this approach very suitable for finding robust optima. However, this remains to be studied.

Finally, this work has provided a benchmark setup for empirical comparison. Based on test problems from literature, two benchmark sets have been defined that are small, representative test sets for 1) optimization of noisy objective functions with Gaussian additive noise, and 2) optimization for finding robust optima given uniform input noise. The test problems are described in detail in Appendices A and B. Although the empirical testing in this thesis is limited to 10-dimensional search spaces, the set of benchmarks is generalizable for arbitrary dimensions. The results of the experiments presented in Chapter 6 and Chapter 8 respectively can be used for comparison of new algorithmic methods. Extending the empirical study to higher dimensional problems is a relatively simple, but is interesting future step.

Bibliography

[AB02] D.V. Arnold and H.-G. Beyer. Performance Analysis of Evolution Strategies with Multi-Recombination in High-Dimensional RN-Search Spaces Disturbed by Noise. *Theoretical Computer Science*, 289:629–647, October 2002.

[AB04] D.V. Arnold and H.-G. Beyer. Performance Analysis of Evolutionary Optimization with Cumulative Step Length Adaptation. *IEEE Transactions on Automatic Control*, 49(4):617–622, April 2004.

[AB08] D.V. Arnold and H.-G. Beyer. Evolution Strategies with Cumulative Step Length Adaptation on the Noisy Parabolic Ridge. *Natural Computing*, 7:555–587, December 2008.

[AMB+92] B. Abraham, J. MacKay, G. Box, R.N. Kacker, T.J. Lorenzen, J.M. Lucas, R.H. Myers, G.G. Vining, J.A. Nelder, M.S. Phadke, J. Sacks, W.J. Welch, A.C. Shoemaker, K.L. Tsui, S. Taguchi, and C.F.J. Wu. Taguchi's Parameter Design: A Panel Discussion. *Technometrics*, 34(2):127–161, May 1992.

[AW93] A.N. Aizawa and B.W. Wah. Dynamic Control of Genetic Algorithms in a Noisy Environment. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 48–55. Morgan Kaufman, 1993.

[AW94] A.N. Aizawa and B.W. Wah. Scheduling of Genetic Algorithms in a Noisy Environment. *Evolutionary Computation*, 2(2):97–122, June 1994.

[BA06] C. Barrico and C. Henggeler Antunes. Robustness Analysis in Multi-Objective Optimization Using a Degree of Robustness Concept. In *IEEE Congress on Evolutionary Computation (CEC 2006)*. IEEE Press, July 2006.

[Bäc96] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[Bad10] J.M. Bader. *Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods*. CreateSpace, 2010.

[BBC11] D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and Applications of Robust Optimization. *SIAM Review*, 53(3):464–501, August 2011.

[Bey94] H.-G. Beyer. Toward a Theory of Evolution Strategies: The (μ, λ)-Theory. *Evolutionary Computation*, 2:381–407, December 1994.

[Bey95] H.-G. Beyer. Towards a Theory of Evolution Strategies: On the Benefits of Sex — The (μ/ρ, λ) Theory. *Evolutionary Computation*, 3:81–111, March 1995.

[Bey96] H.-G. Beyer. On the Asymptotic Behavior of Multi-Recombinant Evolution Strategies. In *Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *LNCS*, pages 122–133. Springer, 1996.

[Bey98] H.-G. Beyer. Mutate Large, but Inherit Small! On the Analysis of Rescaled Mutations in (1,λ)-ES with Noisy Fitness Data. In *Parallel Problem Solving from Nature (PPSN V)*, volume 1498 of *LNCS*, pages 109–118. Springer-Verlag, 1998.

[Bey00] H.-G. Beyer. Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):239–267, June 2000.

[BH94] T. Bäck and U. Hammel. Evolution Strategies Applied to Perturbed Objective Functions. In *IEEE Congress on Evolutionary Computation (CEC 1994)*, pages 40–45. IEEE Press, 1994.

[BNT10] D. Bertsimas, O. Nohadani, and K.M. Teo. Robust Optimization for Unconstrained Simulation-Based Problems. *Operations Research*, 58(1):161–178, January 2010.

[BOS03] H.-G. Beyer, M. Olhofer, and B. Sendhoff. On the Behavior of $(\mu/\mu_I, \lambda)$-ES Optimizing Functions Disturbed by Generalized Noise. In *Foundations of Genetic Algorithms 7*, pages 307–328. Morgan Kaufmann, 2003.

[BP09] B. Burgstaller and F. Pillichshammer. The Average Distance Between Two Points. *Bulletin of the Australian Mathematical Society*, 80(3):353–359, May 2009.

[Bra98] J. Branke. Creating Robust Solutions by Means of Evolutionary Algorithms. In *Parallel Problem Solving from Nature (PPSN V)*, volume 1498 of *LNCS*, pages 119–128. Springer-Verlag, 1998.

[Bra01] J. Branke. Reducing the Sampling Variance when Searching for Robust Solutions. In *Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 235–242. Morgan Kaufmann, 2001.

[BS02] H.-G. Beyer and H.-P. Schwefel. Evolution Strategies A Comprehensive Introduction. *Natural Computing*, 1:3–52, May 2002.

[BS06a] H.-G. Beyer and B. Sendhoff. Evolution Strategies for Robust Optimization. In *IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 1346–1353. IEEE Press, 2006.

[BS06b] H.-G. Beyer and B. Sendhoff. Functions with Noise-induced Multi-Modality: A Test for Evolutionary Robust Optimization — Properties and Performance Analysis. *IEEE Transactions on Evolutionary Computation*, 10(5):507–526, October 2006.

[BS07] H.-G. Beyer and B. Sendhoff. Robust optimization — A Comprehensive Survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, July 2007.

[BSS01] J. Branke, C. Schmidt, and H. Schmeck. Efficient Fitness Estimation in Noisy Environments. In *Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 243–250, 2001.

[BST+88] G. Box, A.C. Shoemaker, K.-L. Tsui, R.V. León, W.C. Parr, V.N. Nair, D. Pregibon, R.J. Carroll, D. Ruppert, B. Gunter, and N.R. Ullman. Signal-To-Noise Ratios, Performance Criteria, and Transformations. *Technometrics*, 30(1):1–40, February 1988.

[BTGGN04] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable Robust Solutions of Uncertain Linear Programs. *Mathematical Programming*, 99(2):351–376, March 2004.

[BZ70] R.E. Bellman and L.A. Zadeh. Decision-Making in a Fuzzy Environment. *Management Science*, 17(4), December 1970.

[CG92] G. Casella and E.I. George. Explaining the Gibbs Sampler. *The American Statistician*, 46(3):167–174, August 1992.

[Cox05] D.R. Cox. Frequentist and Bayesian Statistics: A Critique (Keynote Address). In *Statistical Problems in Particle Physics, Astrophysics and Cosmology: Proceedings of PHYSTAT05*, pages 3–6. Imperial College Press, September 2005.

[CP04] E. Cantú-Paz. Adaptive Sampling for Noisy Problems. In *Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 947–958. ACM, 2004.

[CWZ99] W. Chen, M.M. Wiecek, and J. Zhang. Quality Utility — A Compromise Programming Approach to Robust Design. *Journal of Mechanical Design*, 121(2):179–187, June 1999.

[Dan55] G.B. Dantzig. Linear Programming Under Uncertainty. *Management Science*, 1(3-4):197–206, April 1955.

[DG06] K. Deb and H. Gupta. Introducing Robustness in Multi-Objective Optimization. *Evolutionary Computation*, 14(4):463–494, December 2006.

[Dun61] O.J. Dunn. Multiple Comparisons Among Means. *Journal of the American Statistical Association*, 56(293):52–64, March 1961.

[EH02] I.C. Parmee (Editor) and P. Hajela. *Optimization in Industry*. Springer, 2002.

[EHM+08] M.T.M. Emmerich, C.J. Hopfe, R. Marijt, J. Hensen, C. Struck, and P. Stoelinga. Evaluating Optimization Methodologies for Fture Integration in Building Performance Tools. In *Proceedings of the 8th International Conference on Adaptive Computing on Design and Manufacture (ACDM 2008)*, 2008.

[ES03] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[FG88] J.M. Fitzpatrick and J.J. Grefenstette. Genetic Algorithms in Noisy Environments. *Machine Learning*, 3:101–120, October 1988.

[FSK08] A. Forrester, A. Sobester, and A. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, September 2008.

[GA05] S. Gunawan and S. Azarm. Multi-Objective Robust Optimization Using a Sensitivity Region Concept. *Structural and Multidisciplinary Optimization*, 29(1):50–60, January 2005.

[GR87] D.E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic algorithms and their Application*, pages 41–49. L. Erlbaum Associates Inc., 1987.

[GW92] W.R. Gilks and P. Wild. Adaptive Rejection Sampling for Gibbs Sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348, 1992.

[HB94] U. Hammel and T. Bäck. Evolution Strategies on Noisy Functions, How to Improve Convergence Properties. In *Parallel Problem Solving from Nature (PPSN III)*, volume 866 of *LNCS*, pages 159–168. Springer, 1994.

[HFRA09a] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009.

[HFRA09b] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noisy Functions Definitions. Research Report RR-6869, INRIA, 2009.

[HFRA10] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noisy Functions. Technical report, INRIA, 2010.

[HHPW07] C.J. Hopfe, J.L.M. Hensen, W. Plokker, and A.J.T.M. Wijsman. Model Uncertainty and Sensitivity Analysis for Thermal Comfort Prediction. In *Proceedings of the 12th Symposium for Building Physics*, pages 103–112. Technische Universität Dresden, March 2007.

[HM11] V. Heidrich-Meisner. *Evolutionary Direct Policy Search in Noisy Environments*. PhD thesis, Fakultät für Physik und Astronomie der Ruhr-Universität Bochum, 2011.

[HMI09a] V. Heidrich-Meisner and C. Igel. Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 401–408. ACM, 2009.

[HMI09b] V. Heidrich-Meisner and C. Igel. Uncertainty Handling CMA-ES for Reinforcement Learning. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1211–1218. ACM, 2009.

[HNGK09] N. Hansen, A.S.P. Niederberger, L. Guzzella, and P. Koumoutsakos. A Method for Handling Uncertainty in Evolutionary Optimization with an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, February 2009.

[HO96] N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *IEEE Congress on Evolutionary Computation (CEC 1996)*, pages 312–317. IEEE Press, 1996.

[HO01] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, June 2001.

[Hop09] C.J. Hopfe. *Uncertainty and Sensitivity Analysis in Building Performance Simulation for Decision Support and Design Optimization*. PhD thesis, Eindhoven University of Technology, 2009.

[HTHB11] P. Hoes, M. Trcka, J.L.M. Hensen, and B. Hoekstra Bonnema. Optimizing Building Designs Using a Robustness Indicator with respect to User Behavior. In *Building Simulation 2011 — Proceedings of the 12th Conference of the International Building Performance Simulation Association*, pages 1710–1717, 2011.

[IR00] M. Inuiguchi and J. Ramík. Possibilistic Linear Programming: A Brief Review of Fuzzy Mathematical Programming and a Comparison with Stochastic Programming in Portfolio Selection Problem. *Fuzzy Sets and Systems*, 111(1):3–28, April 2000.

[JB05] Y. Jin and J. Branke. Evolutionary Optimizationin Uncertain Environments — A Survey. *IEEE Transaction on Evolutionary Computation*, 9(3):303–317, June 2005.

[Jin05] Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing*, 9:3–12, January 2005.

[JL02] D.H. Jung and B.C. Lee. Development of a Simple and Efficient Method for Robust Optimization. *International Journal for Numerical Methods in Engineering*, 53(9):2201–2215, March 2002.

[JS03] Y. Jin and B. Sendhoff. Trade-off Between Performance and Robustness: An Evolutionary Multiobjective Approach. In *Proceedings of the Second International Conference on Evolutionary Multicriterion Optimization*, pages 237–251. Springer-Verlag, 2003.

[JSW98] 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, December 1998.

[KAE+09] J.W. Kruisselbrink, A. Aleman, M.T.M. Emmerich, A.P. IJzerman, A. Bender, T.H.W. Bäck, and E. van der Horst. Enhancing Search Space Diversity in Multi-Objective Evolutionary Drug Molecule Design using Niching. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 217–224. ACM, 2009.

[KBIvdH08] J.W. Kruisselbrink, T.H.W. Bäck, A.P. IJzerman, and E. van der Horst. Evolutionary Algorithms for Automated Drug Design Towards Target Molecule Properties. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1555–1562. ACM, 2008.

[KD09] A. Der Kiureghian and O. Ditlevsen. Aleatory or Epistemic? Does it Matter? *Structural Safety*, 31(2):105–112, March 2009.

[KEB09a] J.W. Kruisselbrink, M.T.M. Emmerich, and T.H.W. Bäck. On the Limitations of Adaptive Resampling Using the Student's t-Test in Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 2649–2656. ACM, 2009.

[KEB+09b] J.W. Kruisselbrink, M.T.M. Emmerich, T.H.W. Bäck, A.P. IJzerman, and E. van der Horst. Combining Aggregation with Pareto Optimization: A Case Study in Evolutionary Molecular Design. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)*, volume 5467 of *LNCS*, pages 453–467. Springer-Verlag, 2009.

[KEB10] J.W. Kruisselbrink, M.T.M. Emmerich, and T. Bäck. An Archive Maintenance Scheme for Finding Robust Solutions. In *Parallel Problem Solving from Nature (PPSN XI)*, volume 6238 of *LNCS*, pages 214–223. Springer-Verlag, 2010.

[KEDB10a] J.W. Kruisselbrink, M.T.M. Emmerich, A.H. Deutz, and T. Bäck. A Robust Optimization Approach using Kriging Metamodels for Robustness Approximation in the CMA-ES. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8, 2010.

[KEDB10b] J.W. Kruisselbrink, M.T.M. Emmerich, A.H. Deutz, and T. Bäck. Exploiting Overlap when Searching for Robust Optima. In *Parallel Problem Solving from Nature (PPSN XI)*, volume 6238 of *LNCS*, pages 63–72. Springer-Verlag, 2010.

[Kol33] A.N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, 1933.

[KRD+11] J.W. Kruisselbrink, E. Reehuis, A.H. Deutz, T. Bäck, and M.T.M. Emmerich. Using the Uncertainty Handling CMA-ES for Finding Robust Optima. In

Genetic and Evolutionary Computation Conference (GECCO 2011). ACM, 2011.

[KW94] P. Kall and S.W. Wallace. *Stochastic Programming*. John Wiley, 1994.

[KYG04] P.N. Koch, R.-J. Yang, and L. Gu. Design for Six Sigma Through Robust Optimization. *Structural and Multidisciplinary Optimization*, 26(3-4):235–248, January 2004.

[LAA05] M. Li, S. Azarm, and V. Aute. A Multi-Objective Genetic Algorithm for Robust Design Optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 771–778. ACM, 2005.

[Li09] R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. PhD thesis, University of Leiden, 2009.

[LLDF01] C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney. Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Developments Settings. *Advanced Drug Delivery Reviews*, 46(1-3):3–26, March 2001.

[LOL05] D. Lim, Y.-S. Ong, and B.-S. Lee. Inverse Multi-Objective Robust Evolutionary Design Optimization in the Presence of Uncertainty. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 55–62. ACM, 2005.

[Mar01] B.H. Margolius. Permutations with Inversions. *Journal of Integer Sequences*, 4(2):1–4, 2001.

[MBC00] M.D. McKay, R.J. Beckman, and W.J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 42(1):55–61, February 2000.

[MG96] B.L. Miller and D.E. Goldberg. Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. *Evolutionary Computation*, 4:113–131, June 1996.

[MM94] O. Maron and A.W. Moore. Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In *Advances in Neural Information Processing Systems 6*, pages 59–66. Morgan Kaufmann, 1994.

[MM97] O. Maron and A.W. Moore. The Racing Algorithm: Model Selection for Lazy Learners. *Artificial Intelligence Review*, 11(1-5):193–225, February 1997.

[MMA+01] S. Markon, O. Markon, D.V. Arnold, T. Bäck, T. Beielstein, and H.-G. Beyer. Thresholding — A Selection Operator for Noisy ES. In *Congress on Evolutionary Computation (CEC'01)*, pages 465–472. IEEE Press, 2001.

[MNB08] S. Meyer-Nieberg and H.-G. Beyer. Why Noise May Be Good: Additive Noise on the Sharp Ridge. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 511–518, 2008.

[MVZ95] J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust Optimization of Large-Scale Systems. *Operations Research*, 43(2):264–281, March/April 1995.

[NAP09] C.A. Nicolaou, J. Apostolakis, and C.S. Pattichis. De Novo Drug Design Using Multiobjective Evolutionary Graphs. *Journal of Chemical Information and Modeling*, 49(2):295–307, January 2009.

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Operations Research. Springer, 2006.

[O'H04] T. O'Hagan. Dicing with the Unknown. *Significance*, 1(3):132–133, October 2004.

[ONL06] Y.-S. Ong, P.B. Nair, and K.Y. Lum. Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, August 2006.

[Par07] G.-J. Park. *Analytic Methods for Design Practice*. Springer-Verlag, 2007.

[PBJ06] I. Paenke, J. Branke, and Y. Jin. Efficient Search for Robust Solutions by Means of Evolutionary Algorithms and Fitness Approximation. *IEEE Transactions on Evolutionary Computation*, 10(4):405–420, August 2006.

[PC96] M.E. Paté-Cornell. Uncertainties in Risk Analysis: Six Levels of Treatment. *Reliability Engineering & System Safety*, 54(2-3):95–111, November 1996.

[Pha89] M.S. Phadke. *Quality Engineering using Robust Design*. Prentice Hall, 1989.

[PL09] S. Poles and A. Lovison. A Polynomial Chaos Approach to Robust Multiobjective Optimization. In *Hybrid and Robust Approaches to Multiobjective Optimization*, number 09041 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Germany, 2009.

[PLBG07] S. Pannier, M. Liebscher, M. Beer, and W. Graf. Fuzzy Stochastic Simulation of Deep Drawing Processes. In *EUROMECH Colloquium 482 — Efficient Methods for Robust Design and Optimisation*, 2007.

[Rec73] I. Rechenberg. *Evolutionsstrategie Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Friedrich Frommann Verlag, 1973.

[Rec94] I. Rechenberg. *Evolutionstrategie '94*. Frommann-Holzboog, 1994.

[RKD+11] E. Reehuis, J.W. Kruisselbrink, A.H. Deutz, T. Bäck, and M.T.M. Emmerich. Multiobjective Optimization of Water Distribution Networks using SMS-EMOA. In *Proceedings of EUROGEN 2011*, pages 269–279. CIRA, 2011.

[Rud01] G. Rudolph. A Partial Order Approach to Noisy Fitness Functions. In *IEEE Congress on Evolutionary Computation (CEC 2001)*, pages 318–325. IEEE Press, 2001.

[Sah04] N.V. Sahinidis. Optimization under Uncertainty: State-of-the-Art and Opportunities. *Computers and Chemical Engineering*, 28(6-7):971–983, June 2004.

[SB09] O.M. Shir and T. Bäck. Niching with Derandomized Evolution Strategies in Artificial and Real-World Landscapes. *Natural Computing*, 8(1):171–196, March 2009.

[SBO04] B. Sendhoff, H.-G. Beyer, and M. Olhofer. The Influence of Stochastic Quality Functions on Evolutionary Search. In *Recent Advances in Simulated Evolution and Learning*, Advances in Natural Computation, pages 152–172. World Scientific, 2004.

[Sch77] H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen Mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, 1977.

[SH04] O. Schenk and M. Hillmann. Optimal Design of Metal Forming die Surfaces with Evolution Strategies. *Computers and Structures*, 82(20-21):1695–1705, August 2004.

[SHL+05] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real Parameter Optimization. Technical report, Nanyang Technological University, 2005.

[SJ08] G.I. Schuëller and H.A. Jensen. Computational Methods in Optimization Considering Uncertainties — An Overview. *Computer Methods in Applied Mechanics and Engineering*, 198(1):2–13, May 2008.

[SK00] Y. Sano and H. Kita. Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search. In *Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *LNCS*, pages 571–580. Springer, 2000.

[SK02] Y. Sano and H. Kita. Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search with Test of Estimation. In *IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 360–365. IEEE Press, 2002.

[SPS98] N.J. Samsatli, L.G. Papageorgiou, and N. Shah. Robustness Metrics for Dynamic Optimization Models Under Parameter Uncertainty. *AIChE Journal*, 44(9):1993–2006, April 1998.

[SRS11] A. Saha, T. Ray, and W. Smith. Towards Practical Evolutionary Robust Multi-Objective Optimization. In *IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 2123–2130, 2011.

[Sta98] P. Stagge. Averaging Efficiently in the Presence of Noise. In *Parallel Problem Solving from Nature (PPSN V)*, volume 1498 of *LNCS*, pages 188–200. Springer-Verlag, 1998.

[SWMW89] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–423, November 1989.

[TA84] H. Tanaka and K. Asai. Fuzzy Linear Programming with Fuzzy Numbers. *Fuzzy Sets and Systems*, 13(1):1–10, May 1984.

[Tag78] G. Taguchi. Performance Analysis Design. *International Journal of Production Research*, 16(6):521–530, 1978.

[Tag86] G. Taguchi. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. Quality Resources (Asian Productivity Organisation), 1986.

[Tag89] G. Taguchi. *Introduction to Quality Engineering*. American Supplier Institute, 1989.

[TAW03] M.W. Trosset, N.M. Alexandrov, and L.T. Watson. New Methods for Robust Design Using Computer Simulations. In *Proceedings of the Section on Physical and Engineering Science*, pages 4287–4291. American Statistical Association, 2003.

[TG97] S. Tsutsui and A. Ghosh. Genetic Algorithms with a Robust Solution Searching Scheme. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, September 1997.

[TG03] S. Tsutsui and A. Ghosh. Effects of Adding Perturbations to Phenotypic Parameters in Genetic Algorithms for Searching Robust Solutions. In *Advances in Evolutionary Computing: Theory and Applications*, pages 351–365. Springer-Verlag, 2003.

[TGF96] S. Tsutsui, A. Ghosh, and Y. Fujimoto. A Robust Solution Searching Scheme in Genetic Search. In *Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *LNCS*, pages 543–552. Springer-Verlag, 1996.

[TOA74] H. Tanaka, T. Okuda, and K. Asai. On Fuzzy Mathematical Programming. *Journal of Cybernetics*, 3(4):37–46, 1974.

[Tob70] W.R. Tobler. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, 46(2):234–240, June 1970.

[Tro97] M.W. Trosset. Taguchi and Robust Optimization. Technical report 96–31, Department of Computational & Applied Mathematics, Rice University, March 1997.

[Tsu99] S. Tsutsui. A Comparative Study on the Effects of Adding Perturbations to Phenotypic Parameters in Genetic Algorithms with a Robust Solution Searching Scheme. In *IEEE Systems, Man, and Cybernetics Conference (SMC 1999)*, pages 585–591, 1999.

[TZ89] A. A. Törn and A. Zilinskas. *Global Optimization*, volume 350 of *LNCS*. Springer, 1989.

[Wet66] R.J.B. Wets. Programming Under Uncertainty: The Equivalent Convex Program. *SIAM Journal on Applied Mathematics*, 14(1):89–105, January 1966.

[WHB98] D. Wiesmann, U. Hammel, and T. Bäck. Robust Design of Multilayer Optical Coatings by means of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 2(4):162–167, November 1998.

[Zad65] L.A. Zadeh. Fuzzy Sets. *Information Control*, 8(3):338–353, June 1965.

[Zay96] A.I. Zayed. *Handbook of Function and Generalized Function Transformations*. Mathematical Sciences Reference Series. CRC Press, 1996.

[Zim76] H.J. Zimmermann. Description and Optimization of Fuzzy Systems. *International Journal of General Systems*, 2(4):209–215, 1976.

Appendix A

Test Problems for Noisy Optimization

A.1 Noisy Sphere Problem

The noisy sphere is a simple scalable test function for studying optimization of noisy real-valued objective functions using Evolution Strategies. It reads

$$f(\mathbf{x}) = \sum_{i=1}^{n} z_i^2, \tag{A.1}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*, \tag{A.2}$$

with $\mathbf{x}^* \in \mathbb{R}^n$ being the location of the optimum. The noisy sphere function, reads:

$$f(\mathbf{x}) = \sum_{i=1}^{n} z_i^2 + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.3}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.4}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 1$. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-5, \ldots, -5]$ and $\mathbf{x}_\mathrm{u} = [5, \ldots, 5]$.

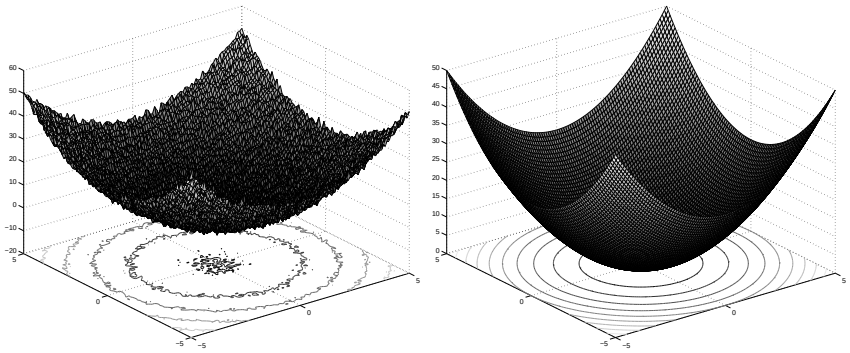


Figure A.1: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.2 Noisy Ellipsoid Problem

The ellipsoid function is a transformed (stretched and rotated) version of the sphere function.

$$f(\mathbf{x}) = \sum_{i=1}^{n} z_i^2, \tag{A.5}$$

$$\mathbf{z} = \mathbf{RD}\left(\mathbf{x} - \mathbf{x}^*\right). \tag{A.6}$$

with $\mathbf{x}^* \in \mathbb{R}^n$ being the location of the optimum, $\mathbf{R}$ being a rotation matrix, and $\mathbf{D}$ being a diagonal (scaling) matrix. The noisy ellipsoid function reads:

$$f(\mathbf{x}) = \sum_{i=1}^{n} z_i^2 + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.7}$$

$$\mathbf{z} = \mathbf{RD}\left(\mathbf{x} - \mathbf{x}^*\right). \tag{A.8}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 2$. The rotation matrix $\mathbf{R}$ is generated from normally distributed entries and scaling matrix $\mathbf{D}$ has a condition number of 10 with equally spaced eigenvalues. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-1, \ldots, -1]$ and $\mathbf{x}_\mathrm{u} = [1, \ldots, 1]$.

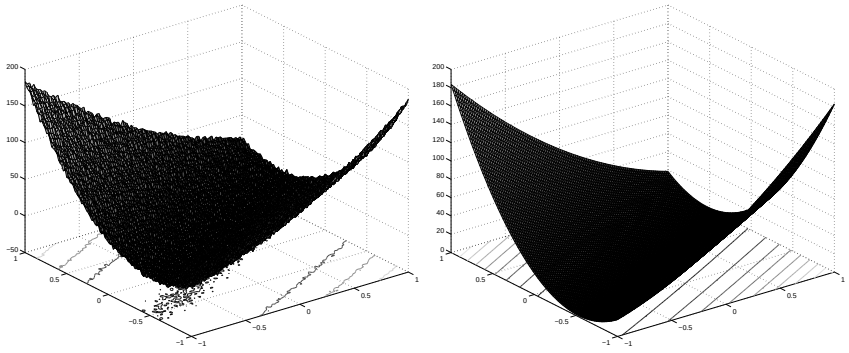


Figure A.2: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.3 Noisy Step Ellipsoid Problem

The step ellipsoid function is an ellipsoid function transformed such that it has a non-steady surface consisting of plateaus. It reads

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lfloor z_i \rfloor^2, \tag{A.9}$$

$$\mathbf{z} = \mathbf{RD}\left(\mathbf{x} - \mathbf{x}^*\right). \tag{A.10}$$

Here $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum, $\mathbf{R}$ is a rotation matrix, and $\mathbf{D}$ is a diagonal (scaling) matrix. The noisy step ellipsoid function reads

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lfloor z_i \rfloor^2 + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.11}$$

$$\mathbf{z} = \mathbf{RD}\left(\mathbf{x} - \mathbf{x}^*\right). \tag{A.12}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 2$. The rotation matrix $\mathbf{R}$ is generated from normally distributed entries and scaling matrix $\mathbf{D}$ has a condition number of 10 with equally spaced eigenvalues (these matrices are the same as for the noisy ellipsoid function). Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-1, \ldots, -1]$ and $\mathbf{x}_\mathrm{u} = [1, \ldots, 1]$.

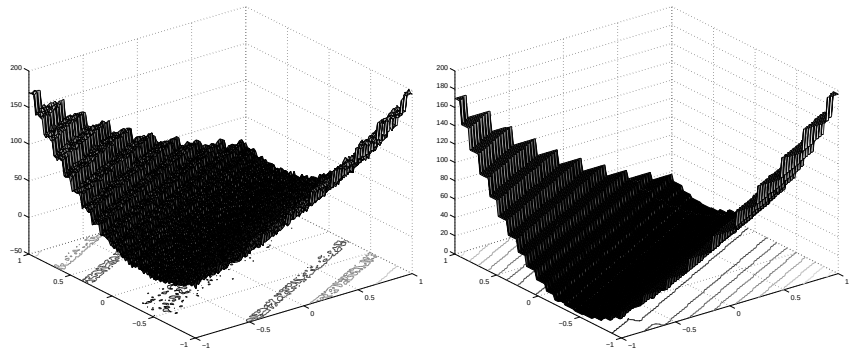


Figure A.3: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.4 Noisy Rosenbrock Problem

The Rosenbrock function is a unimodal test function for real-valued optimization. The particular characteristic of this function is that its fitness landscape shows a bent valley that leads towards the optimum and the direction of the steepest descent changes continuously when nearing the optimum. It is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right), \tag{A.13}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{x}^*) + 1. \tag{A.14}$$

Here $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum. The noisy Rosenbrock function reads

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.15}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{x}^*) + 1. \tag{A.16}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 2$. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-2, \ldots, -2]$ and $\mathbf{x}_\mathrm{u} = [2, \ldots, 2]$.

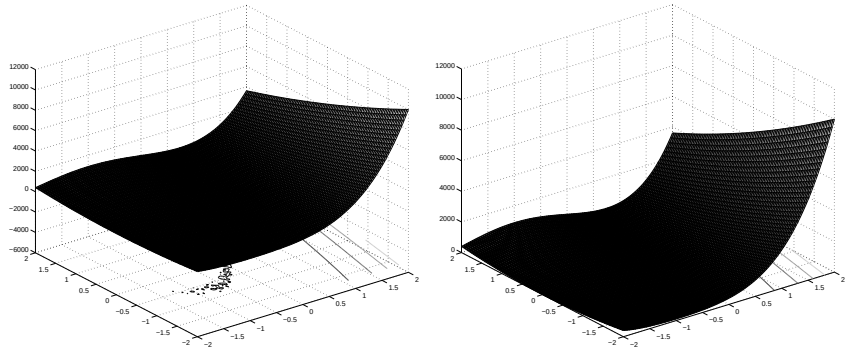


Figure A.4: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.5 Noisy Ackley Problem

The Ackley function is a multi-modal test function, defined as

$$\begin{aligned} f(\mathbf{x}) &= -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(c_3 \cdot z_i)\right) \\ &\quad + c_1 + \exp(1), \end{aligned} \tag{A.17}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.18}$$

Here, $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2\pi$, and $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum. The noisy Ackley function reads

$$\begin{aligned} f(\mathbf{x}) &= -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(c_3 \cdot z_i)\right) \\ &\quad + c_1 + \exp(1) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \end{aligned} \tag{A.19}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.20}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 1$. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-5, \ldots, -5]$ and $\mathbf{x}_\mathrm{u} = [5, \ldots, 5]$.

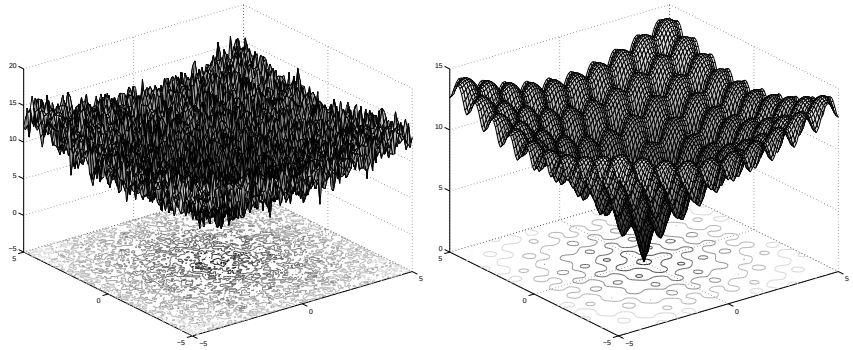


Figure A.5: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.6 Noisy Griewank Problem

The Griewank function is a multi-modal test function, defined as

$$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right), \tag{A.21}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.22}$$

Here $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum. The noisy Griewank function reads

$$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.23}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.24}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 0.5$. Unless stated otherwise, we will assume stationary noise. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-60, \ldots, -60]$ and $\mathbf{x}_\mathrm{u} = [60, \ldots, 60]$.

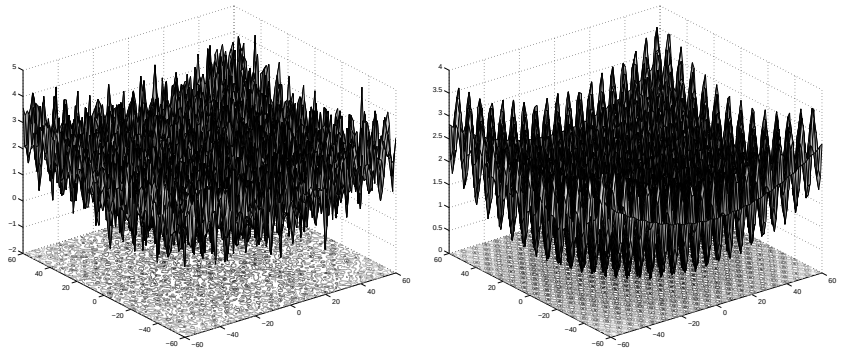


Figure A.6: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.7 Noisy Rastrigin Problem

The Rastrigin function is a multi-modal test function, defined as

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi \cdot z_i)), \tag{A.25}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.26}$$

Here $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum. The noisy Griewank function reads

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi \cdot z_i)) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.27}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.28}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 2$. Unless stated otherwise, we will assume stationary noise. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-5, \ldots, -5]$ and $\mathbf{x}_\mathrm{u} = [5, \ldots, 5]$.

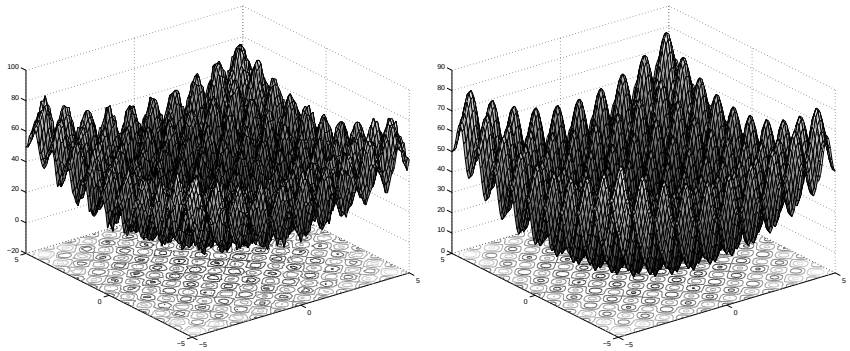


Figure A.7: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.8 Noisy Schaffer's F7 Problem

Schaffer's F7 function is a multi-modal test function, defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (z_i^2 + z_{i+1}^2)^{0.25} (\sin^2(50(z_i^2 + z_{i+1}^2)^{0.1}) + 1), \tag{A.29}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.30}$$

Here $\mathbf{x}^* \in \mathbb{R}^n$ is the location of the optimum. The noisy Griewank function reads

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (z_i^2 + z_{i+1}^2)^{0.25} (\sin^2(50(z_i^2 + z_{i+1}^2)^{0.1}) + 1) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.31}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{x}^*. \tag{A.32}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 1$. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-5, \ldots, -5]$ and $\mathbf{x}_\mathrm{u} = [5, \ldots, 5]$.

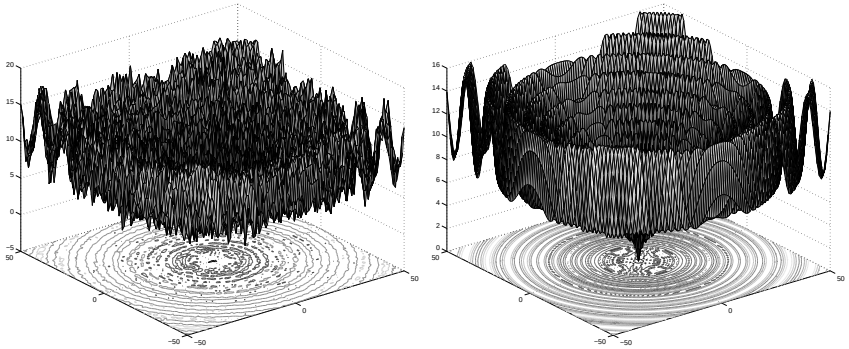


Figure A.8: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.9 Noisy Branke's Multipeak Problem

Branke's Multipeak function is a multi-modal test function with 2^n peaks, defined as

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i-1}^{n}\left(c - g(x_i)\right), \tag{A.33}$$

$$g(x_i) = \begin{cases} -\left(x_i+1\right)^2 + 1 & \text{if } -2 \leq x_i < 0 \\ c \cdot 2^{-8|x_i-1|} & \text{if } 0 \leq x_i \leq 2 \\ 0 & \text{otherwise} \end{cases}. \tag{A.34}$$

Here $c = 1.3$ and the global optimum is located at $\mathbf{x} = [1, \ldots, 1]$. The noisy version of Branke's multipeak function reads

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i-1}^{n}\left(c - g(x_i)\right) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.35}$$

$$g(x) = \begin{cases} -\left(x+1\right)^2 + 1 & \text{if } -2 \leq x < 0 \\ c \cdot 2^{-8|x-1|} & \text{if } 0 \leq x \leq 2 \\ 0 & \text{otherwise} \end{cases}. \tag{A.36}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 0.1$. Unless stated otherwise, we will assume stationary noise. Furthermore, in experimental settings, the search interval is set to $\mathbf{x}_\mathrm{l} = [-2, \ldots, -2]$ and $\mathbf{x}_\mathrm{u} = [2, \ldots, 2]$.

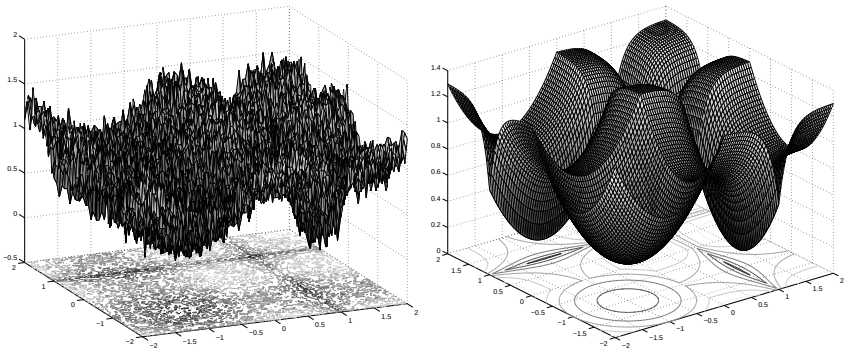


Figure A.9: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

A.10 Noisy Keane's Bump Problem

Keane's Bump function is a highly multi-modal test function, defined as

$$\min \qquad f(\mathbf{x}) = -\frac{|\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}(\cos^2(x_i))|}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}}, \tag{A.37}$$

$$\text{s.t.} \qquad \prod_{i=1}^{n} x_i > 0.75 \ , \ \sum_{i=1}^{n} x_i < \frac{15n}{2} \ , \ x_i \in]0, 10[. \tag{A.38}$$

The global minimizer for this function is unknown. The noisy version adopted in this work uses a penalty mechanism to aggregate the constraints in one objective function. It reads

$$f(\mathbf{x}) = g(\mathbf{x}) + \mathcal{N}(0, \sigma_\epsilon^2(\mathbf{x})), \tag{A.39}$$

$$g(\mathbf{x}) = \begin{cases} -\frac{|\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}(\cos^2(x_i))|}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}} & \text{if } \prod_{i=1}^{n} x_i > 0.75 \text{ and } \sum_{i=1}^{n} x_i < \frac{15n}{2} \\ 0 & \text{otherwise} \end{cases}. \tag{A.40}$$

In this work, stationary noise is assumed with $\sigma_\epsilon^2(\mathbf{x}) = 0.05$. Unless stated otherwise, we will assume stationary noise. Furthermore, in experimental settings, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\text{l} = [0, \ldots, 0]$ and $\mathbf{x}_\text{u} = [10, \ldots, 10]$.

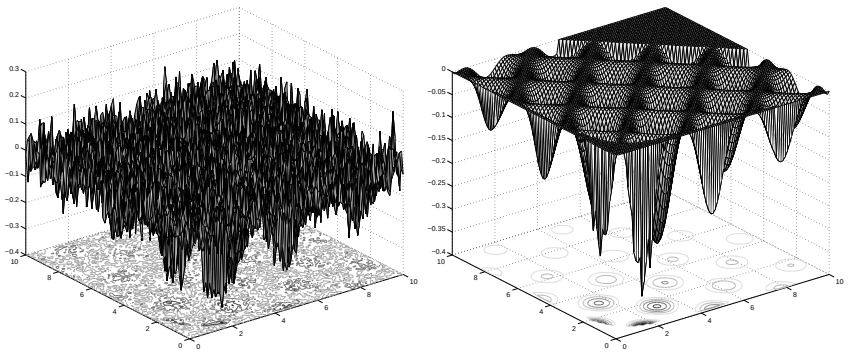


Figure A.10: Left: 2D visualization of the noisy objective function landscape. Right: 2D visualization of the noise-free underlying objective function landscape.

Appendix B

Test Problems for Finding Robust Optima

B.1 RO Sphere Problem

A simple unimodal test function for finding robust optima that can be used to assess the quality of a robust optimization algorithm w.r.t. zooming in on the robust optimum. For this function the robust optimizer and the optimizer of the original function are the same. The sphere function reads

$$f(\mathbf{x}) = \sum_{i=1}^{n} z_i^2 \quad , \quad \mathbf{z} = \mathbf{x} - \mathbf{x}^*, \tag{B.1}$$

with $\mathbf{x}^* \in \mathbb{R}^n$ being the location of the optimum. In this work, the optimum location is set to $\mathbf{x}^* = [0, \ldots, 0]$ and the search interval is set to $\mathbf{x}_\mathrm{l} = [-5, \ldots, -5]$ and $\mathbf{x}_\mathrm{u} = [5, \ldots, 5]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{1}, \mathbf{1}). \tag{B.2}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_\mathrm{exp} = \mathbf{x}^*. \tag{B.3}$$

Figure B.1 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

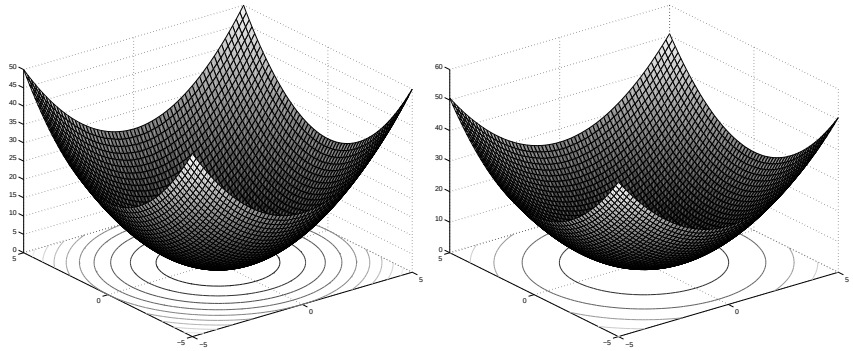


Figure B.1: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.2 RO Heaviside Sphere Problem

The Heaviside sphere function reads

$$f(\mathbf{x}) = \left(1 - \prod_{i=1}^{2} g(x_i)\right) + \sum_{i=1}^{n} \left(\frac{x_i}{10}\right)^2 \quad , \quad g(x_i) = \begin{cases} 0 & \text{if } x_i < 0 \\ 1 & \text{otherwise} \end{cases}, \tag{B.4}$$

with search interval $\mathbf{x}_\mathrm{l} = [-10, \ldots, -10]$ and $\mathbf{x}_\mathrm{u} = [10, \ldots, 10]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim U(-\mathbf{1}, \mathbf{1}). \tag{B.5}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_{\mathrm{exp}} = [1, \ldots, 1]^n. \tag{B.6}$$

Figure B.2 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

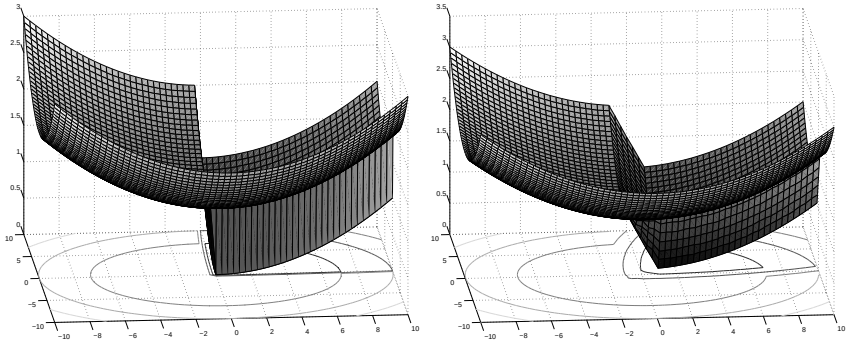


Figure B.2: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.3 RO Sawtooth Problem

The sawtooth function, originally proposed in [Bra01], reads

$$f(\mathbf{x}) = 1 - \frac{1}{n}\sum_{i=1}^{n} g(x_i) \quad , \quad g(x_i) = \begin{cases} x_i + 0.8 & \text{if } -0.8 \leq x_i < 0.2 \\ 0 & \text{otherwise} \end{cases}, \tag{B.7}$$

$$\tag{B.8}$$

with search interval $\mathbf{x}_\mathrm{l} = [-1, \ldots, -1]$ and $\mathbf{x}_\mathrm{u} = [1, \ldots, 1]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim U(-\mathbf{0.2}, \mathbf{0.2}). \tag{B.9}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_\mathrm{exp} = [0, \ldots, 0]^n. \tag{B.10}$$

Figure B.3 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

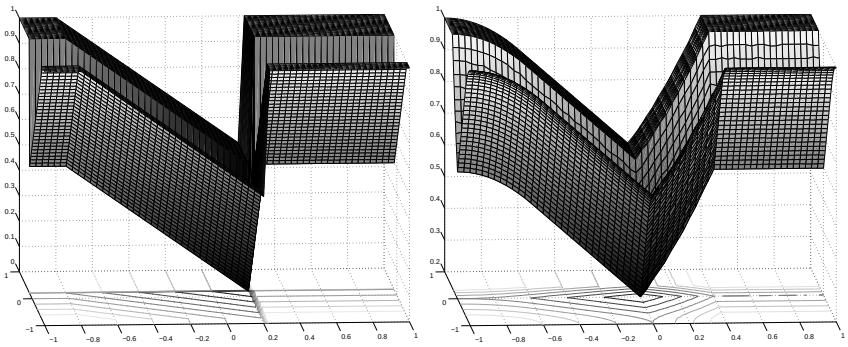


Figure B.3: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.4 RO Volcano Problem

The volcano function reads

$$f(\mathbf{x}) = \begin{cases} \sqrt{||\mathbf{x}||} - 1 & \text{if } ||\mathbf{x}|| > 1 \\ 0 & \text{otherwise} \end{cases}, \tag{B.11}$$

with search interval $\mathbf{x}_{\mathrm{l}} = [-10, \ldots, -10]$ and $\mathbf{x}_{\mathrm{u}} = [10, \ldots, 10]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim U(-\mathbf{1.5}, \mathbf{1.5}). \tag{B.12}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_{\mathrm{exp}} = [0, \ldots, 0]^n. \tag{B.13}$$

Figure B.4 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

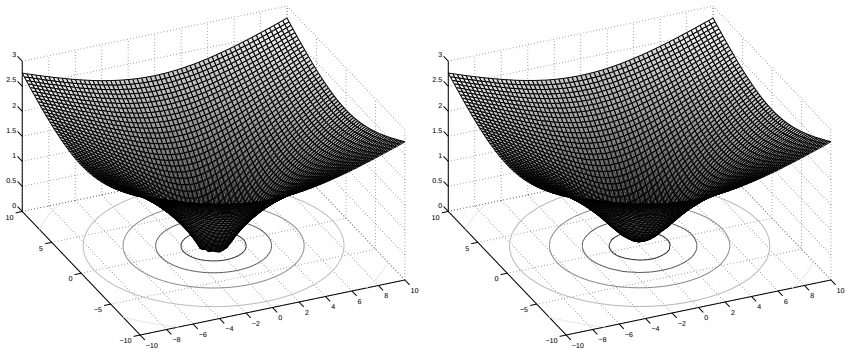


Figure B.4: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.5 RO Pickelhaube Problem

The objective function reads

$$f(\mathbf{x}) = \frac{5}{5-\sqrt{5}} - \max\left\{g_0(\mathbf{x}), g_{1a}(\mathbf{x}), g_{1b}(\mathbf{x}), g_2(\mathbf{x})\right\}, \tag{B.14}$$

$$g_0(\mathbf{x}) = \frac{1}{10} \cdot e^{-\frac{1}{2}\cdot||\mathbf{x}||}, \tag{B.15}$$

$$g_{1a}(\mathbf{x}) = \frac{5}{5-\sqrt{5}} \cdot \left(1 - \sqrt{\frac{||\mathbf{x}+\mathbf{5}||}{5\cdot\sqrt{n}}}\right), \tag{B.16}$$

$$g_{1b}(\mathbf{x}) = c_1 \cdot \left(1 - \left(\frac{||\mathbf{x}+\mathbf{5}||}{5\cdot\sqrt{n}}\right)^4\right), \tag{B.17}$$

$$g_2(\mathbf{x}) = c_2 \cdot \left(1 - \left(\frac{||\mathbf{x}+\mathbf{5}||}{5\cdot\sqrt{n}}\right)^{d_2}\right), \tag{B.18}$$

with $c_1 = 625/624$, $c_2 = 1.5975$, $d_2 = 1.1513$, and search interval $\mathbf{x}_\mathrm{l} = [-10, \ldots, -10]$ and $\mathbf{x}_\mathrm{u} = [10, \ldots, 10]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{1}, \mathbf{1}). \tag{B.19}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_{\mathrm{exp}} = [5, \ldots, 5]^n. \tag{B.20}$$

Figure B.5 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

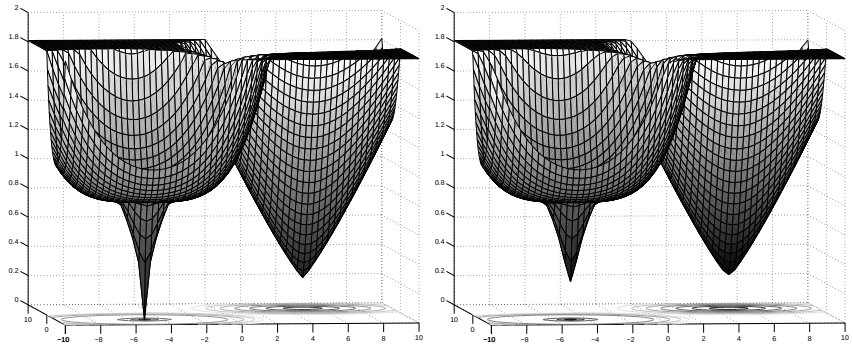


Figure B.5: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.6 RO Branke's Multipeak Problem

Objective function from [Bra98]. Originally posed as a maximization problem, converted to a minimization problem. The objective function reads

$$f(\mathbf{x}) = \max\{c_1, c_2\} - \frac{1}{n}\sum_{i-1}^{n} g(x_i), \tag{B.21}$$

$$g(x_i) = \begin{cases} c_1 \cdot \left(1 - \frac{4\left(x_i + \frac{b_1}{2}\right)^2}{(b_1)^2}\right) & \text{if } -b_1 \leq x_i < 0 \\ c_2 \cdot 16^{\frac{-2|b_2 - 2x_i|}{b_2}} & \text{if } 0 \leq x_i \leq b_2 \\ 0 & \text{otherwise} \end{cases}, \tag{B.22}$$

with $b_1 > 0$, $b_2 > 0$, $c_1 > 0$, $c_2 > 0$ and search interval $\mathbf{x}_\text{l} = [-b_1, \ldots, -b_1]$ and $\mathbf{x}_\text{u} = [b_2, \ldots, b_2]$. In this work, we use the settings $b_1 = 2$, $b_2 = 2$, $c_1 = 1$, $c_2 = 1.3$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.5}, \mathbf{0.5}). \tag{B.23}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_\text{exp} = [-b_1/2, \ldots, -b_1/2]^n. \tag{B.24}$$

Figure B.6 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

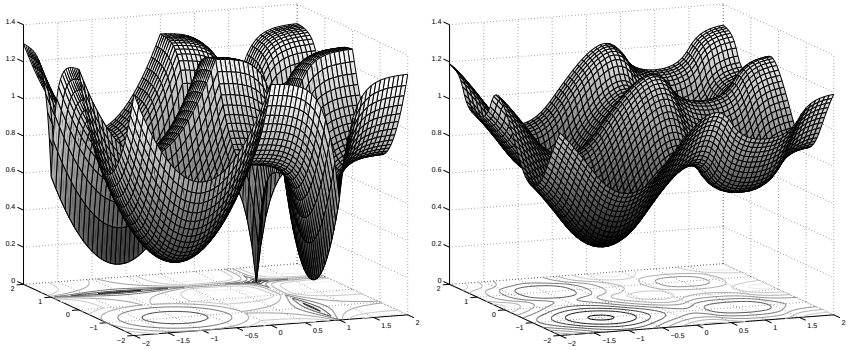


Figure B.6: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.7 RO Multipeak F1 Problem

A multimodal test problem used in, e.g., [TGF96, TG97]. Originally posed as maximization problem, converted to minimization. The objective function reads

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x}), \tag{B.25}$$

$$g(x_i) = \begin{cases} e^{-2\ln 2\left(\frac{x-0.1}{0.8}\right)^2}\sqrt{|\sin(5\pi x_i)|}, & \text{if } 0.4 < x_i \leq 0.6 \\ e^{-2\ln 2\left(\frac{x-0.1}{0.8}\right)^2}\sin^6(5\pi x_i), & \text{otherwise} \end{cases}, \tag{B.26}$$

with search interval $\mathbf{x}_\mathrm{l} = [0,\ldots,0]$ and $\mathbf{x}_\mathrm{u} = [1,\ldots,1]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.0625}, \mathbf{0.0625}). \tag{B.27}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_{\mathrm{exp}} \approx [0.4911,\ldots,0.4911]^n. \tag{B.28}$$

Figure B.7 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

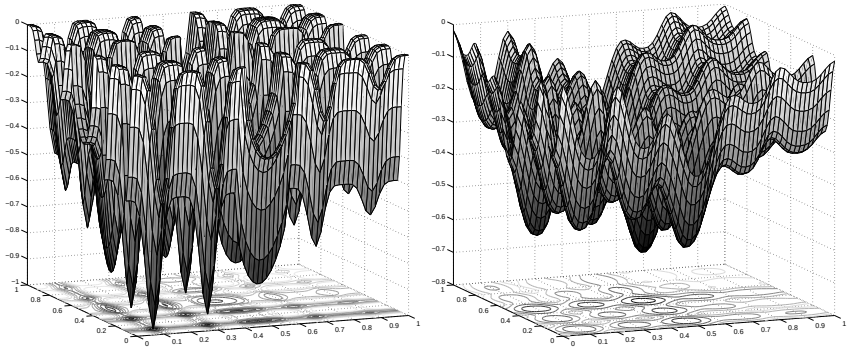


Figure B.7: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

B.8 RO Multipeak F2 Problem

A multimodal test problem used in, e.g., [PBJ06]. Originally posed as maximization problem, converted to minimization. The objective function reads

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x}), \tag{B.29}$$

$$g(x_i) = 2\sin\left(10\exp(-0.2x_i)x_i\right)\exp\left(-0.25x_i\right), \tag{B.30}$$

with search interval $\mathbf{x}_\mathrm{l} = [0,\ldots,0]$ and $\mathbf{x}_\mathrm{u} = [10,\ldots,10]$. The uncertainty in the design variables is of the form

$$\mathbf{x} = \mathbf{x} + \boldsymbol{\delta} \quad , \quad \boldsymbol{\delta} \sim \mathcal{U}(-\mathbf{0.5}, \mathbf{0.5}). \tag{B.31}$$

This function has as robust optimizer for the expected objective function

$$\mathbf{x}^*_\mathrm{exp} \approx [3.5,\ldots,3.5]^n. \tag{B.32}$$

Figure B.8 shows 2D visualizations of the original objective function landscape and the effective objective function landscape.

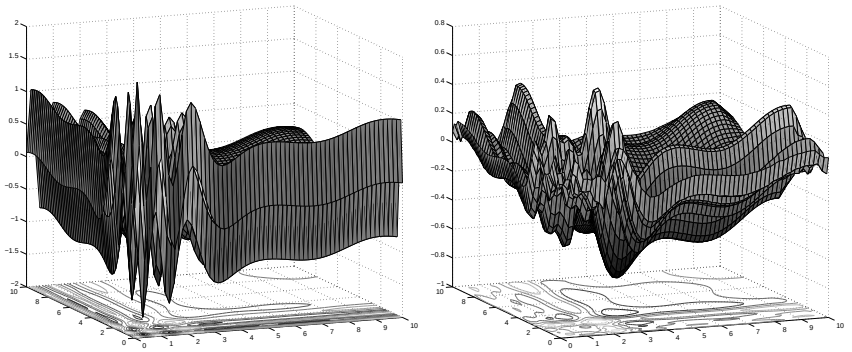


Figure B.8: Left: 2D visualization of the original objective function landscape. Right: 2D visualization of the effective objective function landscape.

Appendix C

Kriging Metamodeling

Kriging Metamodeling

A basic assumption of function modeling through Kriging is that the deviation of a function from a general trend or mean value can be modeled as a realization of a Gaussian random field $\mathcal{F}_{\mathbf{x}}$, $\mathbf{x} \in \mathbb{R}^n$, where $\mathcal{F}_{\mathbf{x}}$ is a Gaussian random variable indexed by space. Two random variables $\mathcal{F}_{\mathbf{x}}$ and $\mathcal{F}_{\mathbf{x}'}$ are correlated via a spatial correlation function $c(\mathbf{x}, \mathbf{x}')$. Often $c(\mathbf{x}, \mathbf{x}')$ is based on the difference $\mathbf{x} - \mathbf{x}'$ (stationary correlation) or on the distance $d(\mathbf{x}, \mathbf{x}')$ (isotropic correlation). When c is fully specified, then it is possible to compute the conditional distribution of $\mathcal{F}_{\mathbf{x}}$ for a new point $\mathbf{x}$ given a number of measured realizations $y_1 = \mathcal{F}_{\mathbf{x}_1}, \ldots, y_m = \mathcal{F}_{\mathbf{x}_m}$. The mean value of the conditional distribution can be interpreted as a predictor for the function value $f(\mathbf{x})$ and the standard deviation of $\mathcal{F}_{\mathbf{x}}$ can be interpreted as a measure of prediction uncertainty. *Ordinary Kriging*, used in this work, assumes a constant trend β. It thus estimates $f(\mathbf{x})$ as $\mathcal{F}_{\mathbf{x}} = \beta + \mathcal{R}_{\mathbf{x}}$, where $\mathcal{R}_{\mathbf{x}}$, $\mathbf{x} \in \mathbb{R}^n$ is a Gaussian random field model with mean zero and global variance s^2. The Kriging predictor $\hat{f}(\mathbf{x})$ for an unknown point $\mathbf{x}$ is

$$\hat{f}(\mathbf{x}) = \beta + (\mathbf{y} - \mathbf{1} \cdot \beta)^T \cdot \mathbf{C}^{-1} \cdot \mathbf{c}(\mathbf{x}), \tag{C.1}$$

with $\mathbf{y} = [y_1, \ldots, y_m]^T$, $\mathbf{C} = [c(\mathbf{x}_i, \mathbf{x}_j)]_{i=1,\ldots,m, j=1,\ldots m}$, and $\mathbf{c}(\mathbf{x}) = [c(\mathbf{x}, \mathbf{x}_1), \ldots, c(\mathbf{x}, \mathbf{x}_n)]^T$. Here, β and s^2 are estimated using generalized least square estimates (see, [JSW98]),

$$\hat{\beta} = \frac{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{y}}{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{1}}, \tag{C.2}$$

$$\hat{s} = \frac{(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{y} - \mathbf{1} \cdot \hat{\beta})}{m}. \tag{C.3}$$

The correlation function $c(\mathbf{x}, \mathbf{x}')$ can have different shapes. The choice of the correlation function can be based on a-priori knowledge or on the correlation structure, e.g., maximum likelihood estimation. Typically, isotropic kernel functions are of the form

$$c_\theta(\mathbf{x}, \mathbf{x}') = \exp\left(-\theta \cdot |\mathbf{x} - \mathbf{x}'|^q\right) \ , q > 0. \tag{C.4}$$

Setting θ is done by maximizing the likelihood of $\mathcal{F}_{\mathbf{x}_1} = \mathbf{y}_1 \wedge \ldots \wedge \mathcal{F}_{\mathbf{x}_m} = \mathbf{y}_m$ by minimizing

$$m \log \hat{s}(\theta) + \log \det \mathbf{C}(\theta). \tag{C.5}$$

The same principle can be applied for multiparametric kernels, though this requires multi-dimensional optimization demanding considerably more evaluations of the log-likelihood expression.

The calibration procedure is algorithmically described in Algorithm C.1. In this description, the minimization of θ is omitted. In this work, a grid search method is adopted using a logarithmically scaled grid on the interval $[10^{-50}, 10^5]$.

Advantages of Kriging are that it is an exact interpolator (i.e., it returns the sample value as the estimate at the sample points) and that the prediction confidence range can be locally assessed.

Algorithm C.1: Ordinary Kriging Calibration

Input: Archive $A = \{(\mathbf{x}_1, f_1), \ldots, (\mathbf{x}_m, f_m)\}$

Output: Kriging model $f_{\text{krig}}(\mathbf{x})$

1: **Set parameters:** $q \leftarrow 2$, minimization method for θ

2: $\theta \leftarrow \min_\theta \{m \log \hat{s}(\theta) + \log\det \mathbf{C}(\theta)\}$, with

$$\mathbf{C} \leftarrow \begin{bmatrix} c_\theta(\mathbf{x}_1, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_1, \mathbf{x}_m) \\ \vdots & \ddots & \vdots \\ c_\theta(\mathbf{x}_m, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}, \; c_\theta(\mathbf{x}, \mathbf{x}') = \exp\left(-\theta \cdot |\mathbf{x} - \mathbf{x}'|^q\right),$$

$$\hat{s} = \frac{(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{y} - \mathbf{1} \cdot \hat{\beta})}{m}.$$

3: $f_{\text{krig}}(\mathbf{x}) \leftarrow \hat{\beta} + (\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \cdot \mathbf{C}^{-1} \cdot \mathbf{c}(\mathbf{x})$, with

$$\hat{\beta} = \frac{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{y}}{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{1}}, \; \mathbf{c}(\mathbf{x}) = [c_\theta(\mathbf{x}, \mathbf{x}_1), \ldots, c_\theta(\mathbf{x}, \mathbf{x}_n)]^T.$$

4: **return** $f_{\text{krig}}(\mathbf{x})$

A disadvantage is the computational effort (repeated inversion and determinant computation for $\mathbf{C}(\theta)$ within the likelihood minimization over θ). Besides, the points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ do not only have to be unique so that the $\mathbf{C}$ is positive definite, but one should also ensure that these points are well distributed to prevent that $\mathbf{C}$ gets ill-conditioned, which could cause failure due to numerical errors.

Samenvatting (Dutch)

Evolutionaire algoritmen zijn door natuurlijke evolutie geïnspireerde algoritmen voor het oplossen van complexe optimalisatieproblemen. Door het simuleren van evolutie worden binnen deze klasse van algoritmen populaties van kandidaatoplossingen gekweekt tot (sub)optimale oplossingen. Hierbij wordt de evolutionaire "fitness" van een kandidaatoplossing bepaald door de kwaliteit ten opzichte van het optimalisatieprobleem.

Praktische optimalisatieproblemen zijn vaak onderhevig aan onzekerheid en ruis. Wanneer hiermee geen rekening wordt gehouden, kan optimalisatie falen of leiden tot oplossingen die onbruikbaar zijn in de praktijk. Robuuste optimalisatie is de praktijk van optimalisatie waarbij actief rekening gehouden wordt met onzekerheid en ruis. Het doel van dit onderzoek is het afbakenen van het begrip robuuste optimalisatie en het bestuderen hoe evolutiestrategieën, een subklasse van evolutionaire algoritmen voor reële parameter optimalisatie, zich gedragen in robuuste optimalisatie scenario's of hoe deze hiervoor moeten worden aangepast.

Robuuste optimalisatie beslaat alle soorten van onzekerheid en ruis die voor kunnen komen binnen het model of systeem dat wordt beschouwd voor optimalisatie. Het behelst echter niet eventuele onzekerheden in de formulering van de doelen en randvoorwaarden. Het doel is tweeledig: het vinden van optimale oplossingen ondanks dat onzekerheden en ruis de optimalisatie bemoeilijken en het vinden van optimale oplossingen die robuust zijn ten opzichte van onzekerheden en ruis. Robuuste optimalisatie vereist de integratie van de notie van robuustheid in de specificatie van de kwaliteit van oplossingen. Dit zijn de zogenaamde effectieve doelfuncties en randvoorwaarden.

De verschillende soorten van onzekerheid en ruis die kunnen bestaan binnen een optimalisatieprobleem zorgen voor een combinatorische explosie van verschillende scenario's voor robuuste optimalisatie. Echter, sommige scenario's komen vaker voor dan andere. In dit proefschrift zijn twee scenarios's eruit gelicht: optimalisatie van systemen met ruis en het vinden van robuuste optima.

Optimalisatie van systemen met ruis vereist een kwaliteitsmaat die de notie van robuustheid ten opzichte van de ruis omvat: een effectieve doelfunctie. Uitgaande van de verwachtingswaarde van de kwaliteit (de verwachte doelfunctie) van kandidaatoplossingen zijn evolutiestrategieën robuust wanneer de ruis relatief klein is. Wanneer er echter een hoge convergentieprecisie vereist is, zijn extra aanpassingen nodig.

Impliciet en expliciet middelen zijn simpele technieken om de convergentieprecisie van evolutiestrategieën te vergroten. Impliciet middelen is de praktijk van het vergroten van de populatiegrootte en expliciet middelen betreft het evalueren van de kwaliteit van kandidaatoplosssingen door het middelen over meerdere evaluaties. Het nadeel van deze twee technieken is dat ze een a priori specificatie van een populatiegrootte of het aantal evaluaties voor middeling vereisen en dat de convergentieprecisie nog steeds beperkt is. Adaptieve middelingstechnieken zijn uitbreidingen die de evaluatie-intensiteit automatisch proberen aan te passen (dus, te vergroten) gedurende de optimalisatie.

In dit proefschrift beschouwen we adaptieve middelingstechnieken die zijn gebaseerd op expliciet middelen. Hiervoor is voor een simpel testprobleem aangetoond dat een exponentieel groeiend aantal evaluaties per kandidaatoplossing nodig is om lineaire convergentie ten opzichte van het aantal generaties te bereiken. Een empirische studie laat zien dat een onzekerheidsmaat gebaseerd op rangverschillen de meestbelovende methode is voor het kwantificeren van onzekerheid. Daarnaast is aangetoond dat adaptieve middelingstechnieken vergelijkbare resultaten kunnen opleveren als optimaal ingestelde statische ruisbehandelingsmethoden. Voor één scenario geldt dit echter niet; een optimaal ingestelde impliciete middelingsmethode in de CMA-ES werkt beter dan alle andere geteste evaluatietechnieken voor doelfuncties met ruis. Omdat ze minder gevoelig zijn voor parameterinstellingen zijn adaptieve middelingstechnieken een goed alternatief voor impliciete en expliciete middeling.

Het vinden van robuuste optima is van belang wanneer kandidaatoplossingen niet exact gerealiseerd kunnen worden, maar afwijken of fluctueren. Van de verschillende effectieve doelfuncties die voor optimalisatie mogelijk zijn is de verwachte kwaliteit een veelgebruikte maat. De moeilijkheid van zulke scenario's is dat exacte evaluatie van de effectieve doelfunctie vaak onmogelijk is en er daarom benaderingsmethoden nodig zijn voor de bepaling van de robuuste kwaliteit van kandidaatoplossingen. Diverse methoden voor het vinden van robuuste optima kunnen worden onderscheiden.

De simpelste methode voor het vinden van robuuste optima is om niets te doen en erop te vertrouwen dat evolutiestrategieën vanuit zichzelf al convergeren naar de robuuste pieken in het functielandschap. Deze myopische methode wordt gesteund door de observatie dat evolutionaire algoritmen een inherente neiging hebben om naar de robuustere pieken te convergeren, maar faalt wanneer er sprake is van een verschoven robuust optimum.

Monte-Carlo integratietechnieken kunnen worden gebruikt om de robuustheid van kandidaatoplossingen te benaderen. Dit verschuift het probleem in de richting van het optimaliseren van doelfuncties met ruis. Hoewel deze technieken de effectieve doelfunctie benaderen zijn ze beperkt in precisie en daarmee beperken ze de convergentieprecisie van evolutiestrategieën. Net als bij doelfuncties met ruis kan ook voor het vinden van robuuste optima gebruik gemaakt worden van adaptieve middelingstechnieken. Dit heeft als voordeel dat het de convergentieprecisie niet beperkt, noch een a priori instelling van het aantal evaluaties voor Monte-Carlo

benadering vereist.

Een andere klasse van methoden wordt gevormd door archief- en metamodelingmethoden. Deze slaan eerder geëvalueerde kandidaatoplossingen op om ze te gebruiken voor het benaderen van de kwaliteit van nieuwe kandidaatoplossingen. Doordat ze efficiënt omgaan met functie-evaluaties zijn deze methoden in het bijzonder bruikbaar wanneer functie-evaluaties (computationeel) duur zijn. Een methode voor het bijhouden van een archief van kandidaatoplossingen is hiervoor vereist om ervoor te zorgen dat het archief bruikbaar is voor het verkrijgen van betrouwbare benaderingen voor nieuwe kandidaatoplossingen. Het gebruiken van metamodeling technieken, zoals Kriging, op basis van dit archief is een uitbreiding hierop.

Het idee om gebruik te maken van niching technieken voor het vinden van robuuste optima heeft het veronderstelde voordeel dat de optimalisatie zich richt op verschillende gebieden van de zoekruimte. Hoewel dit idee theoretisch zinnig is introduceert een directe integratie van een standaard niching strategie meer problemen dan het oplost. De resultaten gepresenteerd in dit proefschrift laten zien dat voor deze doeleinden een niching strategie vereist is die om kan gaan met doelfuncties met ruis.

Een laatste techniek die gebruikt kan worden om de evaluatieprecisie voor het vinden van robuuste optima te vergroten is het uitbuiten van de overlap van de onzekerheidsgebieden bij het vergelijken van paren van kandidaatoplossingen. In plaats van te proberen om zo precies mogelijke benaderingen te krijgen van de robuuste kwaliteit van kandidaatoplossingen wordt er in deze techniek gekeken naar hoe paren van oplossingen zich tot elkaar verhouden. De overlap van de gebieden van onzekerheid kan hierbij vaak worden uitgesloten van evaluatie.

Een empirisch vergelijkende studie laat zien dat voor evolutiestrategieën een adaptieve middelingsstrategie de meestbelovende strategie is voor het vinden van robuuste oplossingen. Hierbij moet voor evaluatie gebruik gemaakt worden van Latin Hypercube sampling en moet voor iedere generatie dezelfde set van perturbaties gebruikt worden voor alle individuen in de populatie. In vergelijking met statische evaluatiemethoden leidt dit tot betere robuuste oplossingen over het hele spectrum van benchmarkproblemen. De archief-gebaseerde methode en de metamodeling methode leiden vooral op bepaalde testproblemen tot goede oplossingen.

De bijdrage van dit onderzoek bestaat uit een aantal onderdelen. Allereerst is er een algemene definitie gegeven voor de term “robuuste optimalisatie” waarmee wordt afgebakend wat er wel en niet onder deze noemer valt. Daarnaast wordt er een algemene wijze van aanpak voor robuuste optimalisatieproblemen gegeven aan de hand van twee representatieve scenario’s: het optimaliseren van systemen met ruis en het vinden van robuuste optima. Voor deze twee scenario’s wordt in detail onderzocht hoe evolutiestrategieën zich gedragen en hoe deze aangepast moeten worden voor het vinden van robuuste optima. Hierbij worden zowel technieken uit de literatuur beschouwd als nieuwe methoden voorgesteld. Als laatste worden er in dit proefschrift op basis van conceptuele overwegingen en empirische vergelijkingen aanbevelingen gegeven voor het toepassen van evolutiestrategieën voor robuuste optimalisatie.

Curriculum Vitae

Johannes Willem Kruisselbrink was born on July 4th 1983 in Nijmegen, the Netherlands. In 2002 he started the study Computer Science at Rijksuniversiteit Groningen where he received his BSc in 2006. The research of the Natural Computing Group of prof. T. Bäck inspired him to move to Leiden in 2006. He received his MSc degree (with distinction) in Computer Science in 2008 at Leiden University. Right after, he joined the Natural Computing Group as a PhD student on the NWO-funded project "Robust Design Optimization" (RODEO). Besides his main research on evolution strategies for robust optimization problems, he also participated in joint research with the group of prof. A. IJzerman of LACDR, Leiden University on molecular design using evolutionary algorithms.